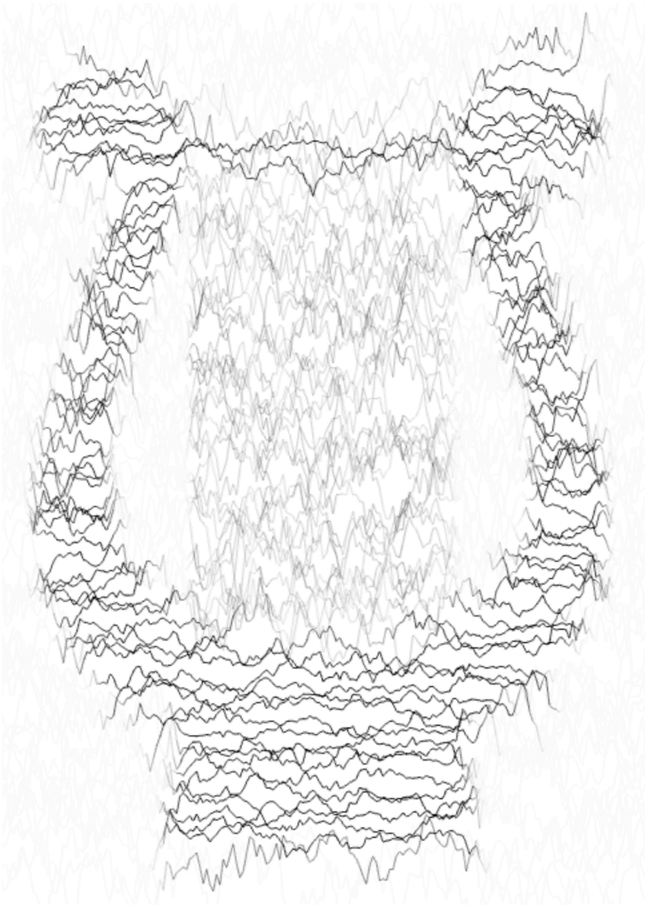
University of Illinois @ Urbana-Champaign



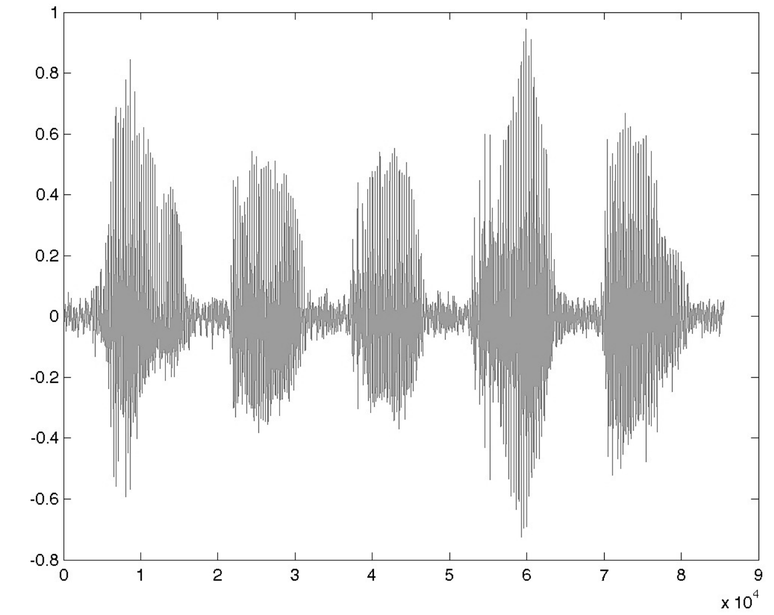
CS 498PS – Audio Computing Lab

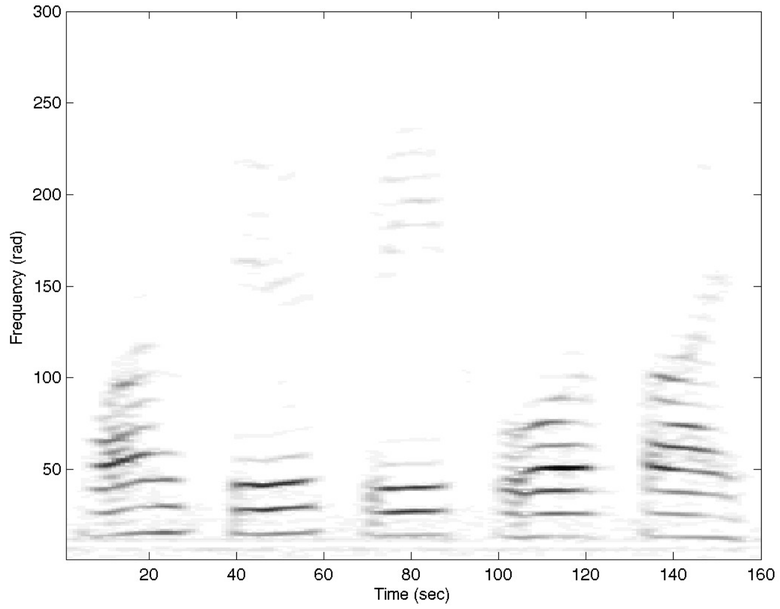
Audio DSP basics

Paris Smaragdis
paris@illinois.edu
paris.cs.illinois.edu

Overview

- Basics of digital audio
- Signal representations
  - Time, Frequency, Time/Frequency
  - Sampling, Quantization
- The Fourier transform
  - DFT and FFT
- The Spectogram



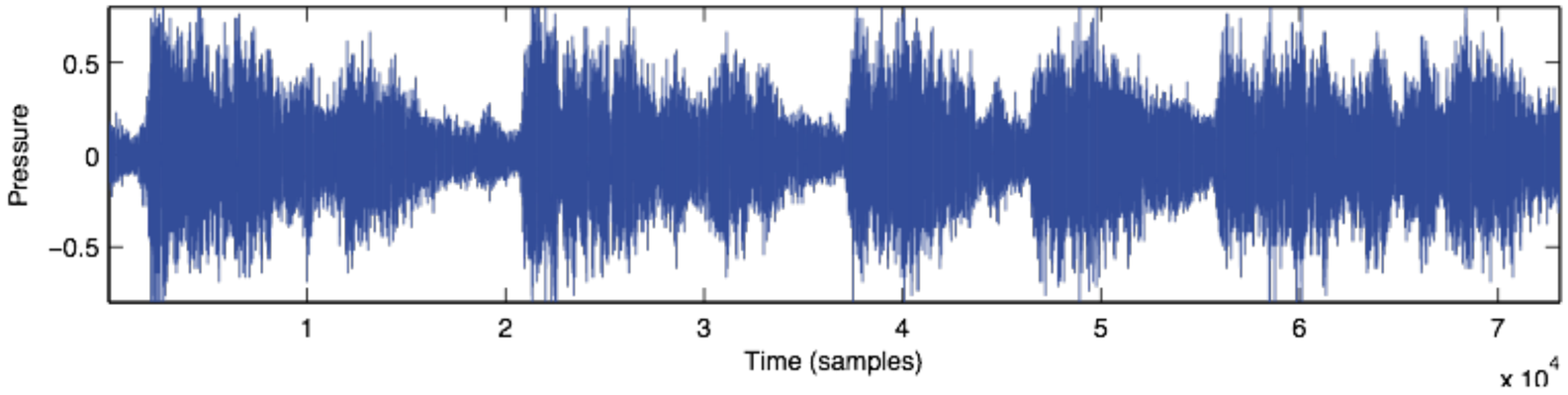


Why digital audio?

- Cheaper
  - Get a smartphone, do anything you want
  - No burning circuits!
- Easier
  - You can easily rewrite code
    - But cannot easily rewire circuits
- Smaller
  - Do everything on one chip

Sound as “numbers”

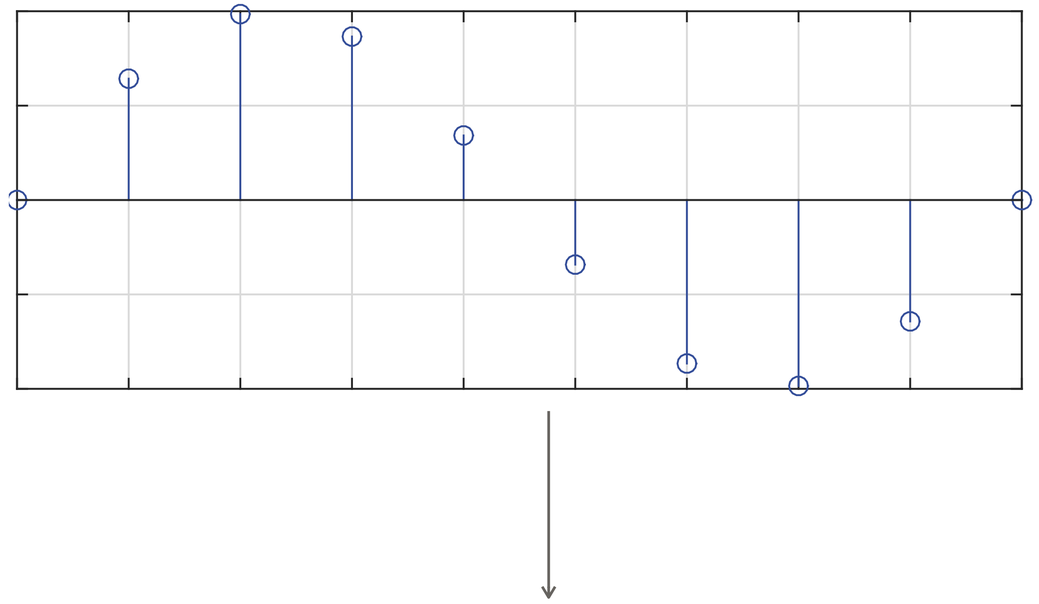
- We treat sound as a series of amplitudes
  - More on the details later



- This is the *waveform* representation
  - Encodes instantaneous pressure over time

PCM format

- "Pulse Code Modulation"
  - Used by CDs, telephones, audio editors, synths, etc.



0, 82, 126, 111, 44, -44, -111, -126, -82, 0

This is a *discrete* and *digital* format

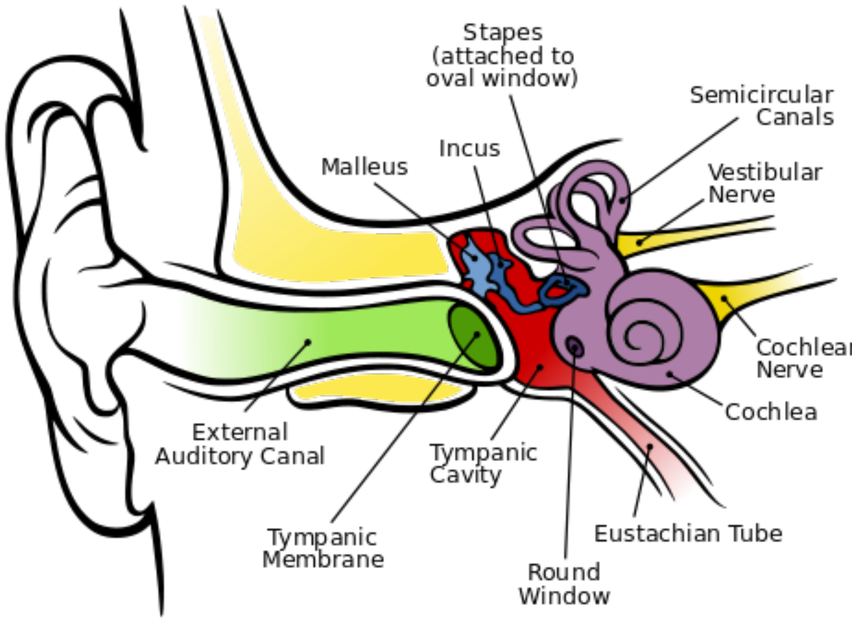
- We do not use continuous values
  - We have finite samples over time
  - We (usually) encode these samples as signed integers
- Common formats
  - Speech: 16kHz / 16-bit (or 8-bit)
  - Music: 44.1kHz / 16-bit (or 95kHz / 24-bit)
- But how do we pick these numbers?
  - What do they mean?

