6.300: Signal Processing

Review #1: Retrospective

Titus K. Roesler `tkr@mit.edu`

Please fill out a subject evaluation for this class at `https://registrar.mit.edu/classes-grades-evaluations/subject-evaluation` — especially if you appreciate the work of the teaching staff!

May 14, 2025

End-of-Term Schedule

05/14 (Wed.)	Review #1	4-270	2:00 – 4:00 PM
	Office Hour	4-270	4:00 – 5:00 PM
05/15 (Thu.)	Office Hour	36-112	4:00 – 5:00 PM
05/18 (Sun.)	Review #2	4-270	2:00 – 4:00 PM
	Office Hour	4-270	4:00 – 5:00 PM
05/19 (Mon.)	Office Hours	34-304	7:30 – 9:30 PM
05/20 (Tue.)	Office Hours	34-304	7:30 – 9:30 PM
05/21 (Wed.)	Final Exam	Johnson	1:30 – 4:30 PM

Review #1: We will review the key concepts and applications, discuss practical problem-solving strategies, and work through toy problems that highlight these strategies.

Review #2: We will solve problems from past exams.

The Signal Processing Story

Fourier Series and Fourier Transforms

02/04	Signal Processing
02/06	Fourier Series: Sinusoids
02/11	Fourier Series: Complex Exponentials
02/13	Discretization: Sampling and Quantization
02/20	Discrete-Time Fourier Series
02/25	Continuous-Time Fourier Transform
02/27	Discrete-Time Fourier Transform
03/04	Quiz #1

Why focus on Fourier?
Fourier relations play important roles in many branches of physics — especially those concerning wave phenomena.

The Signal Processing Story

LTI Systems

03/06 Systems: Linearity and Time-Invariance
03/11 Impulse Response and Convolution
03/13 Frequency Response and Filtering

Discrete Fourier Transform

03/18 DFT: Connections to DTFS and DTFT
03/20 DFT: Resolution and Circular Convolution
04/01 Short-Time Fourier Transforms
04/03 Fast Fourier Transform (FFT)

The **discrete Fourier transform (DFT)** is the Fourier transform used in digital signal processing.

The Signal Processing Story

Applications and Extensions

04/08	Modulation and Communications Systems
04/10	Quiz Review
04/15	Quiz #2
04/17	Speech
04/22	2D Fourier Transforms
04/24	2D Fourier Transforms
04/29	Inverse Filtering (Deconvolution)
05/01	Compression: Discrete Cosine Transform
05/06	Magnetic Resonance Imaging (MRI)
05/08	Synthetic Aperture Optics
05/13	Music Processing (Prof. Eran Egozy) and Portable, Low-Cost MRI (Prof. Jacob White)

Fourier Transforms (CT)

By now, you should know these by heart — or at least have them on your cheat sheet so you can quickly reference them.

$$\begin{aligned} \delta(t) &\iff 1 \\ \delta(t-t_0) &\iff e^{-j\omega t_0} \\ 1 &\iff 2\pi\delta(\omega) \\ e^{j\omega_0 t} &\iff 2\pi\delta(\omega-\omega_0) \end{aligned}$$

Duality: Notice the common trend in transform pairs.

$$\begin{aligned} x(t) &\iff X(\omega) \\ X(t) &\iff 2\pi x(-\omega) \end{aligned}$$

Fourier Transforms (DT)

All discrete-time Fourier transforms are 2π-periodic.

$$\begin{aligned}
\delta[n] &\iff 1 \\
\delta[n-n_0] &\iff e^{-j\Omega n_0} \\
1 &\iff 2\pi\delta(\Omega \bmod 2\pi) \\
e^{j\Omega_0 n} &\iff 2\pi\delta\Big((\Omega-\Omega_0) \bmod 2\pi\Big)
\end{aligned}$$

Duality: Not so easy with discrete-time Fourier transforms. $x[n]$ is discrete in time, but $X(\Omega)$ is continuous in frequency!

Fourier Transforms (DFT)

1D DFTs are periodic in N. 2D DFTs are periodic in RC.

$$
\begin{aligned}
\delta[n] &\iff \frac{1}{N} \\
\delta[n-n_0] &\iff \frac{1}{N} e^{-jk\frac{2\pi}{N}n_0} \\
1 &\iff \delta[k] \\
e^{jk_0\frac{2\pi}{N}n} &\iff \delta[k-k_0] \\
\delta[r,c] = \delta[r]\delta[c] &\iff \frac{1}{RC} \\
\delta[r-r_0, c-c_0] &\iff \frac{1}{RC} e^{-j(k_r\frac{2\pi}{R}r_0 + k_c\frac{2\pi}{C}c_0)} \\
1 &\iff \delta[k] \\
e^{j(k'_r\frac{2\pi}{R}r + k'_c\frac{2\pi}{C}c)} &\iff \delta[k_r-k'_r, k_c-k'_c]
\end{aligned}
$$

Fourier Properties

$$
\begin{aligned}
c_1 x_1[n] + c_2 x_2[n] &\iff c_1 X_1(\Omega) + c_2 X_2(\Omega) \\
(x_1 * x_2)[n] &\iff X_1(\Omega) X_2(\Omega) \\
\tfrac{1}{N}(x_1 \circledast x_2)[n] &\iff X_1[k] X_2[k] \\
x_1[n] x_2[n] &\iff \tfrac{1}{2\pi}\Big(X_1 * X_2\Big)(\Omega) \\
x[-n] &\iff X(-\Omega) \\
x[nM] &\iff X(\tfrac{\Omega}{M}) \\
x[n - n_0] &\iff e^{-j\Omega n_0} X(\Omega) \\
e^{j\Omega_0 n} x[n] &\iff X(\Omega - \Omega_0) \\
n\, x[n] &\iff j\tfrac{d}{d\Omega} X(\Omega) \\
\tfrac{d}{dt} x(t) &\iff j\omega X(\omega)
\end{aligned}
$$

Fourier Properties

Use Fourier properties to cut down the number of calculations.

Example: Time Shift

$$x_1[n] = x_0[n - n_0] \implies X_1(\Omega) = e^{-j\Omega n_0} X_0(\Omega)$$

Example: Add a Time-Shifted Copy

$$x_2[n] = x_0[n] + x_0[n - n_0]$$

$$X_2(\Omega) = (1 + e^{-j\Omega n_0}) X_0(\Omega) = A(\Omega) e^{j\phi(\Omega)} X_0(\Omega)$$

$$\text{where } A(\Omega) = 2\cos\left(\tfrac{n_0}{2}\Omega\right) \text{ and } \phi(\Omega) = -\tfrac{n_0}{2}\Omega$$

Sinusoids and Symmetry

Recall definitions of symmetry and anti-symmetry.

$$\underbrace{x_s[n] = x_s[-n]}_{\text{symmetric}} \qquad \underbrace{x_a[n] = -x_a[-n]}_{\text{anti-symmetry}}$$

Time-domain symmetry is preserved in frequency.

$$\delta[n+1] + \delta[n-1] \iff e^{j\Omega} + e^{-j\Omega} = 2\cos(\Omega)$$

$$\delta[n+1] - \delta[n-1] \iff e^{j\Omega} - e^{-j\Omega} = 2j\sin(\Omega)$$

Conjugate Symmetry (Hermitian Symmetry)

Suppose $x[n]$ is a real signal.

- symmetric $x[n]$ — real, symmetric $X(\Omega)$
- anti-symmetric $x[n]$ — imaginary, anti-symmetric $X(\Omega)$

Sinusoids and Symmetry

The phase of a complex number $z = a + jb$ is the angle that z makes with respect to the positive real axis.

$$\tan(\angle z) = \frac{\text{Im}\{z\}}{\text{Re}\{z\}} = \frac{b}{a}$$

For a purely real number a, $\angle a = 0$ or $\angle a = \pi$.

- $a = 0 \implies \angle a = 0$ — origin
- $a > 0 \implies \angle a = 0$ — directed along the positive real axis
- $a < 0 \implies \angle a = \pi$ — directed along the negative real axis

So, if the Fourier transform $X(\Omega)$ is purely real, then ...

- $X(\Omega) \geq 0 \implies \angle X(\Omega) = 0$
- $X(\Omega) < 0 \implies \angle X(\Omega) = \pi$

Sinusoids and Symmetry

Linear Phase Factorization

We may factor the Fourier transform of a real signal $x[n]$ symmetric (or anti-symmetric) about $n = n_0$ as follows.

$$X(\Omega) = A(\Omega)e^{j\phi(\Omega)}$$

$A(\Omega)$ is an amplitude term, and $\phi(\Omega)$ is a phase term.

- $A(\Omega)$ is real and symmetric for symmetric $x[n]$.
- $A(\Omega)$ is imaginary and anti-symmetric for anti-symmetric $x[n]$.
- $\phi(\Omega)$ is linear: $\phi(\Omega) = m\Omega$ where $m = -n_0/2$.

$$|X(\Omega)| = |A(\Omega)| \qquad \angle X(\Omega) = \angle A(\Omega) + \phi(\Omega)$$

Sinusoids and Symmetry

Example: Linear Phase Factorization

A signal symmetric about an integer or half-integer $n = n_0$ is a mere shift away from being symmetric about $n = 0$.

$$x[n] = x_{\text{sym}}[n - n_0] \iff X(\Omega) = X_{\text{sym}}(\Omega)e^{-j\Omega n_0}$$

$x[n] = \delta[n] + \delta[n-2]$ is symmetric about $n = 1$

$$x_{\text{sym}}[n] = x[n+1] = \delta[n+1] + \delta[n-1]$$

$$X(\Omega) = 1 + e^{-j2\Omega} = e^{-j\Omega}(e^{j\Omega} + e^{-j\Omega}) = 2\cos(\Omega)e^{-j\Omega}$$

$$X_{\text{sym}}(\Omega) = 2\cos(\Omega) \implies X(\Omega) = X_{\text{sym}}(\Omega)e^{-j\Omega}$$

Sinusoids and Symmetry

Sketch the signal $h[n] = \delta[n] - \delta[n-1]$.