Dynamic range

- The choice of bits defines the dynamic range
  - More bits == more dynamic range == more storage
- What is dynamic range?
  - Ratio of highest and lowest represented pressure value
  - Usually measured in *decibels* (dB)
- How much dynamic range do we need though?

It all hinges on how we hear

- Outer ear
  - Sound gets collected at the *pinna*
  - The *ear canal* amplifies (some) sound by ~10dB
  - The *ear drum* vibrates according to incoming pressure
- Middle ear
  - The *ossicles* transfer sound to the *oval window*
  - Amplify sound by ~14dB
    - Also use muscles for damping
- Inner ear
  - Translation to neural signal (more later)



Perception of sound

- The just noticeable sound is:
  - 10^{-12} W/m^2 (cannot hear softer than this)
- And the as noticeable as it get is:
  - 1 W/m^2 (and then you go deaf!)
- Thus our dynamic range is:
  - $10 \log_{10}(1/10^{-2}) = 120$ dB
  - That's a staggering trillion to one!

To get you oriented

- Weakest detectable sound ~0 dB
- Soft breathing ~10dB
- Quiet library ~40 dB
- Office environment ~60 dB
- Food blender ~80 dB
- Lawn mower ~90 dB

Dangerous levels > 90 dB

- Car horn at 1m ~110 dB

Pain begins at 125 dB

- Military jet at 50ft ~130 dB
- Shotgun blast ~165 dB

Pain ends at 180 dB

- Loudest possible sound 194 dB

(cause your ears just blew up)

  - (after which it isn't "sound" anymore it is a "shock wave")

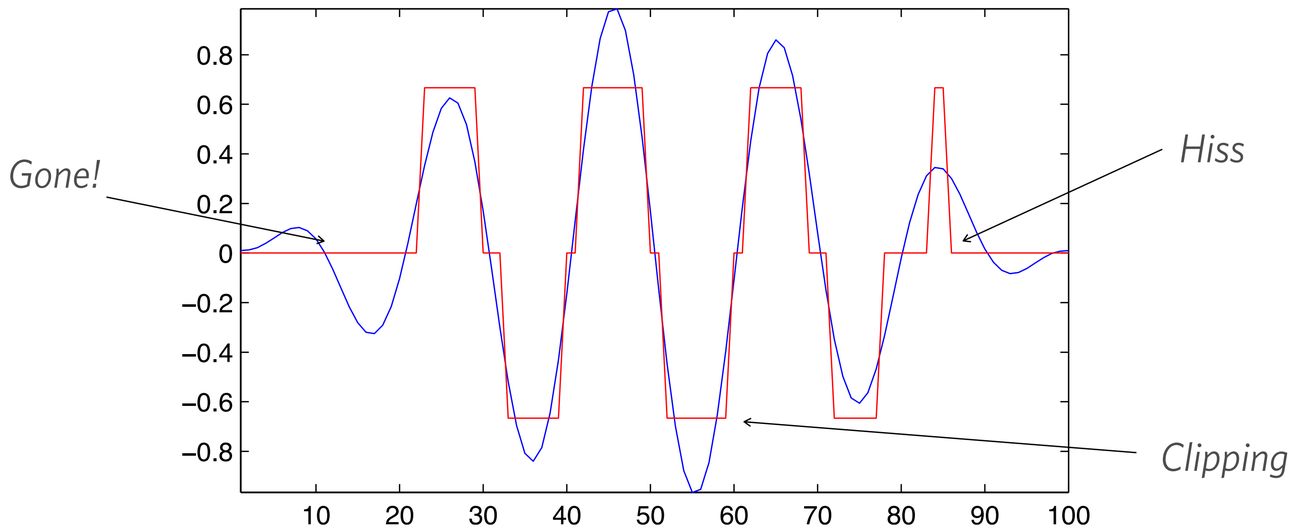
Back to digital sound

- How many dB dynamic range to use?
  - Close to 120 dB ideally

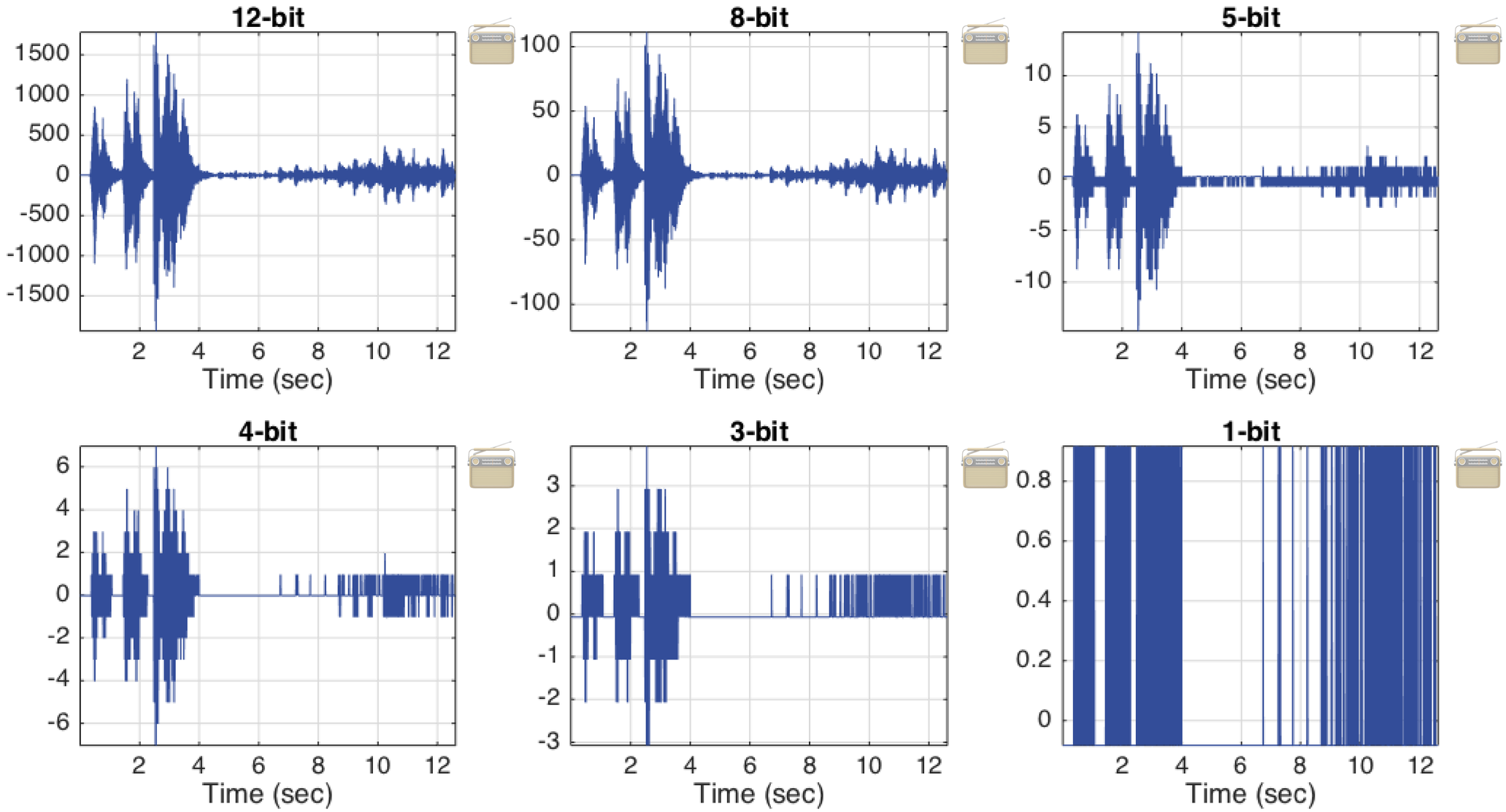
- Common ranges (*headroom*)
  - 16-bit / 96 dB (the industry standard)
  - 12-bit / 72 dB (the cheap standard)
  - 8-bit / 48 db (the 80's standard! hipsters?)
  - 24-bit / 144 dB (the "I'm charging you extra" standard)
  - Floating point (what we will use)

Why worry?

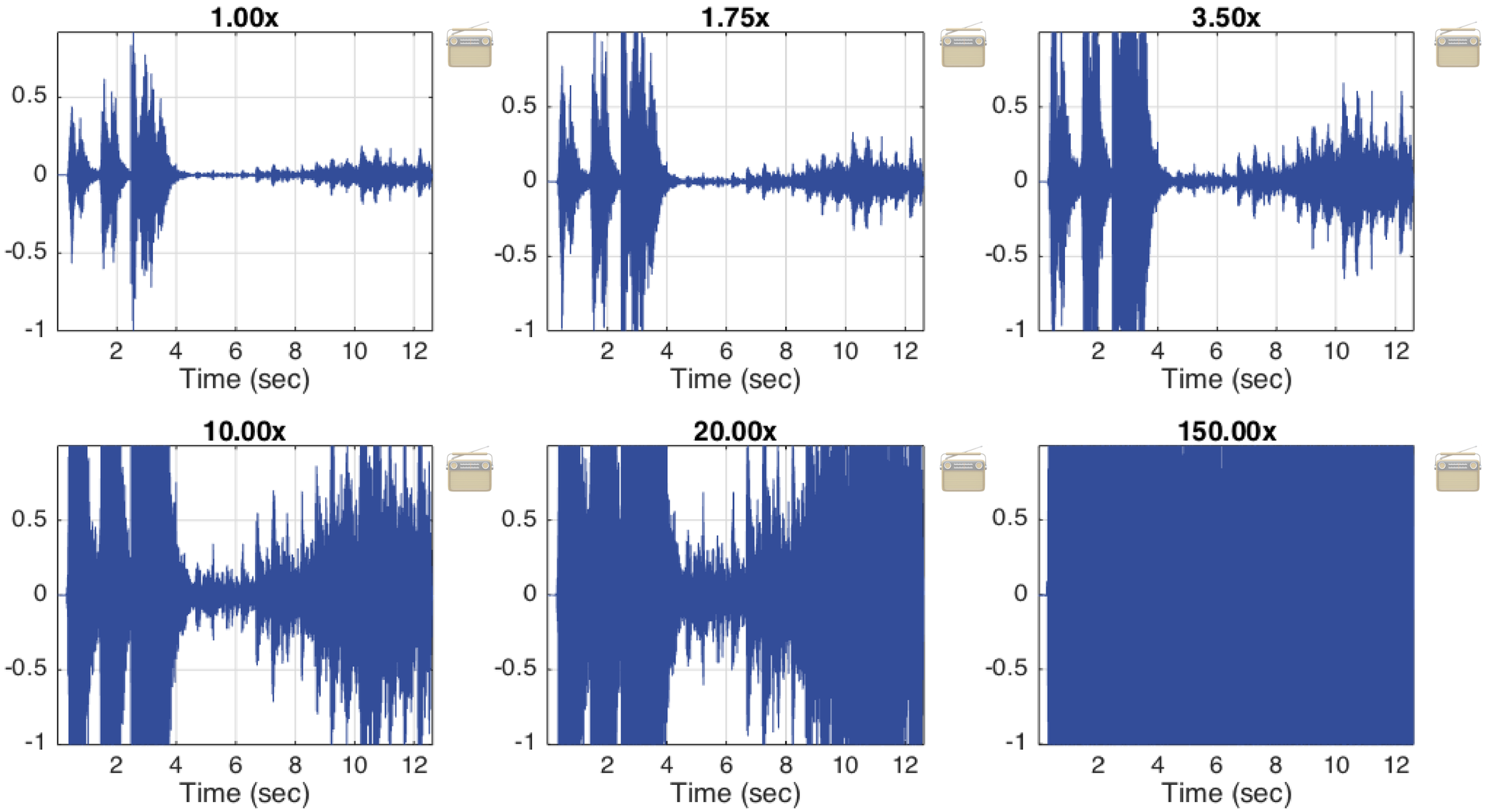
- Need headroom to avoid *clipping* & *quantization noise*
  - These happen when the representation is maxed or zero
    - Very challenging with dynamic content (e.g. classical music)
    - An audio engineer's nightmare! (and digital is worse)