Intuitively, without performing any calculations, would you expect $h[n]$ to function more as a low-pass filter or more as a high-pass filter? Explain your reasoning.

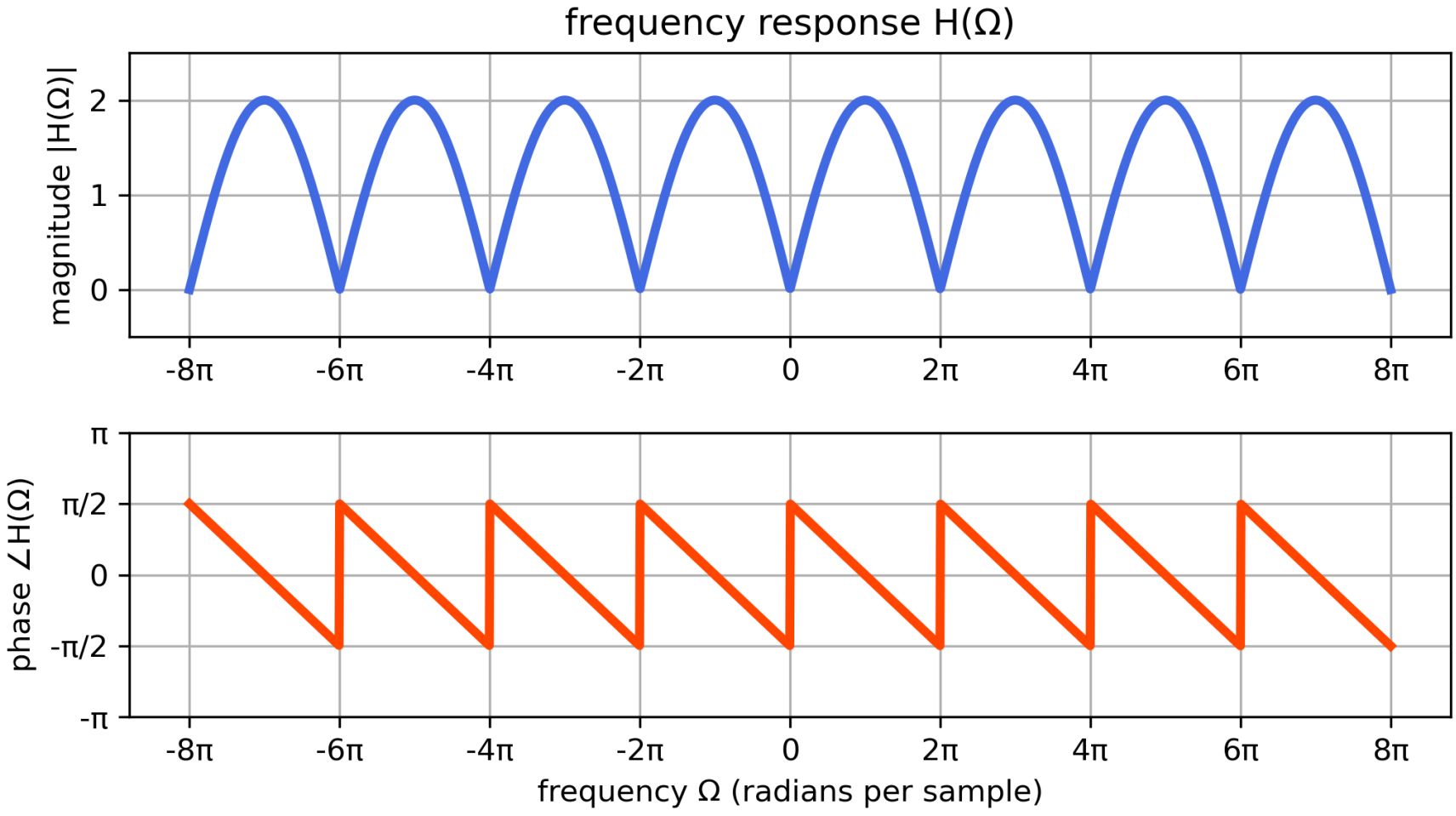
Two equivalent interpretations:

- Time: Search for similar-looking signal — "matched filter."
- Frequency: Shape the frequency spectrum.

Now, let's check with some calculations.

- Compute the DTFT $H(\Omega)$.
- Sketch the magnitude $|H(\Omega)|$.
- Sketch the phase $\angle H(\Omega)$.

Sinusoids and Symmetry



Fast Fourier Transforms

Determine the DTFT for each of the following signals. Recall that $u[n]$ denotes the unit-step function.

(a) $p[n] = u[n]u[N-1-n]$ where $N > 0$ is a constant

(b) $q[n] = u\left[n + \frac{N-1}{2}\right] u\left[\frac{N-1}{2} - n\right]$

(c) $r[n] = q[n_0 - n]$ where n_0 is a constant

(d) $s[n] = \frac{1}{4}r[n+1] + \frac{1}{2}r[n] + \frac{1}{4}r[n-1]$

You may express

- $Q(\Omega)$ in terms of $P(\Omega)$,
- $R(\Omega)$ in terms of $P(\Omega)$ and/or $Q(\Omega)$, and
- $S(\Omega)$ in terms of $P(\Omega)$, $Q(\Omega)$, and/or $R(\Omega)$.

Hint: Apply Fourier transform properties.

LTI Systems

LTI Systems

We examined three representations for LTI systems:

- difference equation (DT) or differential equation (CT)
- unit-sample response (DT) or impulse response (CT)
- frequency response

You should be able to quickly switch between these three representations as needed.

The slides from the quiz #2 review session might be a good reference if you're looking to brush up on these equivalent representations of LTI systems. Today, I want to think about **inverse systems**. (Lecture 12A: "Inverse Filtering")

LTI Systems

The unit-sample response (or impulse response) is the output $y[n] = h[n]$ corresponding to the input $x[n] = \delta[n]$.

$$x[n] = \delta[n] \rightarrow \boxed{\textbf{LTI}} \rightarrow y[n] = h[n]$$

Here is the corresponding inverse system: When the input is $x[n] = h[n]$, the output is $y[n] = \delta[n]$.

$$x[n] = h[n] \rightarrow \boxed{\textbf{LTI}} \rightarrow y[n] = \delta[n]$$

Let $H(\Omega)$ denote the frequency response of an LTI system. The frequency response of the corresponding inverse system is $H(\Omega)^{-1}$ — if such an inverse system exists.

The inverse system may not exist if $H(\Omega_0) = 0$ for some Ω_0. $H(\Omega)^{-1}$ would "blow up" as $\Omega \rightarrow \Omega_0$.

LTI Systems

Define $p[n]$ and $x[n]$ as follows.

$$p[n] = \begin{cases} A & n = 0 \\ B & n = 1 \\ A & n = 2 \\ 0 & \text{otherwise} \end{cases}$$

$$x[n] = p[n] + \alpha p[n-N] + \alpha^2 p[n-2N] + \cdots = \sum_{m=0}^{\infty} \alpha^m p[n-mN]$$

Determine the frequency response of the LTI system below.

$$x[n] = \sum_{m=0}^{\infty} \alpha^m p[n-mN] \rightarrow \boxed{\textbf{LTI}} \rightarrow y[n] = \delta[n]$$

Sampling and Aliasing

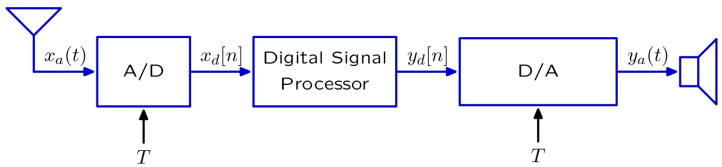
The "question of the day" often involved sampling — even long after the lecture that introduced the notion of sampling.

- Aliasing is a key concern in digitization.
- Whenever you simulate a continuous-time system with a discrete-time model, you better think a bit about sampling.

Question of the Day

Commercial AM radio

- 106 channels
- each channel is allocated 10 kHz bandwidth
- center frequencies from 540 kHz to 1600 kHz

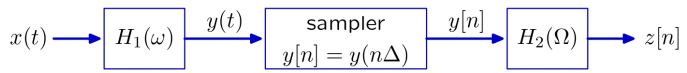


We would like to choose the sampling period T so that $y_a(t)$ can be tuned to receive any of the 106 possible channels by simply changing the program in the digital signal procesor (without changing T).

What is the maximum value of T for which this is possible?

Question of the Day

A continuous-time signal $x(t)$ is passed through an anti-aliasing filter $H_1(\omega)$ to produce a new continuous-time signal $y(t)$.
The new signal $y(t)$ is sampled with sampling interval Δ to produce a discrete-time signal $y[n] = y(n\Delta)$ where $\Delta =$ **1 millisecond**.
Then $y[n]$ is passed through a low pass filter $H_2(\Omega)$ to remove any aliased components introduced by the sampling.



Determine cutoff frequencies ω_L for $H_1(\omega)$ and Ω_L for $H_2(\Omega)$ so that $z[n]$ preserves as many frequencies in $x(t)$ as possible while removing any components that would alias.