Quantization noise examples



Clipping examples

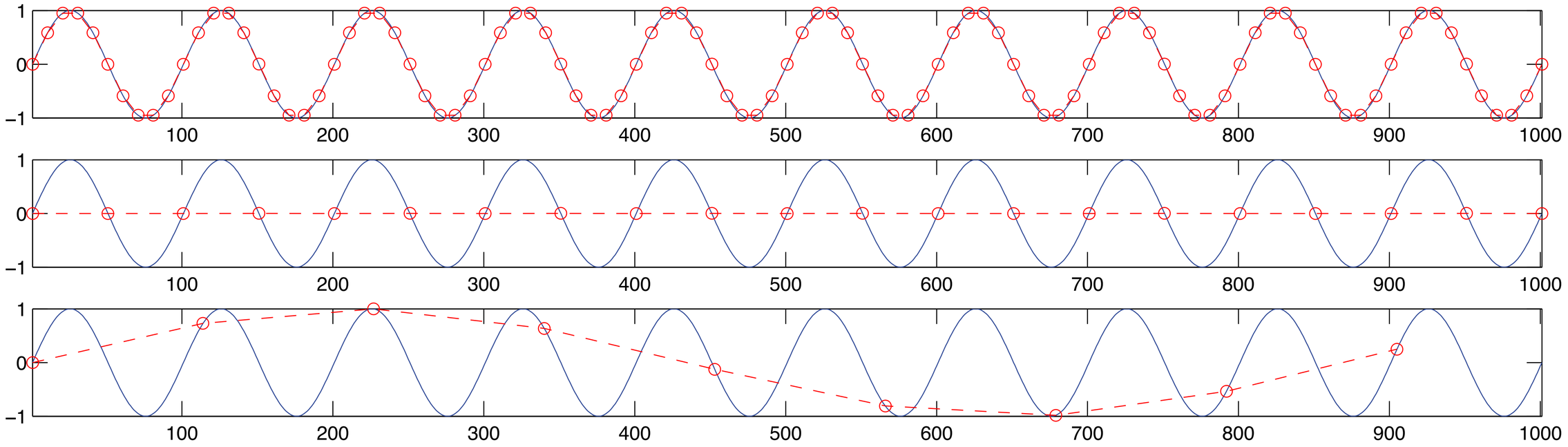


Sampling in time

- Also known as A/D conversion
  - How to we convert real-world sound to a discrete sequence?
- The one parameter we care for: the *sample rate*
  - i.e. how often do we represent the input sound
- Tradeoffs
  - Sample fast and you waste memory and energy
  - Sample slow and you risk *aliasing*

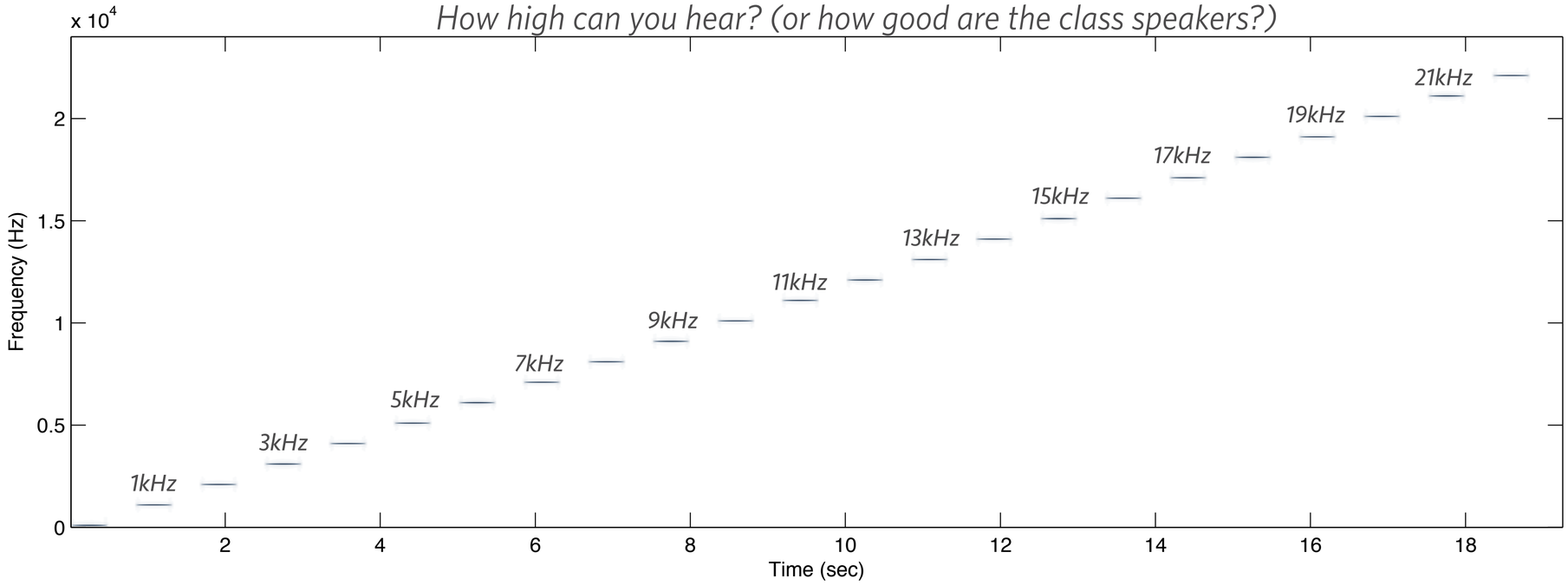
What is aliasing?

- Low sample rates can result in misinterpretations
  - Sample too low and you will miss some of the action
  - Rule of thumb: Sample at least at twice the highest frequency



How high should we go?

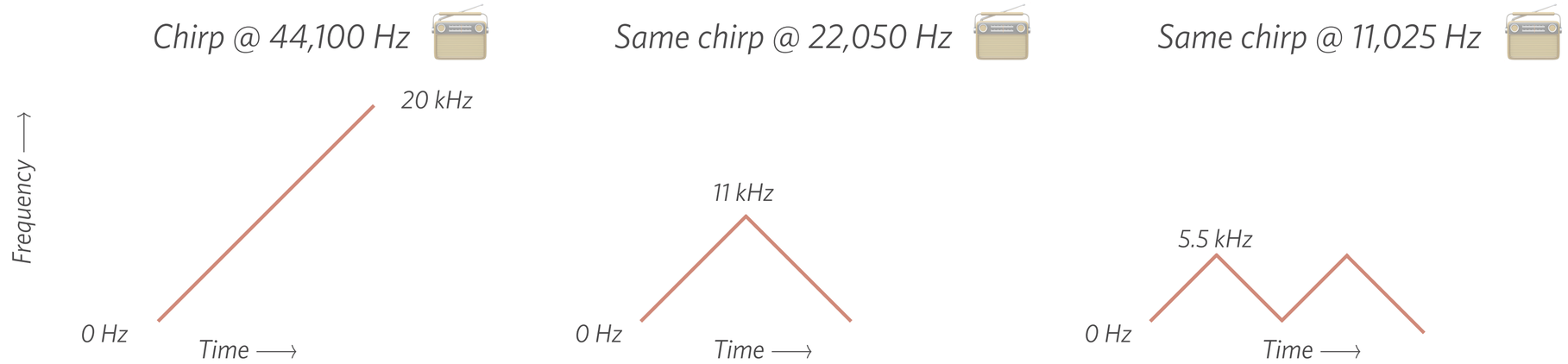
- Highest perceived frequency by humans is 20 kHz
  - Which goes down as you age (or as you abuse your ears)



  - We need to represent up to 20 kHz ⟶ sample at > 40 kHz

What does aliasing sound like?

- Frequencies higher than *Nyquist* fold over
  - Upwards movements go downwards and vice-versa



- Most noticeable with high-frequency content
  - How does that sound?

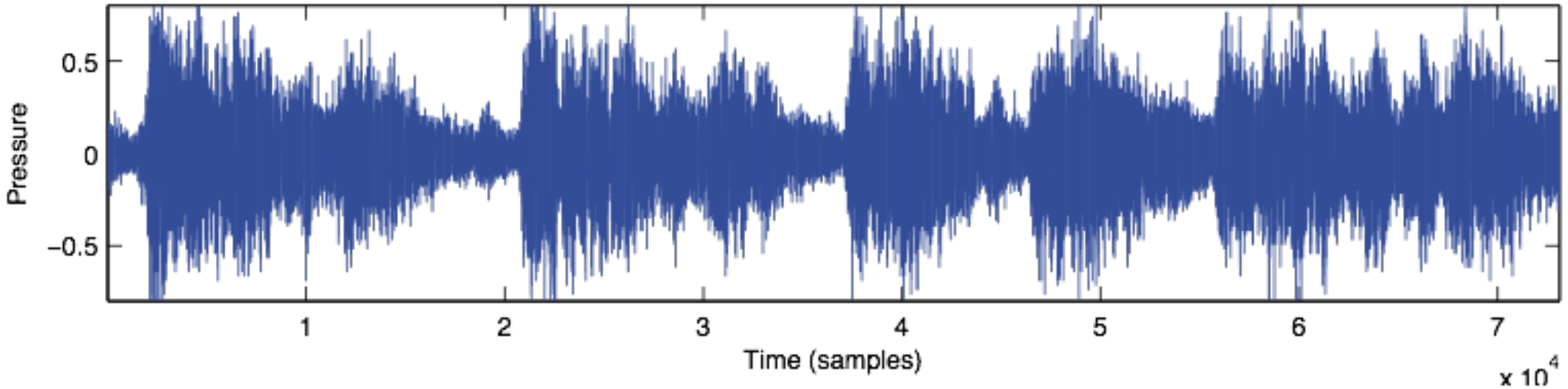
at 44.1kHz at 22kHz at 11kHz at 5kHz at 4kHz at 3kHz

What are the usual settings?