Sampling and Aliasing

Sampling a CT signal $x(t)$ yields a DT signal $x[n]$.

$$x[n] = x(n\Delta)$$

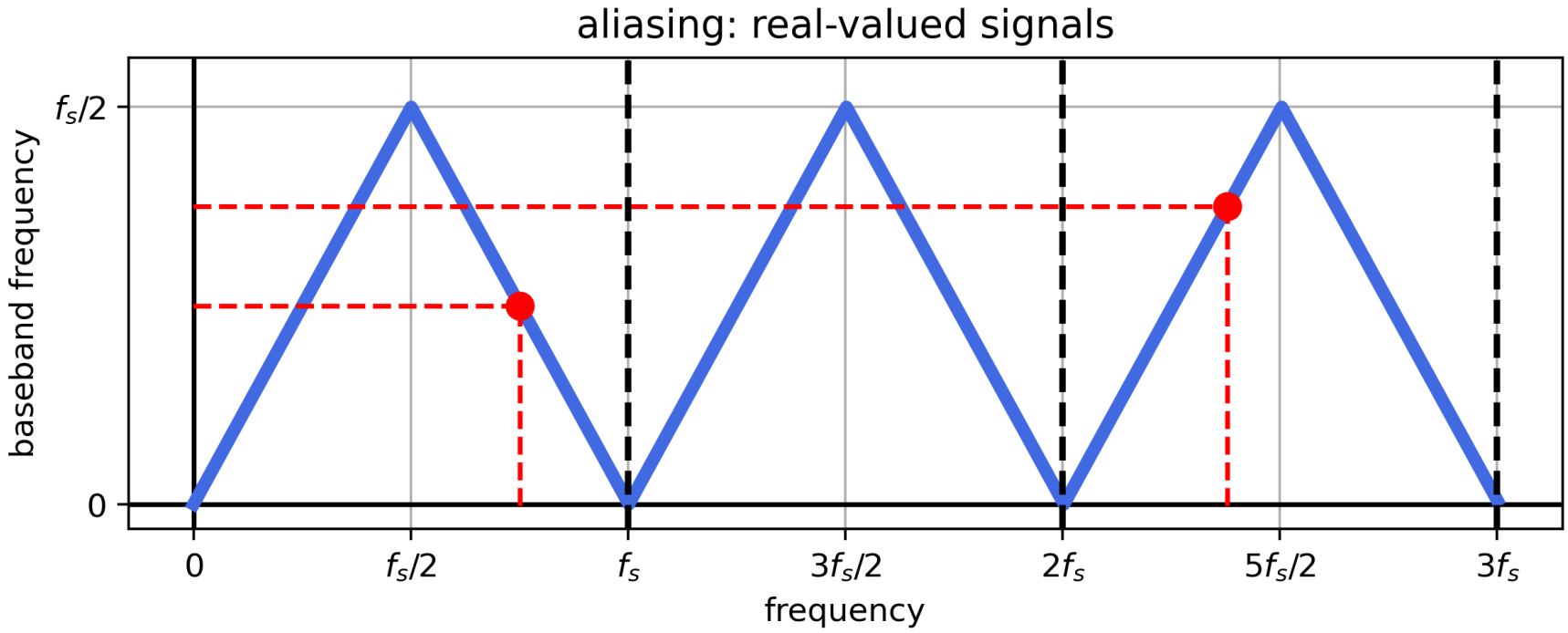
$\Delta = f_s^{-1}$ is the sampling period, sampling interval, time step, or step size between consecutive samples.

Sampling Theorem

Suppose $x(t)$ is a bandlimited signal — a signal such that $X(\omega) = 0$ for some $|\omega| > \omega_c$. The minimum sampling rate ω_s that prevents aliasing is $\omega_s = 2\omega_c$.

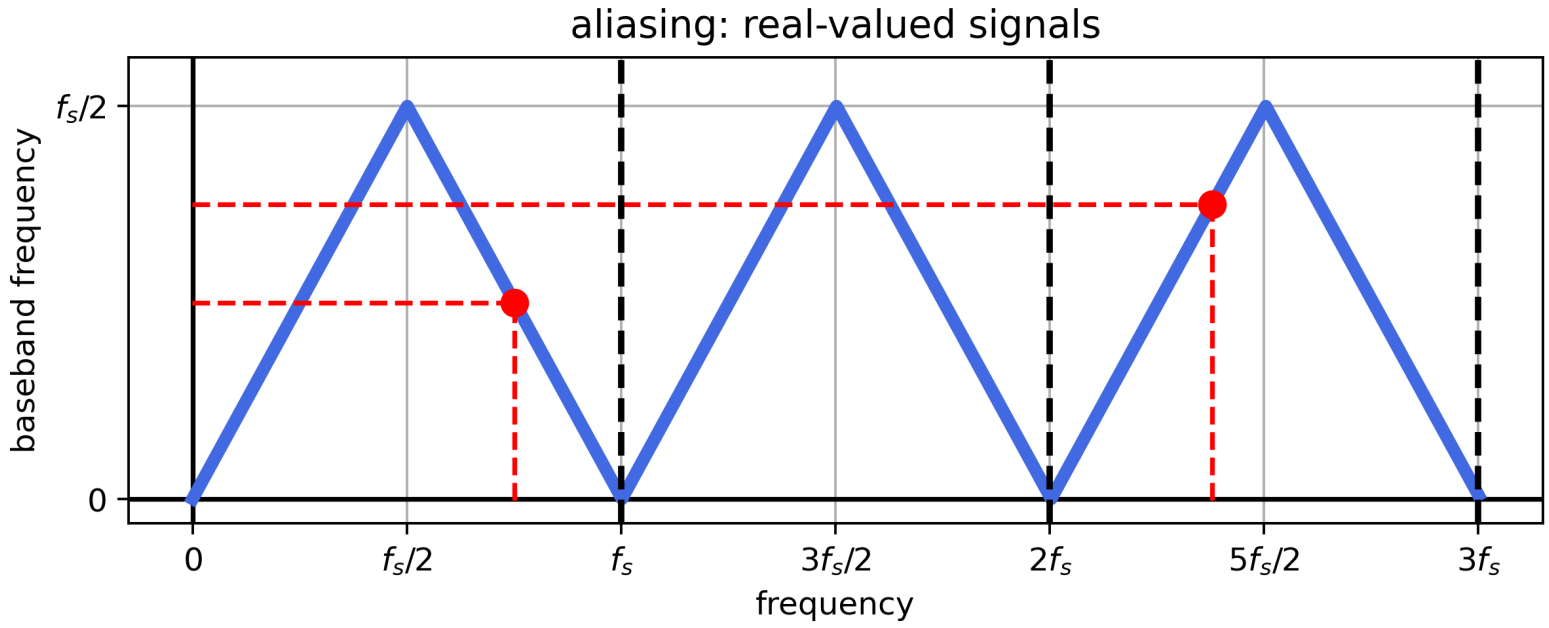
- Sample at twice the highest frequency in the signal.
- If you want to sample frequencies up to $f_c = 22.05$ kHz, you ought to sample at the rate $f_s = 44.1$ kHz.

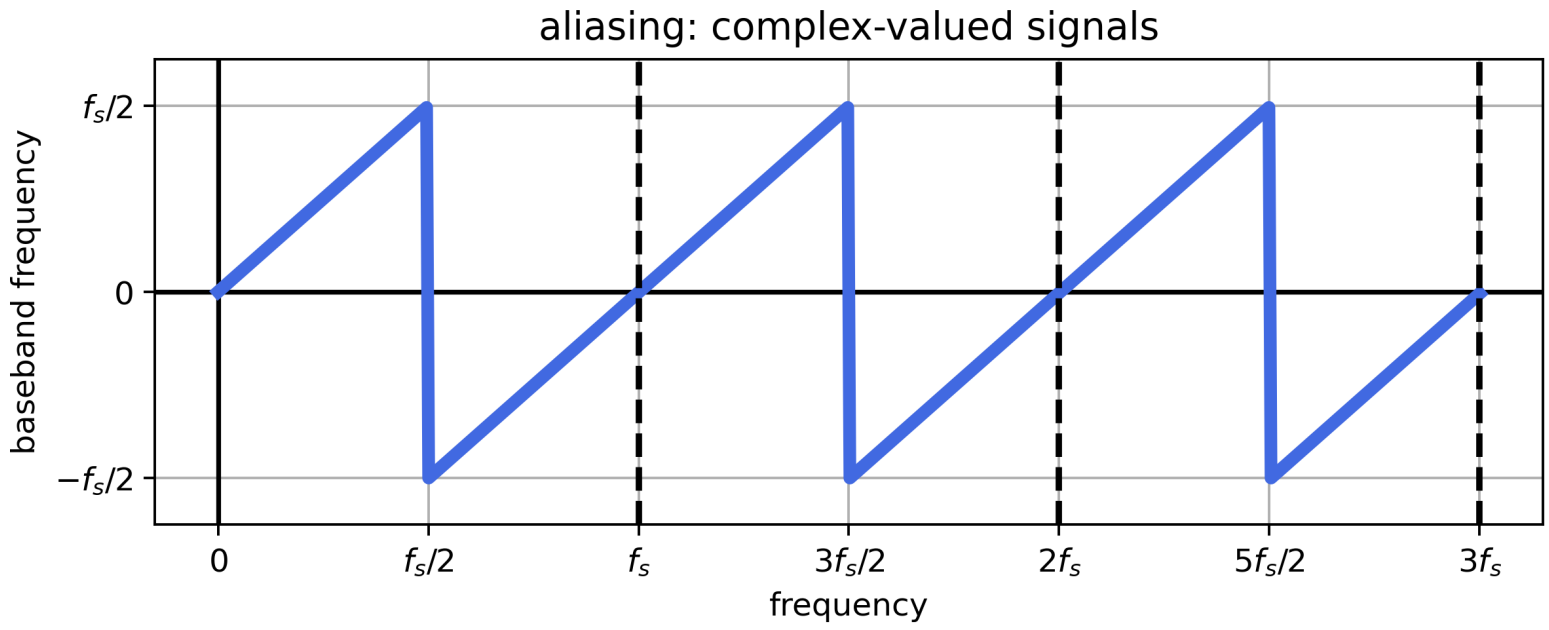
Sampling and Aliasing



Frequencies outside the baseband $[0, \frac{1}{2}f_s]$ alias to frequencies within the baseband.

- $\frac{1}{2}f_s$ is the Nyquist frequency or "folding frequency."
- $\frac{1}{2}f_s$ (samples per second) $\Longleftrightarrow$ π (radians per sample)


aliasing: real-valued signals
baseband frequency
$f_s/2$
0
0
$f_s/2$
f_s
$3f_s/2$
$2f_s$
$5f_s/2$
$3f_s$
frequency


aliasing: complex-valued signals
baseband frequency
$f_s/2$
0
$-f_s/2$
0
$f_s/2$
f_s
$3f_s/2$
$2f_s$
$5f_s/2$
$3f_s$
frequency

Sampling and Aliasing

Example: Sampling and Reconstruction

Suppose that $x(t)$ is a bandlimited CT signal such that $X(\omega) = 0$ for $|\omega| > \omega_c$ where $\omega_c > 0$. Now, sample $x(t)$ at the rate ω_s to produce the DT signal $x[n]$. Next, apply a reconstruction filter to $x[n]$ to produce CT signal $\hat{x}(t)$.

$$x(t) \rightarrow \boxed{\boxed{\textbf{CT/DT}} \rightarrow x[n] \rightarrow \boxed{\textbf{DT/CT}}} \rightarrow \hat{x}(t)$$

Consider $x(t)$ to be the input and $\hat{x}(t)$ to be the output. Is this an LTI system? (Does your verdict change depending on the parameters ω_c and ω_s?)

Sampling and Aliasing

Analog Devices has a visualization of harmonic aliasing on their website: `https://www.analog.com/en/resources/interactive-design-tools/frequency-folding-tool.html`.

Example: Harmonic Aliasing

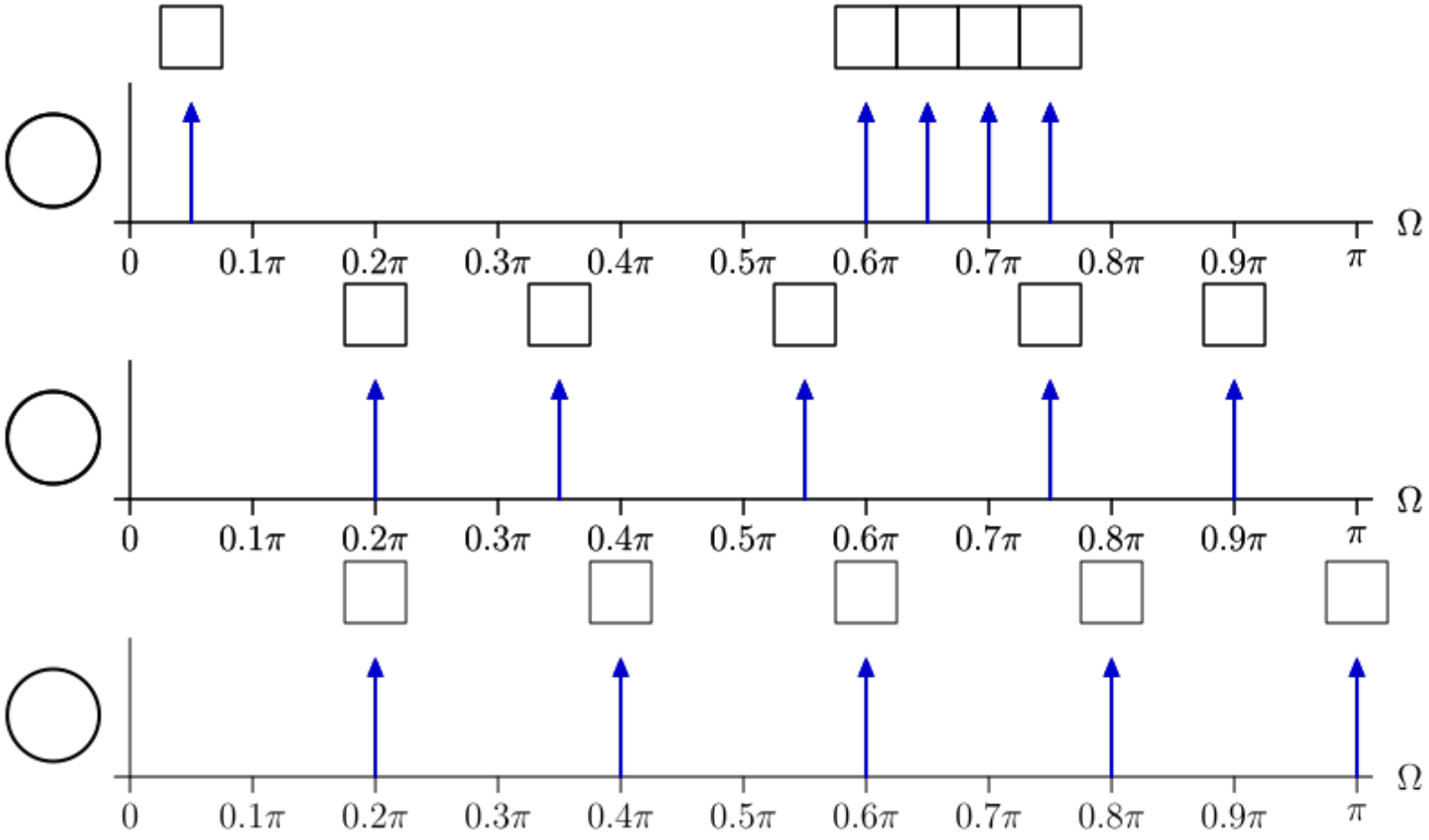
- $x_1(t)$ is periodic with period $T_1 = 1/11$ seconds
- $x_2(t)$ is periodic with period $T_2 = 1/12$ seconds
- $x_3(t)$ is periodic with period $T_3 = 1/13$ seconds

The fundamental frequency of signal $x_i(t)$ is $1/T_i$. Each signal has non-zero harmonics at k/T_i for $k \in \{2, 3, 4, 5\}$.

Sample each continuous-time signal: $x_i[n] = x_i(n/40)$.

Match each $X_i(\Omega)$ to a plot on the next page. In addition, match each impulse to the corresponding CT harmonic.