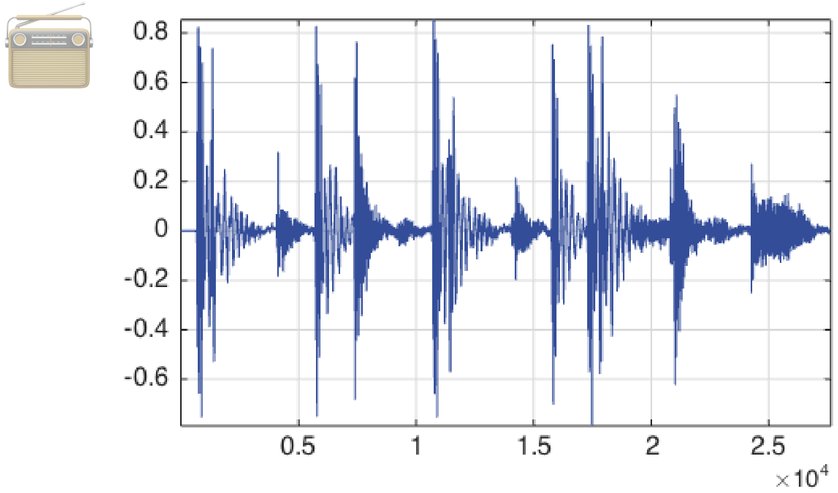
- "High-quality" music: 44.1 kHz
  - Why the extra 4.1 kHz?
- "Super" high quality music: 96 kHz
  - Dogs might like it more
- Speech coding
  - High(ish) quality & in research: 16 kHz
  - Telephony: 8 kHz

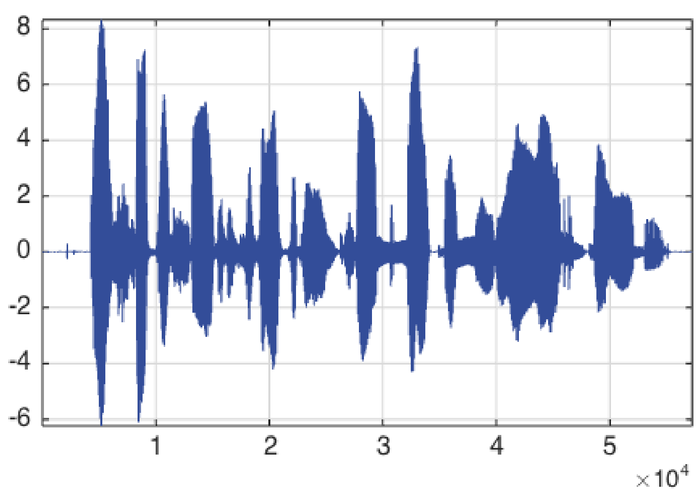
But why do we use the waveform?

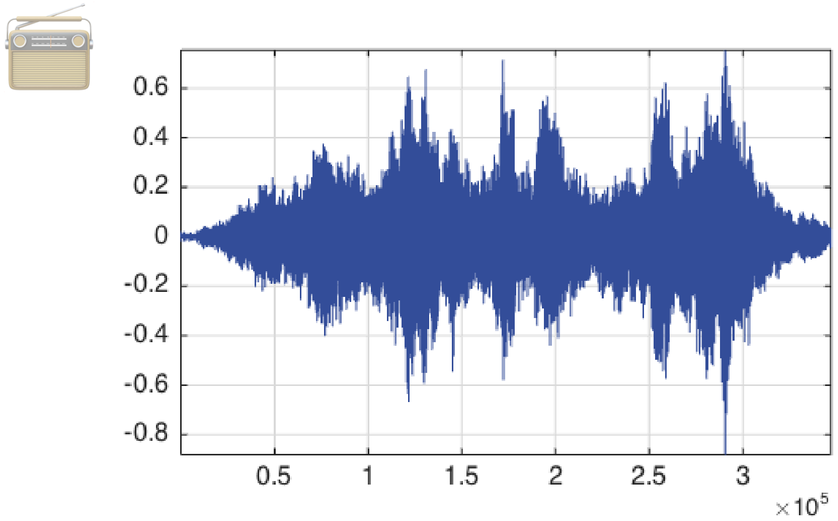
- Do you see a problem with it?

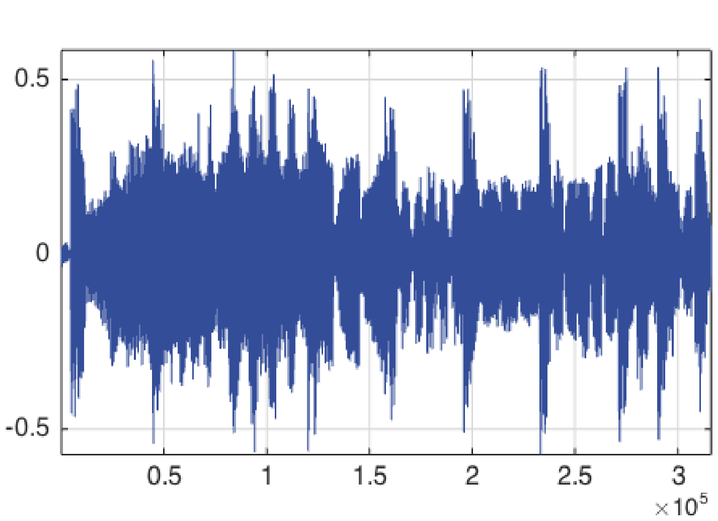


What are these signals?









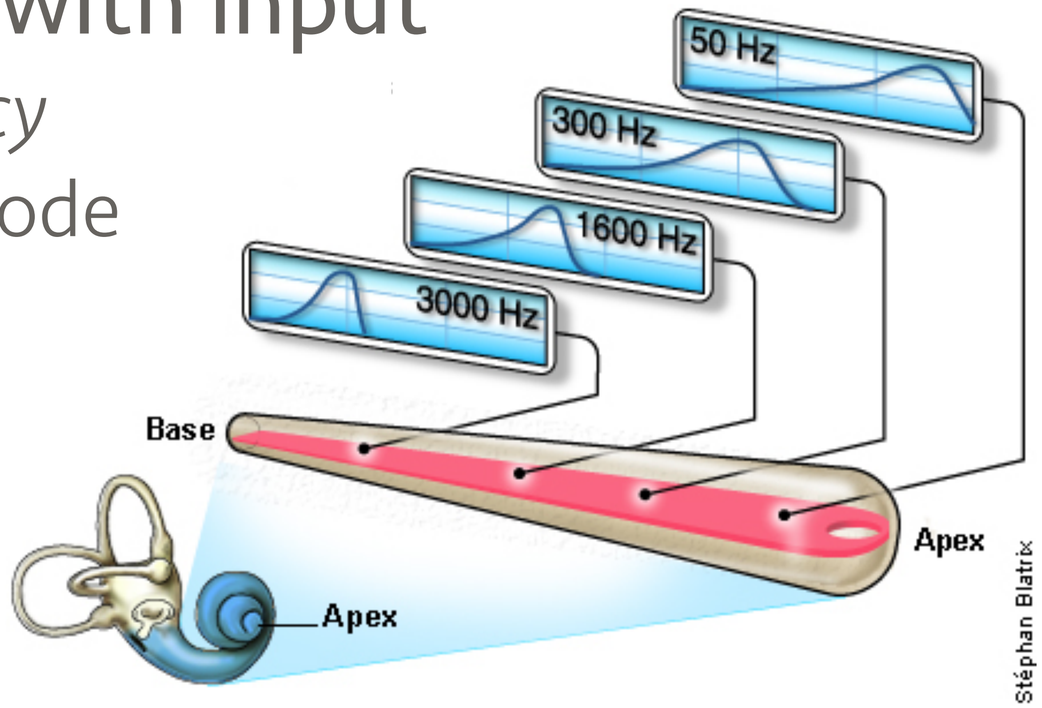
Waveforms are unintuitive at long scales

- Pressure information isn't that perceptually relevant
  - We cannot interpret it as a percept
  - Too much data to parse visually

- Is there a better way to represent sound?
  - How do we start looking for such a way?
  - What is it that is important when listening?

Back to hearing …

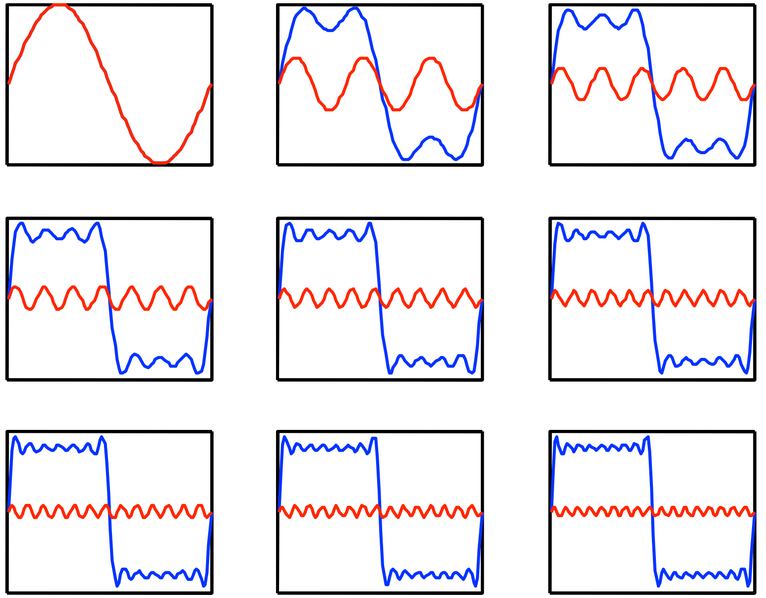
- What happens in the inner ear?
  - After the oval window there's the *cochlea*
- Resonates at different lengths with input
  - Effectively parses sound by *frequency*
  - Transmits that vibration to neural code
- What we care about is frequency content!



What is a frequency component?