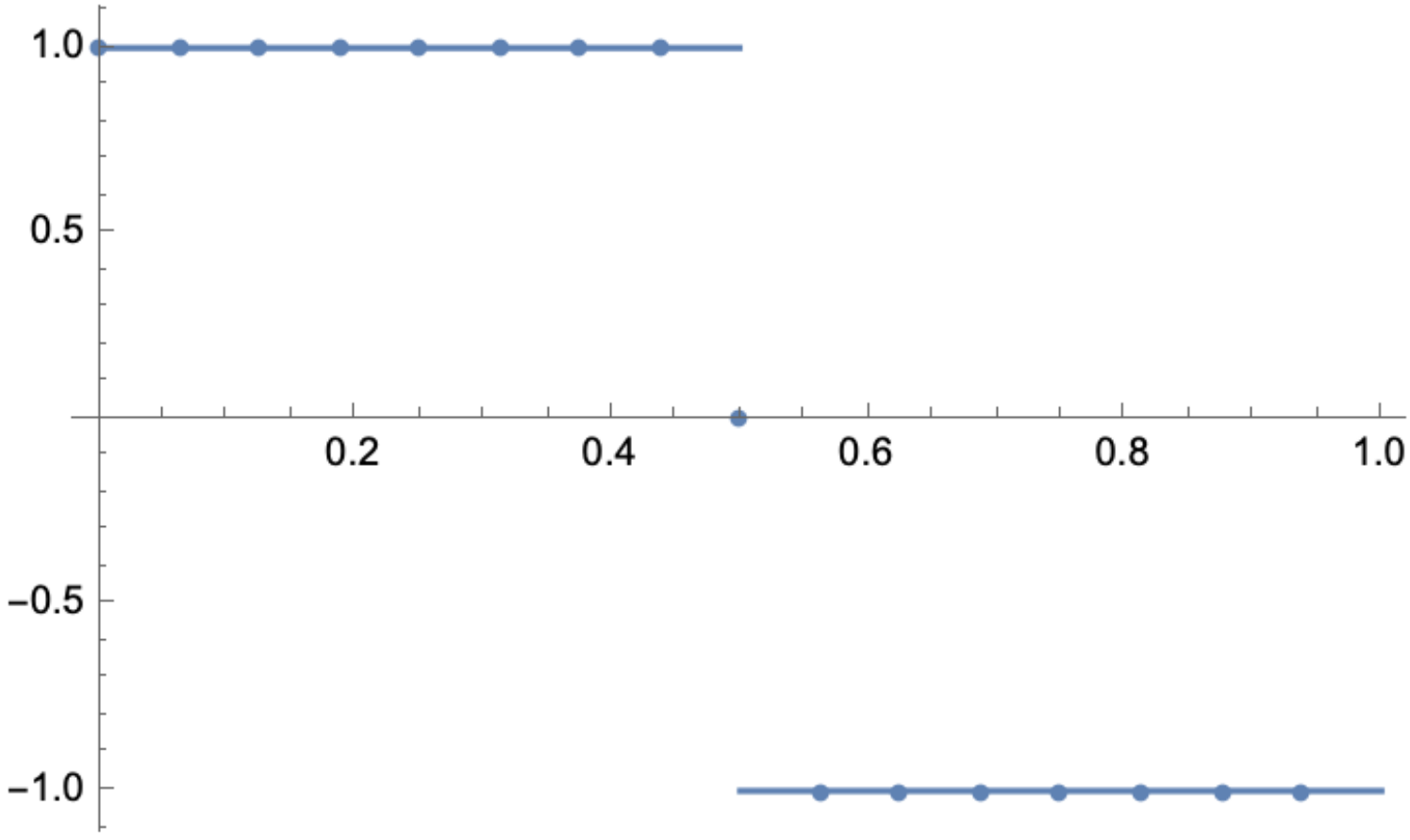
Harmonic Aliasing



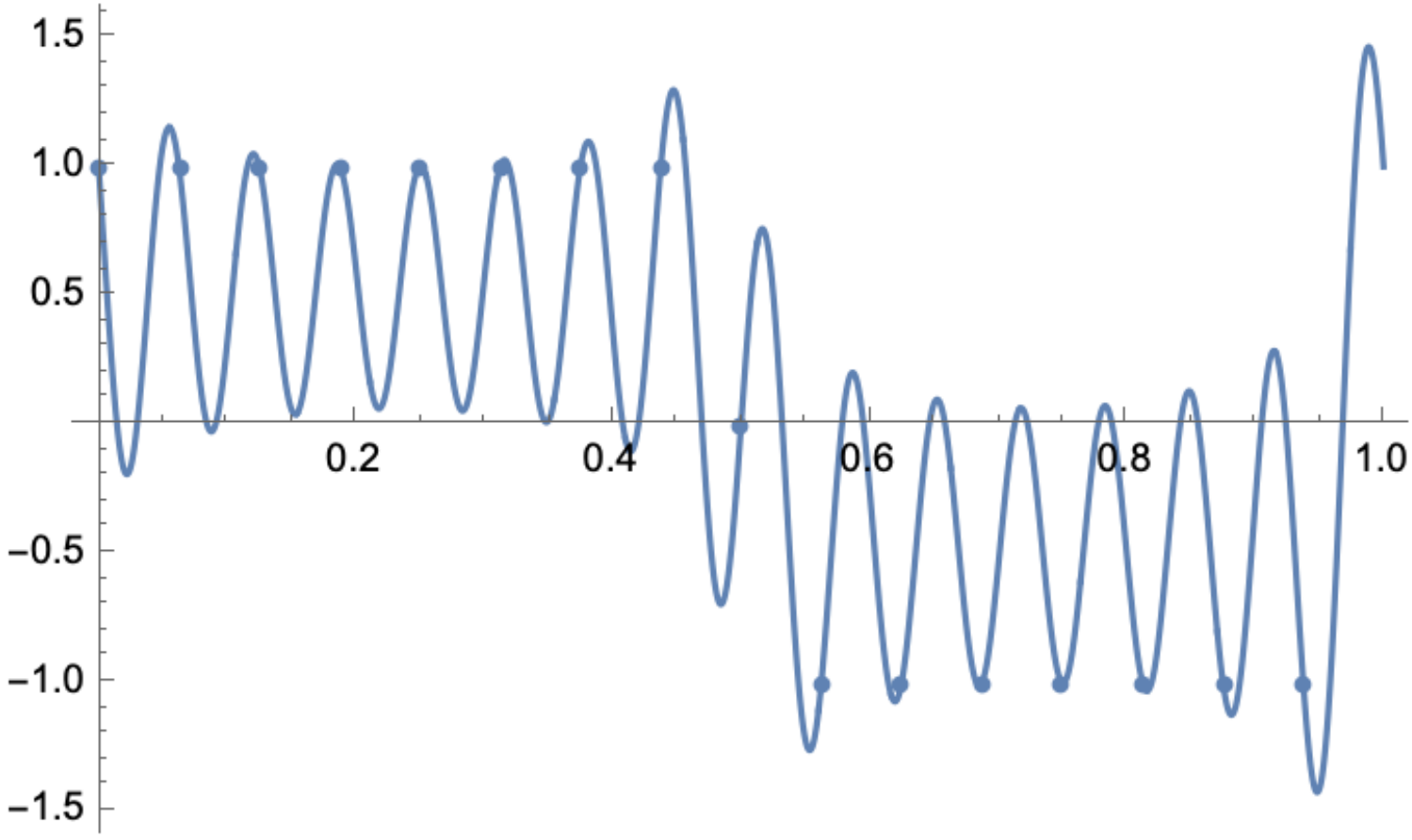
(See the solutions for Problem Set #11.)

Sampling and Aliasing

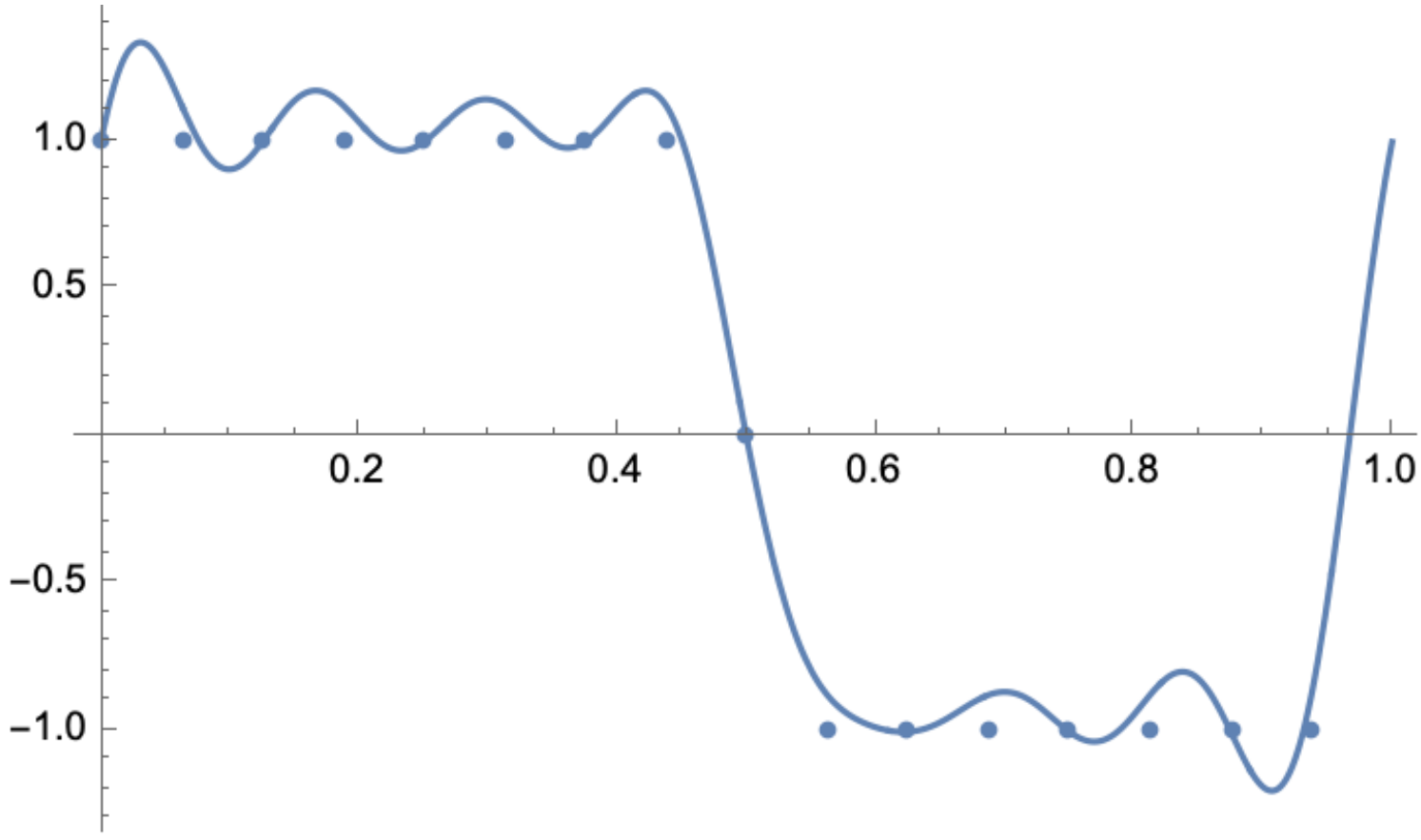
This next problem comes verbatim from an e-mail I received just last weekend. You'll need to act as a "signal processing consultant" for this one.



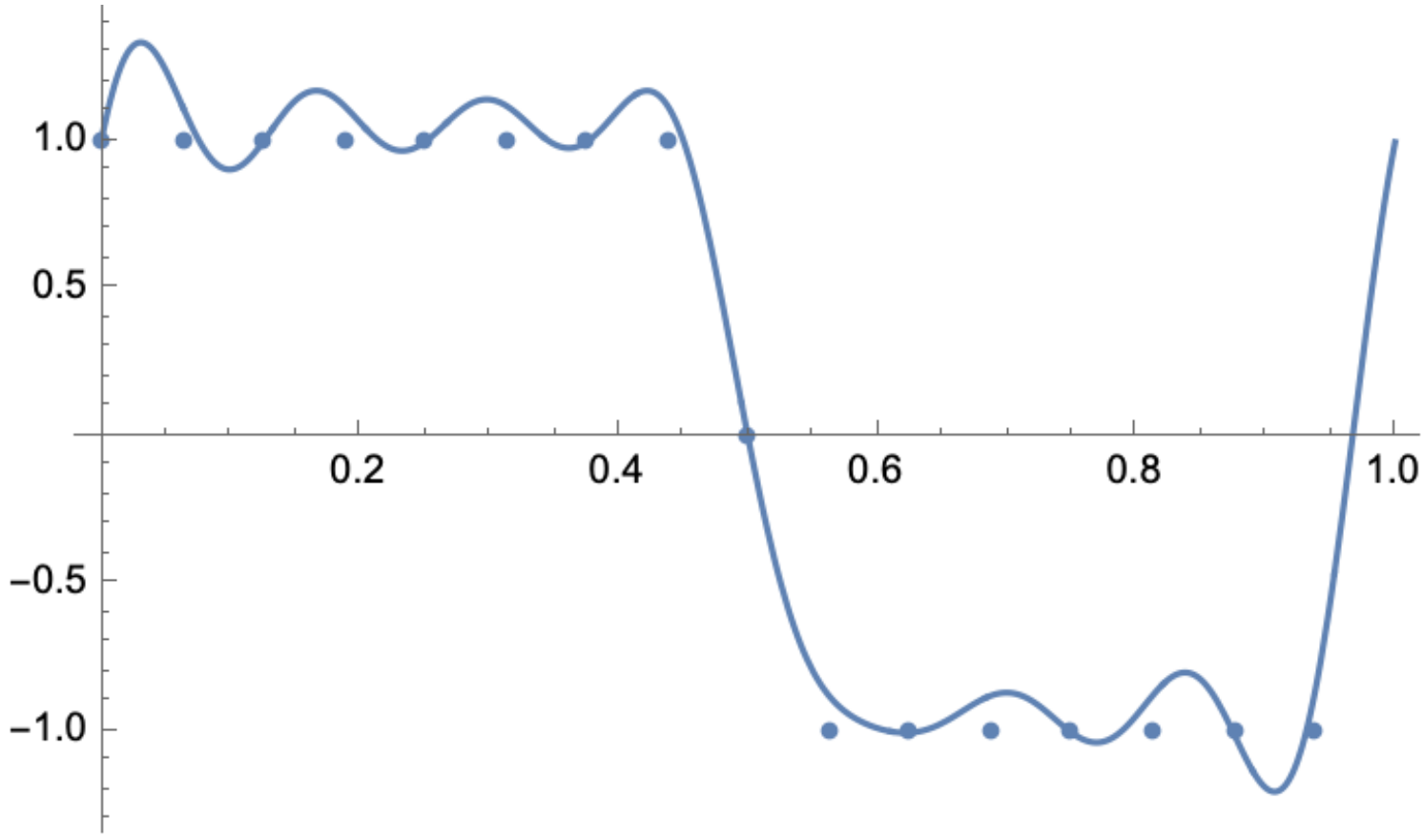
"Imagine I have a signal, like a step function . . . . I sample . . . . I can apply a discrete Fourier transform . . . ."



"When I go back and use the [synthesis] formula . . . I get an approximation to my signal that is wildly oscillating (but still passes through my points)."



"I suspect this has to do with the [sampling] theorem. So to resolve that, I heard that you use the lower half of your frequencies, and you double the resulting amplitudes ... It gave me something closer to what I am expecting."



"Notice how it's not as oscillatory, which seems like something we'd want, but it doesn't quite line up with the sampled points, which seems undesirable . . . I suspect the formula involving the lowest frequencies is incorrect . . . ."

Sampling and Aliasing

What went wrong? (There are multiple issues.)

Sampling and Synthesizing a Square Wave

- We sampled a square wave.
- We tried to re-synthesize from Fourier coefficients.
- We didn't get a square wave, though.

Next, comment on the proposed modification to the synthesis formula: "I heard that you use the lower half of your frequencies, and you double the resulting amplitudes . . . ."

- What does this accomplish?

DFT

Discrete Fourier Transform (DFT)

$$X[k] = \frac{1}{N} \sum_{n=0}^{N-1} x_w[n] e^{-jk\frac{2\pi}{N}n} \qquad x_w[n] = \sum_{k=0}^{N-1} X[k] e^{jk\frac{2\pi}{N}n}$$

where $x_w[n] = x[n]w[n]$ for some window function $w[n]$

Two equivalent perspectives that build on our prior work:

- DTFS of N-periodic extension $\quad X[k] = \text{DTFS}\{x_w[n]\}$
- Sample and scale the DTFT $\quad X[k] = \frac{1}{N} X_w(2\pi k/N)$

DCT

Discrete Cosine Transform (DCT)

$$X_C[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] \cos\left(\frac{\pi k}{N} \left(n + \tfrac{1}{2} \right) \right)$$

$$x[n] = X_C[0] + 2 \sum_{k=1}^{N-1} X_C[k] \cos\left(\frac{\pi k}{N} \left(n + \tfrac{1}{2} \right) \right)$$

The DCT basis functions differ from those of the DFT.

$$\phi_k[n] = \cos\left(\frac{\pi k}{N} \left(n + \tfrac{1}{2} \right) \right)$$

Pattern Matching

This may as well be the most powerful "trick" in this class.

We're often asked to determine the DTFS/DFT/DCT/etc. coefficients for a signal $x[n]$. The synthesis formulæ tell us that $x[n]$ is a sum of DTFS/DFT/DCT/etc. basis functions.

$$x[n] = \sum_{k=0}^{N-1} X[k]\phi_k[n] \qquad \textbf{DTFS/DFT: } \phi_k[n] = e^{jk\frac{2\pi}{N}n}$$

Pattern Matching

If we see that $x[n] = c_0\phi_0[n] + c_1\phi_1[n] + c_2\phi_2[n] + \cdots$ then we can read off the coefficients $X[k]$ without applying the analysis formula! No calculations required!

Pattern Matching

Example: Discrete Fourier Transform Matching

Determine the $N = 2$ DFT of $x[n] = 2 + 3\,(-1)^n$.

Try to determine the DFT without computing a sum.

Hint: Look at the title of this slide. Match the expression for $x[n]$ to the form of the DFT synthesis formula.

For reference, the DFT synthesis formula is given below.

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

DFT basis functions: $\phi_k[n] = e^{jk\frac{2\pi}{N}n}$

Pattern Matching

Example: Discrete Cosine Transform Matching

Determine the $N = 2$ DCT of $x[n] = 5 + 7\cos\left(\frac{\pi}{2}n + \frac{\pi}{4}\right)$.

Try to determine the DCT without computing a sum.

Hint: Look at the title of this slide. Match the expression for $x[n]$ to the form of the DCT synthesis formula.

For reference, the DCT synthesis formula is given below.

$$x[n] = X_C[0] + 2\sum_{k=1}^{N-1} X_C[k]\cos\left(\frac{\pi k}{N}\left(n + \tfrac{1}{2}\right)\right)$$

$$\text{DCT basis functions: } \phi_k[n] = \cos\left(\frac{\pi k}{N}\left(n + \tfrac{1}{2}\right)\right)$$

Pattern Matching

This "trick" also works for the CTFT and DTFT.

$$\textbf{CTFT: } x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{j\omega t}d\omega$$

$$\textbf{DTFT: } x[n] = \frac{1}{2\pi}\int_{2\pi} X(\Omega)e^{j\Omega n}d\Omega$$

Example: Fourier Transform Matching

Determine the DTFT of $x[n] = e^{j\Omega_0 n}$ without a sum.

$$x[n] = \frac{1}{2\pi}\int_{2\pi} \underbrace{2\pi\delta(\Omega-\Omega_0)}_{X(\Omega)} e^{j\Omega n}d\Omega = e^{j\Omega_0 n}$$

A Note on "Tricks"

Applying "tricks" does not mean that we are forgoing the math. Rather, we've done the math so many times and gotten the same result over and over again. After a while, we should be able to expect some results without detailed calculations.

"Tricks" pop up in every discipline:

- common proof strategies (e.g., proof by induction)
- common arguments in physics (e.g., energy and symmetry)
- common circuit topologies (e.g., voltage divider)
- common Fourier transform pairs and properties

To be an expert in some discipline, you do need to know a few things by heart — or at least have an idea of what to expect.

2D Fourier Transforms

The jump from 1D to 2D is like the jump from 18.01 to 18.02.

$$\int f(x)\,dx \rightarrow \iint f(x,y)\,dx\,dy$$

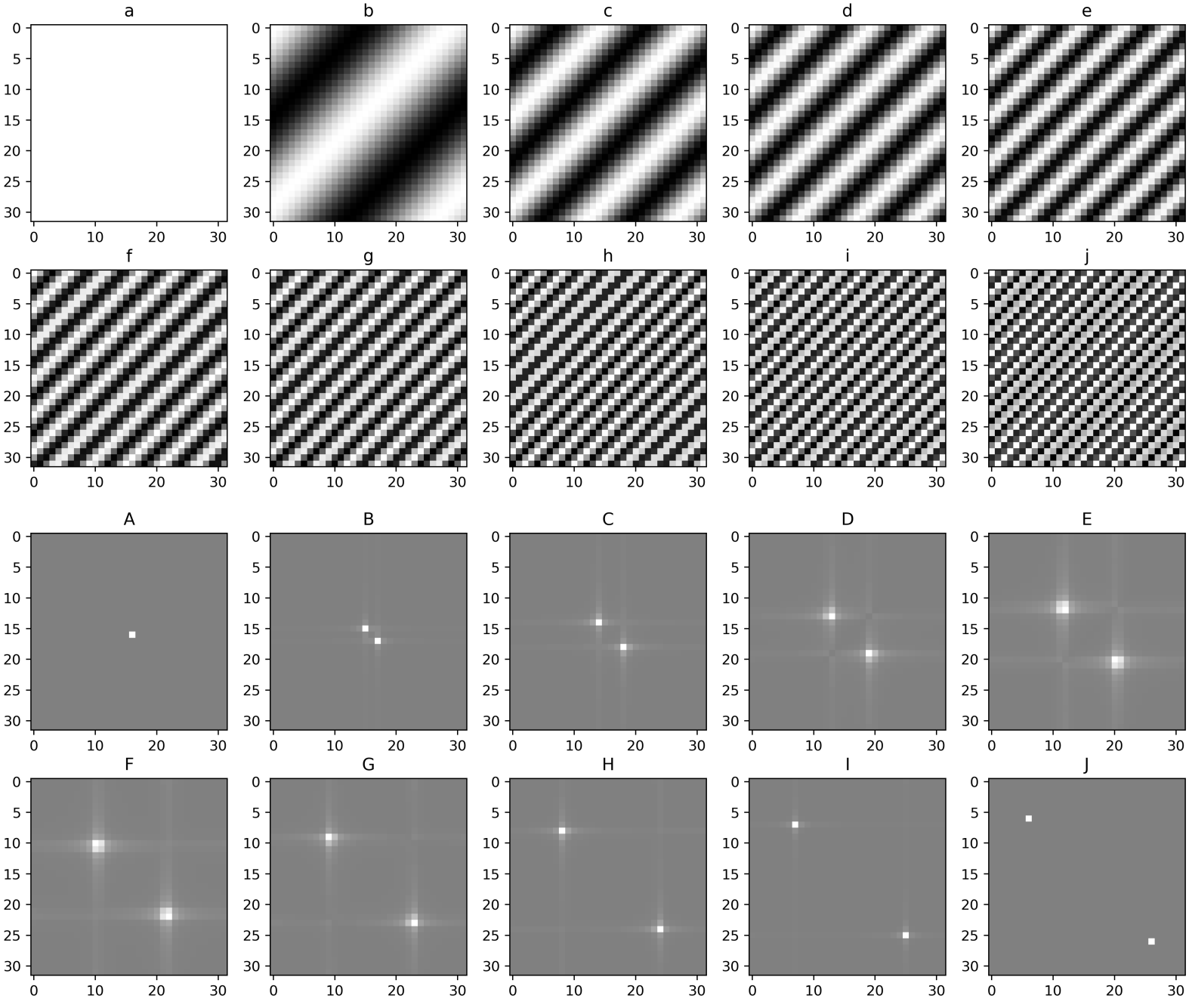
You can't do 2D problems if you can't do 1D problems.