- You can approximate any waveform by adding sinusoids
  - They are the elementary building blocks of sounds
- Sinusoids have three parameters:
  - Amplitude, frequency and phase
  - $s(t) = a(t)\ \sin(f\,t + \varphi)$
- Each sinusoid is a “frequency”
  - Because that is the main distinguishing parameter

Approximating a square wave



Decomposing sounds to sines

- For each sound get reconstructing sine parameters
  - And we'll be lazy and not bother with frequency
    - Just get all amplitudes and phases for all integer frequencies
- For this we use the *Fourier transform*
  - Transforms time samples to the *frequency domain*, and back

"Spectrum" (frequency domain)

$$X[f] = \mathrm{FT}\left(x[t]\right)$$

Waveform (time domain)

$$x[t] = \mathrm{FT}^{-1}\left(X[f]\right)$$

And there are many flavors of it

- Fourier transform (Continuous time $\longleftrightarrow$ Continuous frequency)

$$x(t) = \frac{1}{2\pi} \int_{\omega=-\infty}^{\infty} X(\omega) e^{j\omega t}\, d\omega \leftrightarrow X(\omega) = \int_{t=-\infty}^{\infty} x(t) e^{-j\omega t}\, dt$$

- Discrete Time Fourier Transform (DTFT) (Discrete time $\longleftrightarrow$ Cont. frequency)

$$x[n] = \frac{1}{2\pi} \int_{\omega=-\pi}^{\pi} X_d(\omega) e^{j\omega n}\, d\omega \leftrightarrow X_d(\omega) = \int_{t=-\infty}^{\infty} x[n] e^{-j\omega t}\, dt$$

- Discrete Fourier Transform (DFT) (Discrete time $\longleftrightarrow$ Discrete frequency)

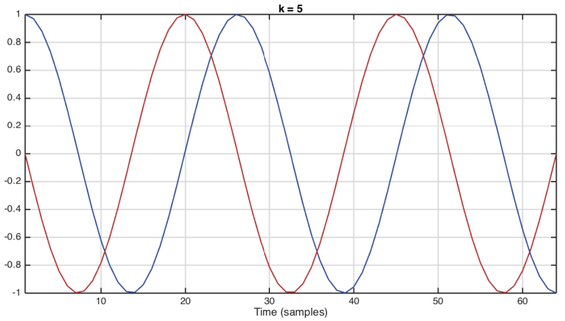
$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi kn}{N}} \leftrightarrow X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi kn}{N}}$$

The one that we will use the most

What really happens here?!?

- Each Fourier basis contains a sine and a cosine

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

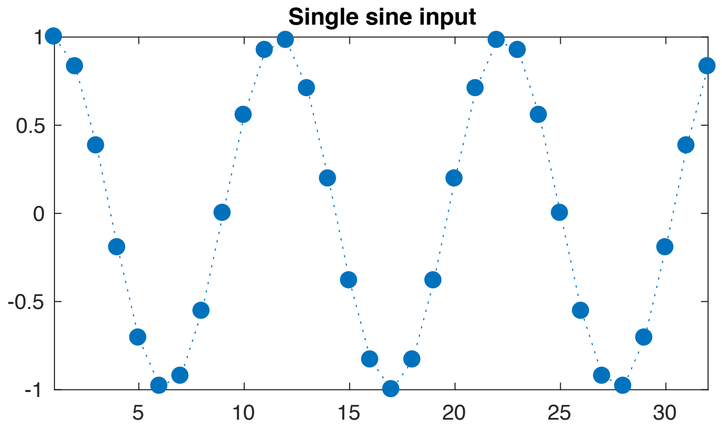


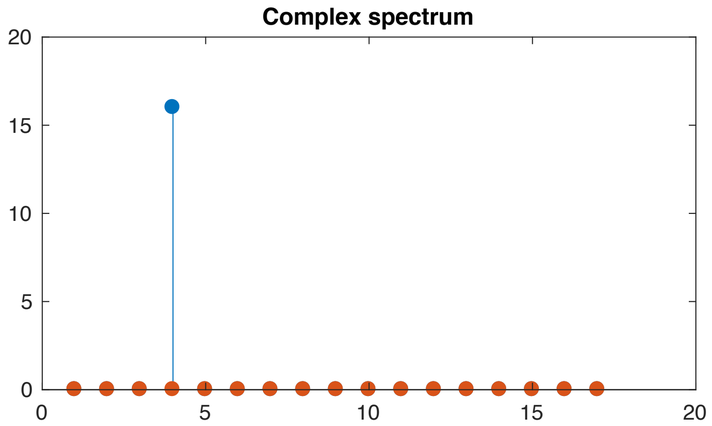
- The summation estimates how much of each sinusoid we have in the input time series (inner product)
  - Thus we get the contribution from each frequency

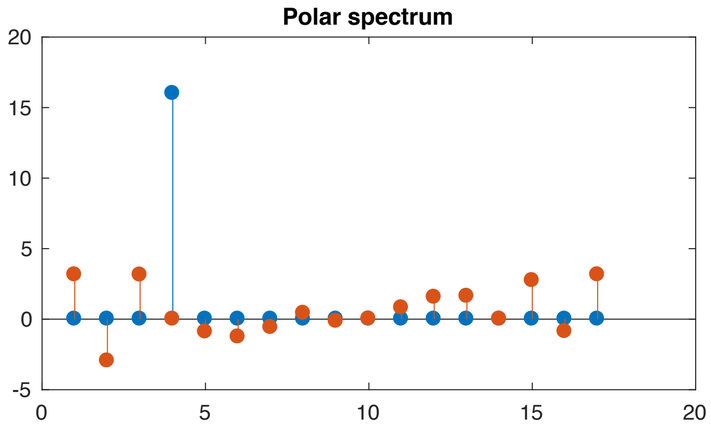
Getting to the sought-after parameters

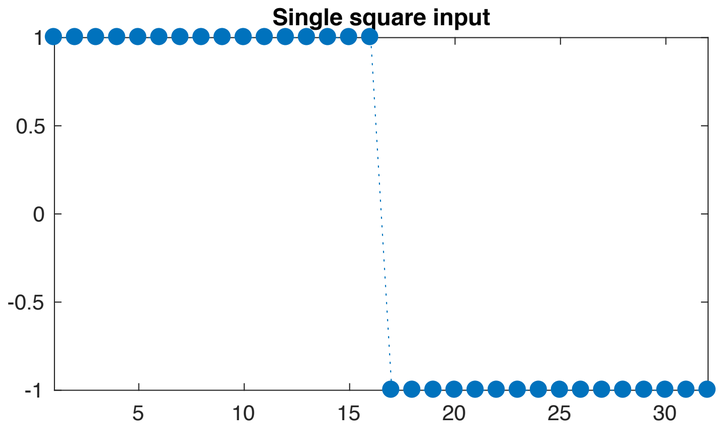
- The magnitude spectrum is $|X[k]|$
  - Tells us how much of each frequency we have (amplitudes)
    - Adding the sine and cosine terms we make phase-shifted sinusoids
      - The contribution of each frequency is the amount of sine/cosine present
- The phase spectrum $\measuredangle X[k]$
  - Gives us each frequency's phase
    - Does so by looking at the relative amplitudes of the same-frequency sine/cosine pairs

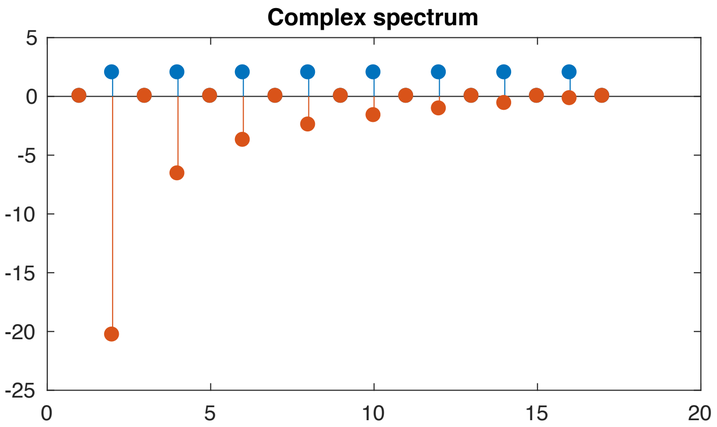
Some examples

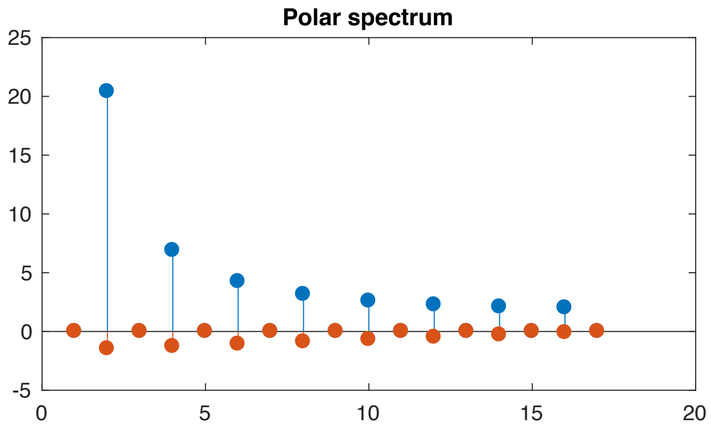










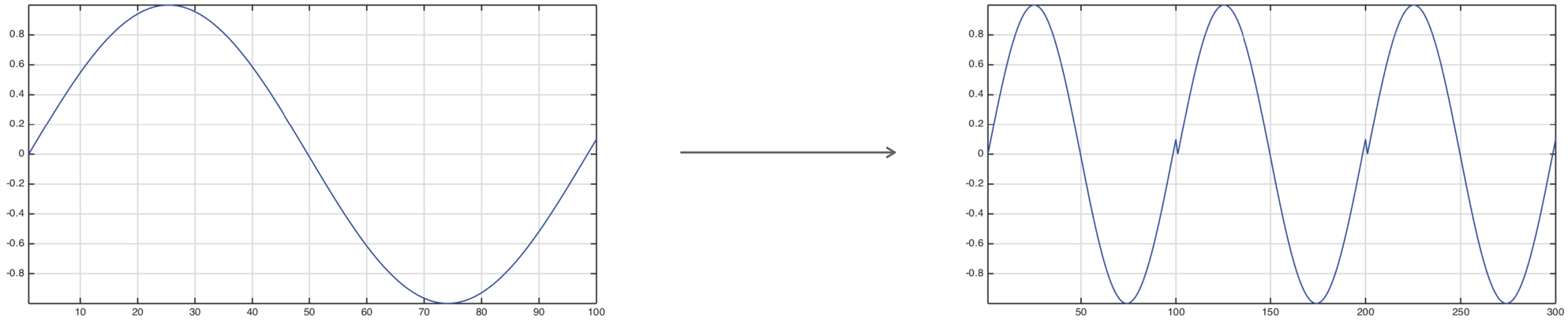


Some extra info

- Audio is real-valued
  - DFT results in a conjugate symmetric transform
    - Upper half is redundant (real-valued routines will give you lower half)
- The first frequency bin is the *DC*
  - Offset of the input signal (“zero” frequency)
  - Doesn’t have a phase value (why?)
- The highest frequency bin is the *Nyquist*
  - Also as no phase value (why?)