The DFT basis functions are complex exponentials in 2D, too.

$$\phi_{k_r,k_c}[r,c] = \phi_{k_r}[r]\phi_{k_c}[c] = e^{j(k_r\frac{2\pi}{R}r + k_c\frac{2\pi}{C}c)} = e^{jk_r\frac{2\pi}{R}r}e^{jk_c\frac{2\pi}{C}c}$$

The 2D case also brings some new trends and properties.

- separability: rows-then-columns, or columns-then-rows
- transform of a horizontal line is a vertical line
- transform of a vertical line is a horizontal line
- rotating a signal rotates its 2D Fourier transform


a
b
c
d
e
f
g
h
i
j
A
B
C
D
E
F
G
H
I
J

2D Fourier Transforms

Many ideas introduced in the 1D case directly extend to 2D. For example, a time (or space) shift adds a phase delay.

$$h[n] = \delta[n - n_0] \iff H(\Omega) = e^{-j\Omega n_0}$$

$$|H(\Omega)| = 1 \text{ and } \angle H(\Omega) = -n_0\Omega$$

$$h[r, c] = \delta[r - r_0, c - c_0] \iff H(\Omega_r, \Omega_c) = e^{-j(\Omega_r r_0 + \Omega_c c_0)}$$

$$|H(\Omega_r, \Omega_c)| = 1 \text{ and } \angle H(\Omega_r, \Omega_c) = -r_0\Omega_r + -c_0\Omega_c$$

Shifting by (r_0, c_0) adds a phase delay in that same direction.

$$g[r, c] = (f * h)[r, c] = f[r - r_0, c - c_0]$$

$$G(\Omega_r, \Omega_c) = F(\Omega_r, \Omega_c) e^{-j(\Omega_r r_0 + \Omega_c c_0)}$$

2D Fourier Transforms

Separability

Factor $f[r, c]$ into the product of a row-dependent signal $f_r[r]$ and a column-dependent signal $f_c[c]$.

$$f[r, c] = f_r[r] f_c[c] \iff F(\Omega_r, \Omega_c) = F_r(\Omega_r) F_c(\Omega_c)$$

Example: 2D Box

Compute the 2D Fourier transform of $f[r, c]$.

$$f[r, c] = \begin{cases} 1 & 0 \leq r \leq R-1 \text{ and } 0 \leq c \leq C-1 \\ 0 & \text{otherwise} \end{cases}$$

2D Fourier Transforms

Factor $f[r, c]$ into the product of a row-dependent signal $f_r[r]$ and a column-dependent signal $f_c[c]$. Compute the transforms of $f_r[r]$ and $f_c[c]$ separately.

$$f_r[r] = \begin{cases} 1 & 0 \leq r \leq R-1 \\ 0 & \text{otherwise} \end{cases}$$

$$f_c[c] = \begin{cases} 1 & 0 \leq c \leq C-1 \\ 0 & \text{otherwise} \end{cases}$$

$$F_r(\Omega_r) = \left(\frac{1 - e^{-j\Omega_r R}}{1 - e^{-j\Omega_r}} \right)$$

$$F_c(\Omega_c) = \left(\frac{1 - e^{-j\Omega_c C}}{1 - e^{-j\Omega_c}} \right)$$

$$F(\Omega_r, \Omega_c) = F_r(\Omega_r) F_c(\Omega_c) = \left(\frac{1 - e^{-j\Omega_r R}}{1 - e^{-j\Omega_r}} \right) \left(\frac{1 - e^{-j\Omega_c C}}{1 - e^{-j\Omega_c}} \right)$$

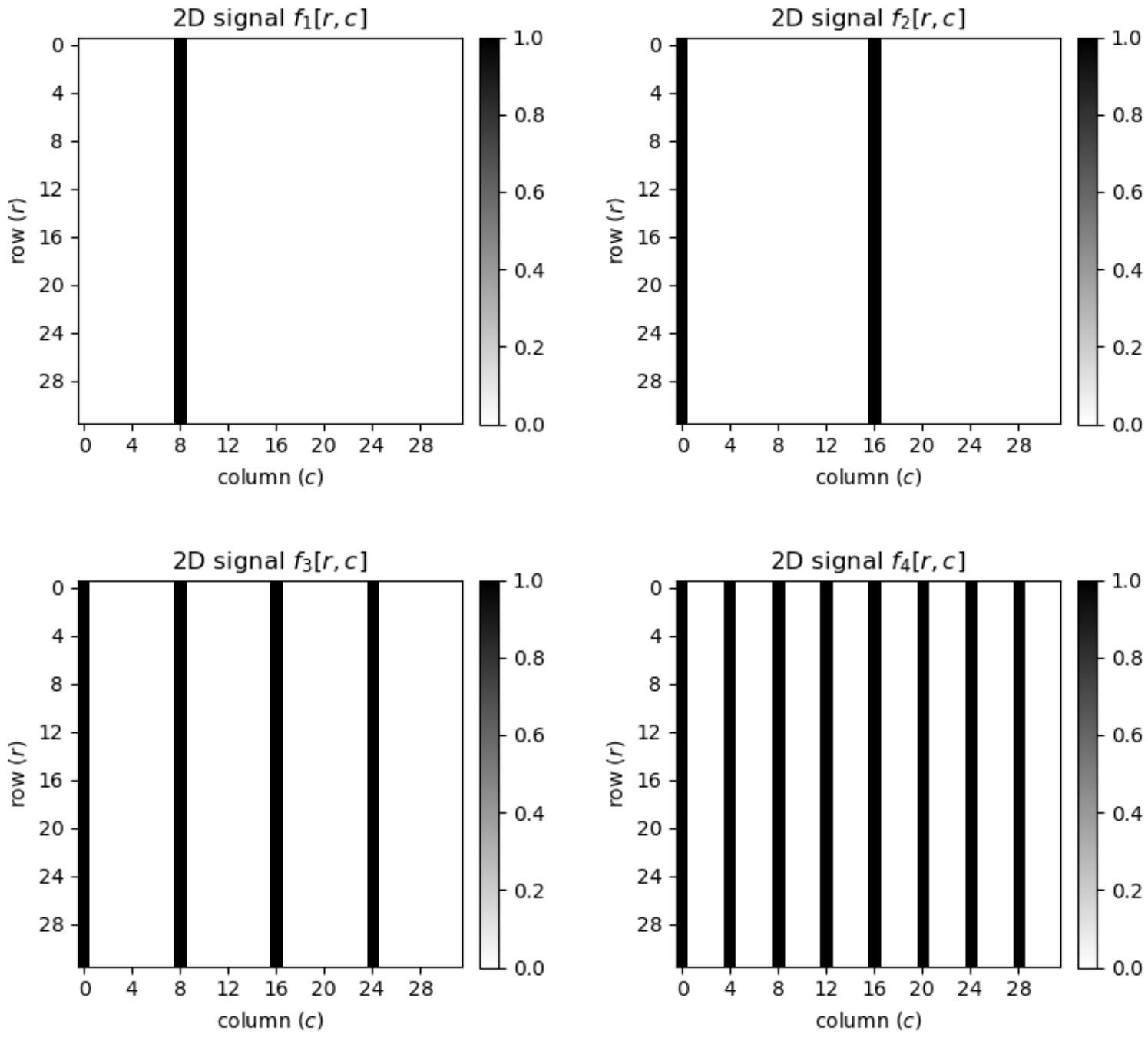
2D Fourier Transforms

On next slide, four 2D signals are shown.

- $f_1[r, c]$
- $f_2[r, c]$
- $f_3[r, c]$
- $f_4[r, c]$

For each 2D signal, compute the 2D DFT, where $R = C = 32$.

(Making insightful 2D DFT matching problems takes a lot of time and creativity. I recommend looking over homework problems and the review questions from recitation on May 8. Those problems should cover all the bases. I'm happy to work through these types of problems in the second review session.)


2D signal $f_1[r, c]$
2D signal $f_2[r, c]$
2D signal $f_3[r, c]$
2D signal $f_4[r, c]$
row (r)
column (c)
0
4
8
12
16
20
24
28
0.0
0.2
0.4
0.6
0.8
1.0

Why Fourier?

If you're thinking, "I took 6.300 and all I got was a bunch of lousy Fourier transforms," these slides are especially for you.

By now, you all ought to be able to answer, "Why Fourier?"

Why not focus exclusively on any of the following transforms?

- discrete cosine transform (DCT)
- Hadamard transform
- Hankel transform
- Hilbert transform
- Legendre transform
- Radon transform
- Wavelet transform
- Weierstrass transform

e.g., `https://en.wikipedia.org/wiki/List_of_transforms`

Why Fourier?

Eigenfunctions

Fourier basis functions are complex exponentials.
Complex exponentials are eigenfunctions of LTI systems.

We can readily analyze and design LTI systems.

- differential or difference equation
- impulse or unit-sample response
- frequency response

6.3010 emphasizes this perspective.

The eigenbasis is, for many purposes, the right basis.

but not all purposes

Eigenfunctions

$$\boldsymbol{x} = \sum_k c_k \boldsymbol{v}_k \rightarrow \boxed{\boldsymbol{A}} \rightarrow \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} = \sum_k c_k \lambda_k \boldsymbol{v}_k$$

$$\phi_k[n] \rightarrow \boxed{\textbf{linear operator}} \rightarrow \lambda(k) \cdot \phi_k[n]$$

$$\sum_k c_k \phi_k[n] \rightarrow \boxed{\textbf{linear operator}} \rightarrow \sum_k c_k \lambda_k \phi_k[n]$$

$$e^{j\Omega n} \rightarrow \boxed{\textbf{LTI}} \rightarrow H(\Omega) e^{j\Omega n}$$

$$\frac{1}{2\pi} \int_{2\pi} X(\Omega) e^{j\Omega n} d\Omega \rightarrow \boxed{\textbf{LTI}} \rightarrow \frac{1}{2\pi} \int_{2\pi} H(\Omega) X(\Omega) e^{j\Omega n} d\Omega$$

Why Fourier?