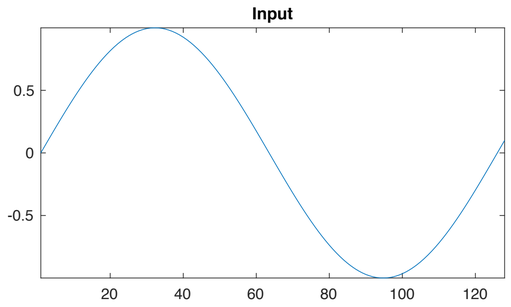
One problem

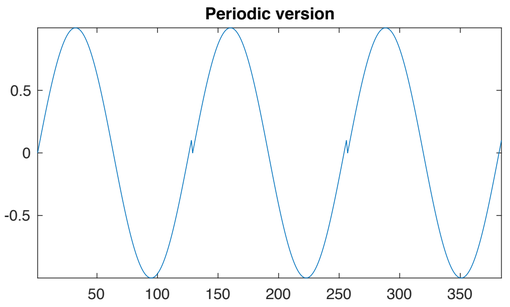
- Sinusoids extend infinitely on both sides
  - i.e. what we approximate is assumed to be periodic
  - So it should transition smoothly from left to right
- We need to ensure smoothness here
  - Discontinuities will result in extra high frequencies

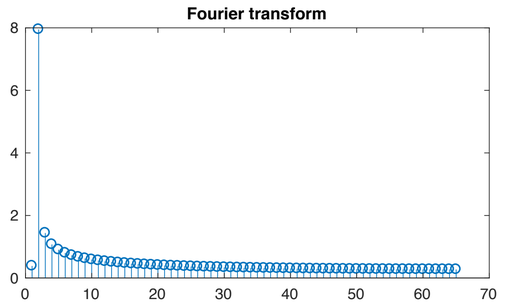


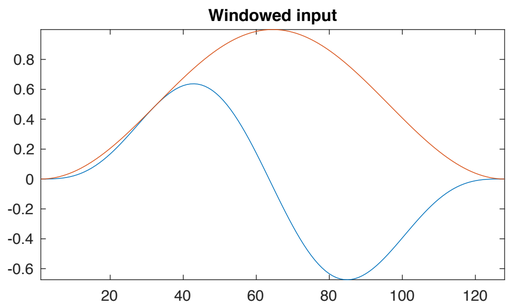
Windowing

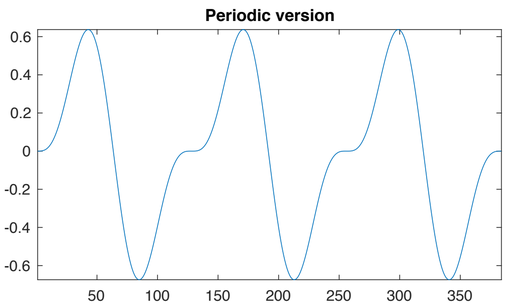
- To avoid periodic discontinuities we can “window”
  - Taper the ends of to zero so that they join better
- But too much tapering changes the signal!
  - Common side effect: “blurring the spectrum”

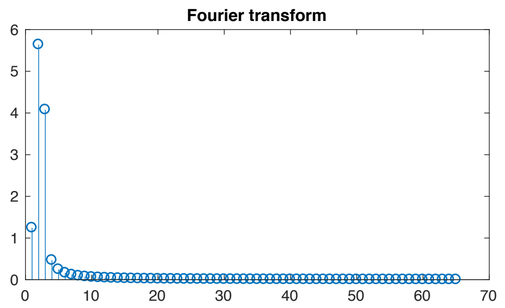






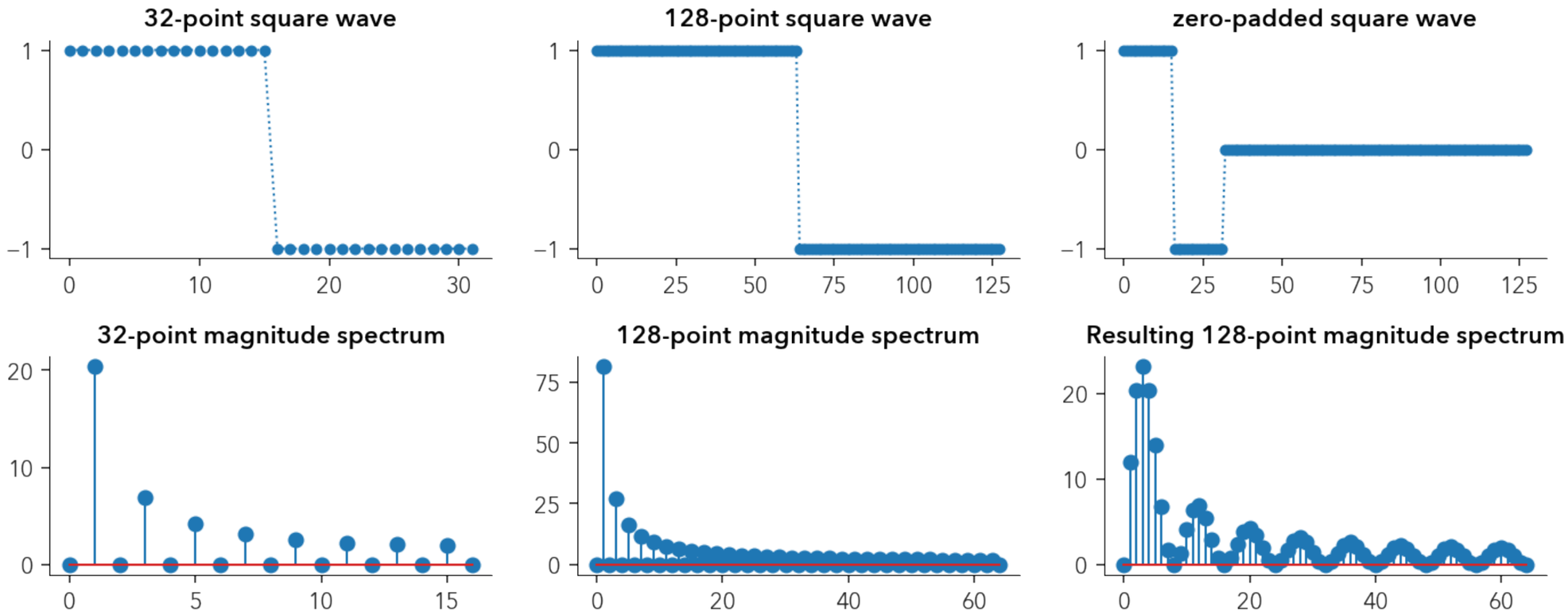






Zero-padding

- Fourier transforms map *N* points to *N* points
  - What if we want to get more outputs? Zero padding!
    - Zero-padding interpolates the frequency domain



Some useful Fourier properties

- Additivity:

$$x[n] \xleftrightarrow{\text{DFT}} X[k], y[n] \xleftrightarrow{\text{DFT}} Y[k] \Rightarrow ax[n] + by[n] \xleftrightarrow{\text{DFT}} aX[k] + bY[k]$$

- Shifting:

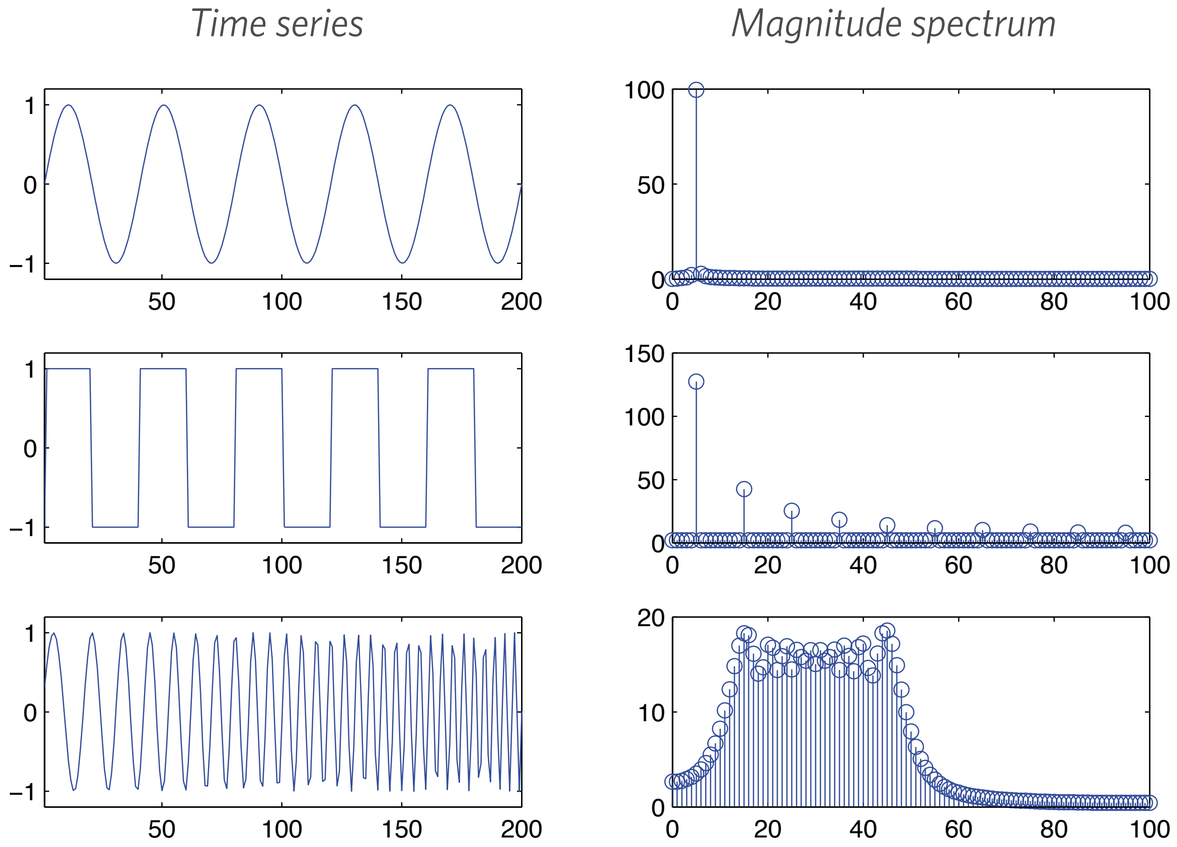
$$x[n] \xleftrightarrow{\text{DFT}} X[k] \Rightarrow x[\ll n - n_o \gg_N] \xleftrightarrow{\text{DFT}} X[k] e^{-j\frac{2\pi k}{N} n_0}$$

- Parseval's theorem

$$x[n] \xleftrightarrow{\text{DFT}} X[k] \Rightarrow \sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |X[k]|^2$$

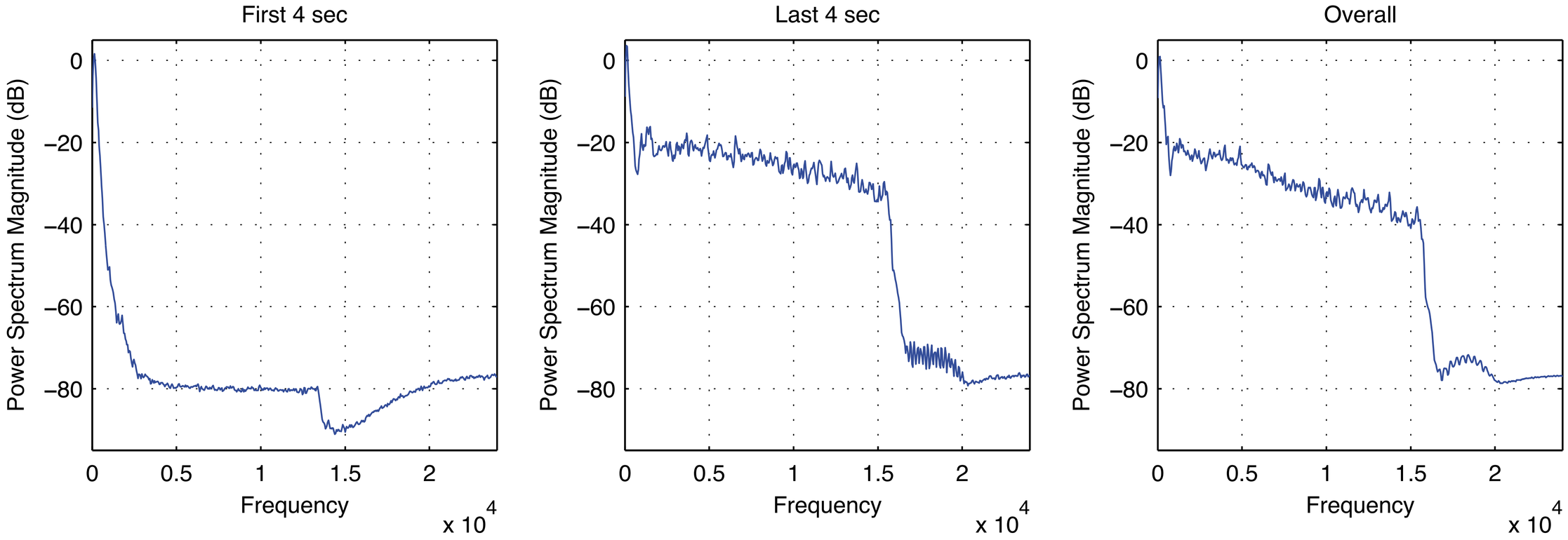
Frequency domain representation

- Representing sounds by frequency content
- Provides a better glimpse of the input
- But provides no temporal information!



On real sounds

- We get a better sense of what's in a signal 
  - But not of the temporal progression

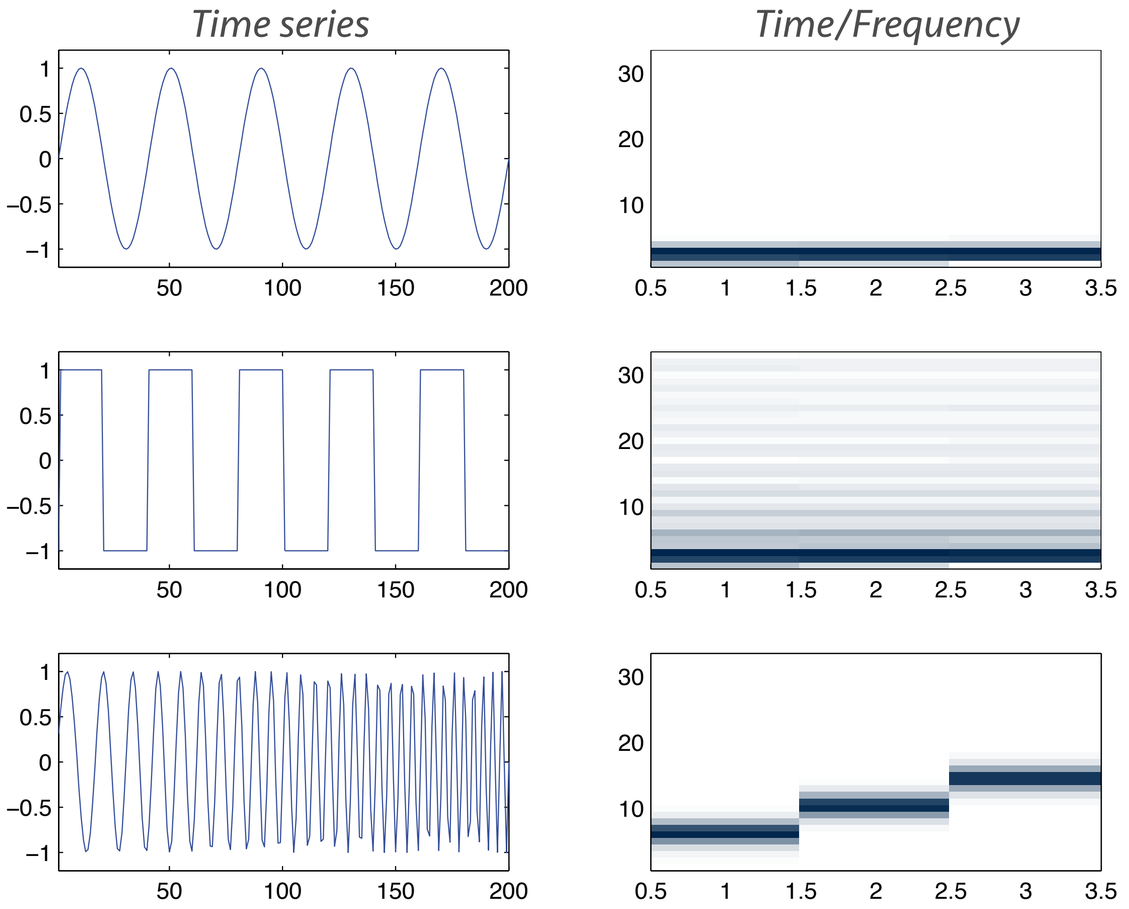


Adding one more dimension

- How about sampling spectra periodically?
  - Each sound segment will have it's own spectrum
    - "Short-time frequency analysis"
- Break input into analysis windows and DFT them
  - Plot all successive spectra side by side
- Keeps time info, but also presents frequency content

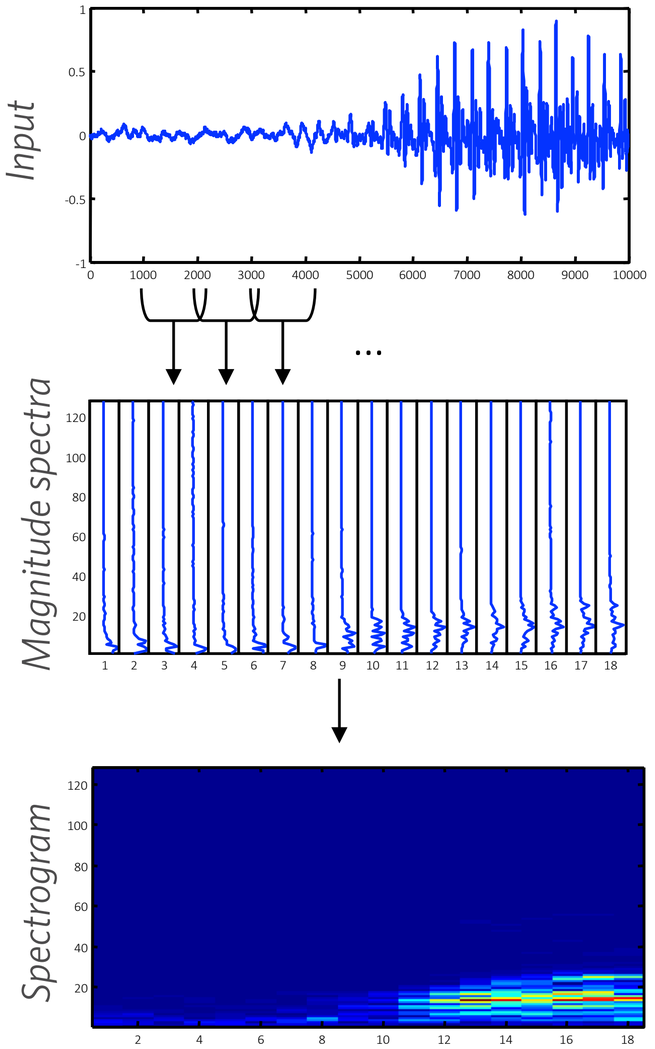
Time/frequency representation

- Many names/varieties
  - Spectrogram, sonogram, periodogram, Short-Time Fourier Transform (STFT), …
- A time-ordered series of frequency compositions
  - Can help show how things change in both time and frequency
- Most useful representation so far!
  - Reveals information about the frequency content without sacrificing the time information

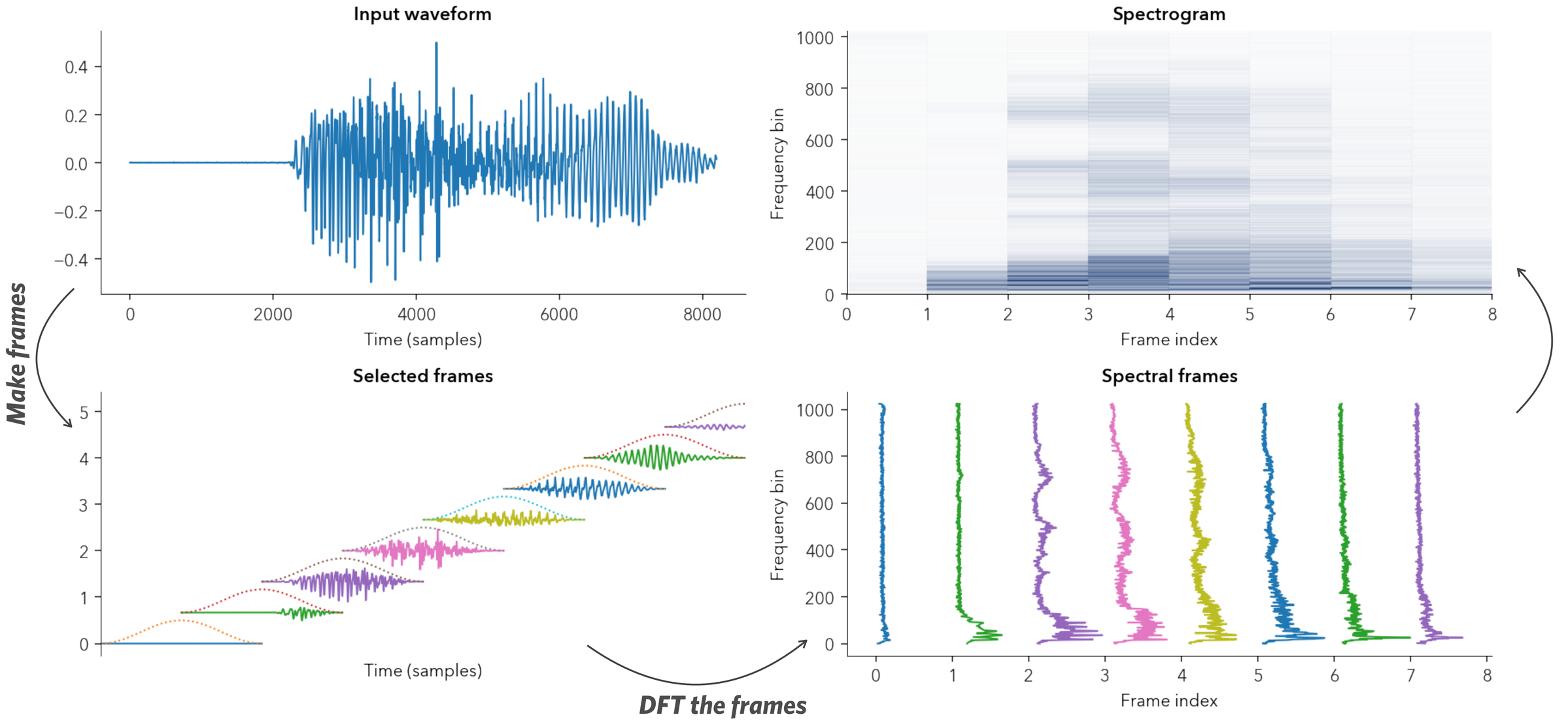


The details

- Get *N* samples, advance *H* samples, repeat
  - *N* is transform size, *H* is hop size
- On each *N*-sample frame apply window
  - Tapers the edges makes for better estimate
- On each windowed frame apply DFT
  - Gets frequency domain of this section only
  - DFT can also be *M*-points (*M* > *N*)
    - Input is zero-padded to length *M*
- Collate all spectra in 2d representation