Anonymous Gear 4 years ago

I took 6.003 last fall, and personally I wouldn't recommend taking it if you don't have to. It was cool learning about Fourier Transforms and doing programming labs with them, but nothing seemed immediately useful outside of the class and I didn't gain a particularly good intuition for Fourier transforms, just a basic one. Towards the end of 6.003 it was revealed that modern compression techniques use different transforms (like DCT) that weren't well covered in the course, so the heavy emphasis on Fourier Transforms felt wasted.

The discrete cosine transform (DCT) basis functions are not eigenfunctions of LTI systems. The Fourier basis functions are eigenfunctions of LTI systems, however. That's one reason why the Fourier transform is particularly important and worthy of our focus. (Fourier pops up across many fields of study!)

Eigenfunctions

Example: A Tale of Two Eigenfunctions

Suppose $x[n] = 1 + (-1)^n$ and $h[n] = 2^{-n}u[n] + 3^n u[-n]$.

$$x[n] \rightarrow \boxed{\textbf{LTI}} \rightarrow y[n] = (x * h)[n]$$

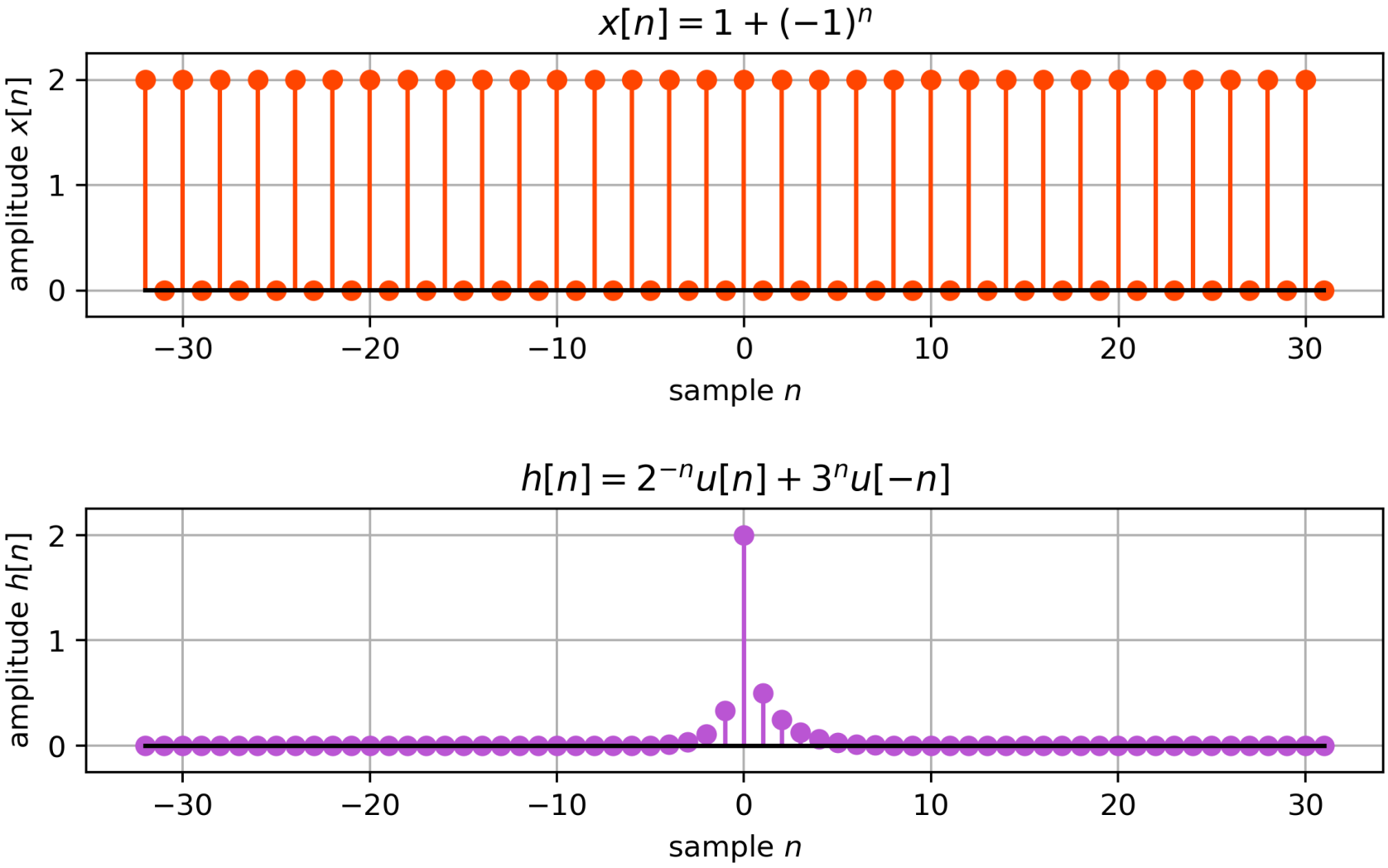
Determine $y[n] = (x * h)[n]$.

Directly computing the convolution sum

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

may be difficult, as $x[n]$ and $h[n]$ are infinitely-long signals.

Eigenfunctions



Follow-On Subjects

(Many are offered this fall; others are offered in spring or every other fall.)

Signal Processing

6.3010	Signals, Systems, and Inference	**U**
6.3020	Fundamentals of Music Processing	**U/G**
6.7000	Discrete-Time Signal Processing	**G**
6.8800	Biomedical Signal and Image Processing	**U/G**

Signals, Systems, and Applications

6.2000	Electrical Circuits: Modeling and Design	**U**
6.2300	Electromagnetics, Waves, and Applications	**U**
6.3100	Dynamical System Modeling and Control	**U/G**
6.4800	Biomedical Imaging with MRI **(CI-M)**	**U**
6.7410	Principles of Digital Communication	**U/G**
6.8620	Spoken Language Processing	**G**
6.C27	Computational Imaging	**U/G**

6.3010: Signals, Systems, and Inference (Spring)

Covers signals, systems and inference in communication, control and signal processing. Topics include input-output and state-space models of linear systems driven by deterministic and random signals; time- and transform-domain representations in discrete and continuous time; and group delay. State feedback and observers. Probabilistic models; stochastic processes, correlation functions, power spectra, spectral factorization. Least-mean square error estimation; Wiener filtering. Hypothesis testing; detection; matched filters.

6.3020: Fundamentals of Music Processing (Fall)

Analyzes recorded music in digital audio form using advanced signal processing and optimization techniques to understand higher-level musical meaning. Covers fundamental tools like windowing, feature extraction, discrete and short-time Fourier transforms, chromagrams, and onset detection. Addresses analysis methods including dynamic time warping, dynamic programming, self-similarity matrices, and matrix factorization. Explores a variety of applications, such as event classification, audio alignment, chord recognition, structural analysis, tempo and beat tracking, content-based audio retrieval, and audio decomposition. Students taking graduate version complete different assignments.

6.4800: Biomedical Imaging with MRI (Fall)

Medical imaging with MRI, motivated by examples of problems in human health that engage students in imaging hardware design, data acquisition and image reconstruction, and signal analysis and inference. Data from scientific and clinical applications in neuro- and cardiac MRI as applied in current practice are sourced for computational labs. Labs include kits for interactive and portable low-cost devices that can be assembled by the students to demonstrate fundamental building blocks of an MRI system. Students program lab MRI systems on their laptops for data collection and image reconstruction. Students apply concepts from lectures in labs for data collection for image reconstruction, image analysis, and inference by their own design, drawing on concepts in signal processing and machine learning.

Summer Projects

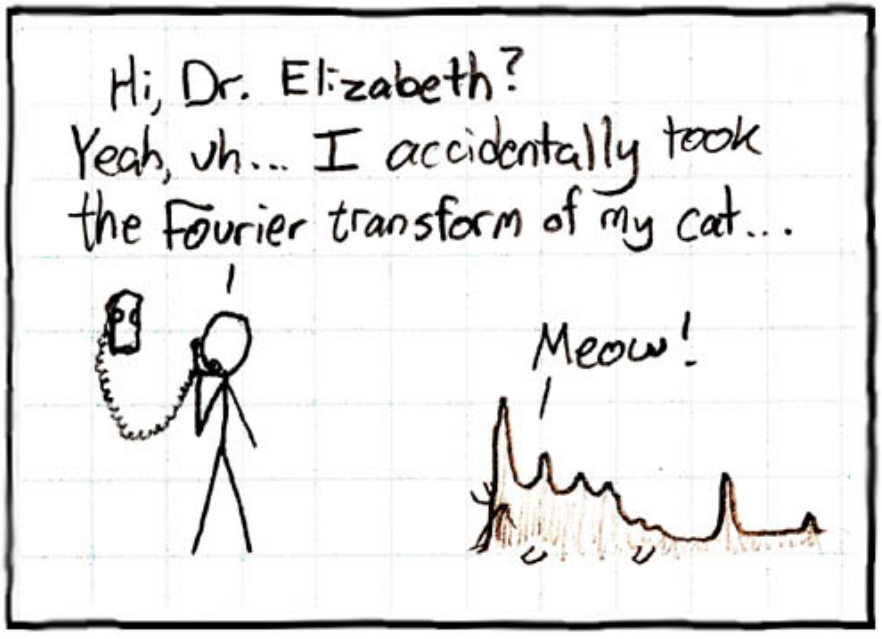
Digital Signal Processing (DSP) with Python

- `numpy` — numerical methods
- `scipy.signal` — signal processing
- `librosa` — music information retrieval
- `lcapy` — linear circuit analysis
- `scikit-rf` — radio-frequency engineering

Software-Defined Radio (SDR)

- GNU Radio

You could apply to be a 6.300 lab assistant (LA) this fall!



xkcd (#26: "Fourier")

Please fill out a subject evaluation for this class at **https://registrar.mit.edu/classes-grades-evaluations/subject-evaluation** — especially if you appreciate the work of the teaching staff!