Pretty picture version

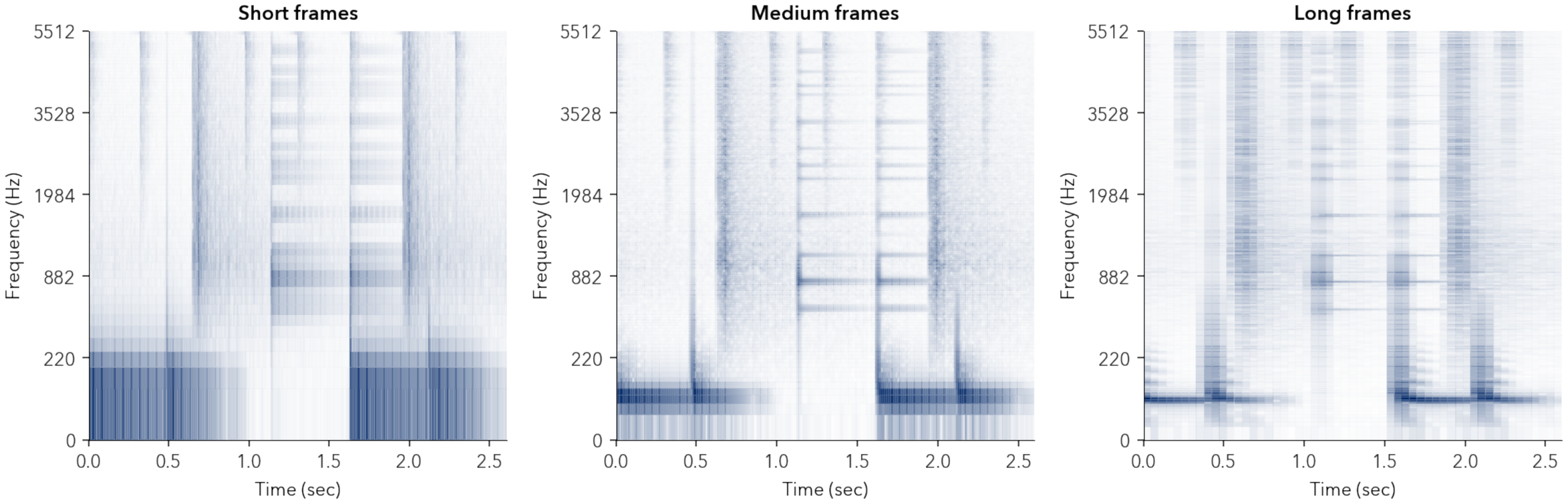


Spectrogram parameters

- Size of the Discrete Fourier Transform
  - Determines how fine the frequency resolution is
  - To get more frequencies we can zero pad
- Hop size
  - Determines how fine the temporal resolution is
  - Should not be larger than transform size! (why?)
- Window
  - Tradeoff between artifacts and frequency resolution
    - Stronger window more blurring, weaker window more artifacts

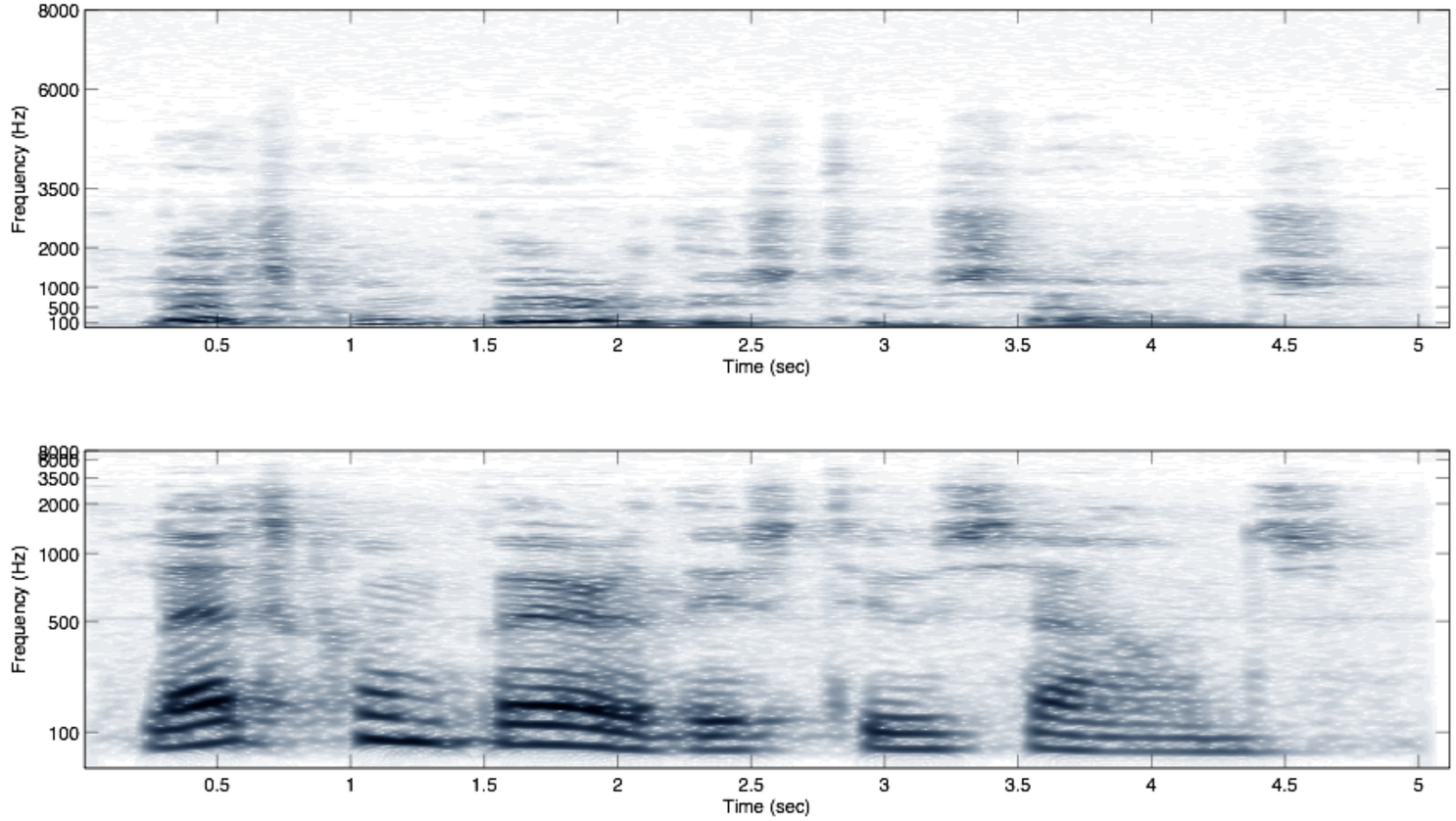
Time/frequency tradeoff

- The more frequencies, the fewer time points 
  - And vice-versa



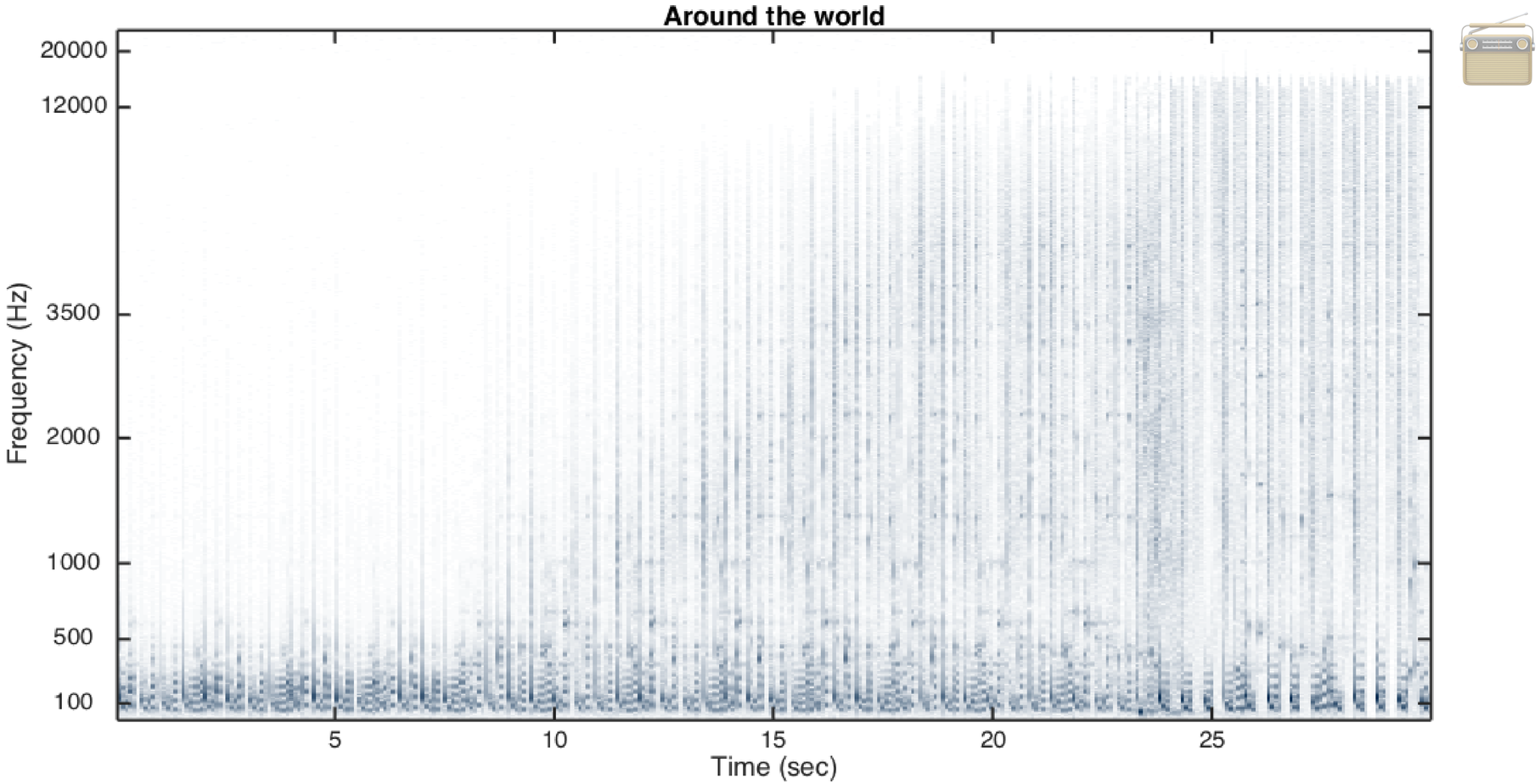
Spectral warping

- Regular spectrograms are hard to read
  - Frequency warping can help (e.g. Mel scale, Bark scale, etc)

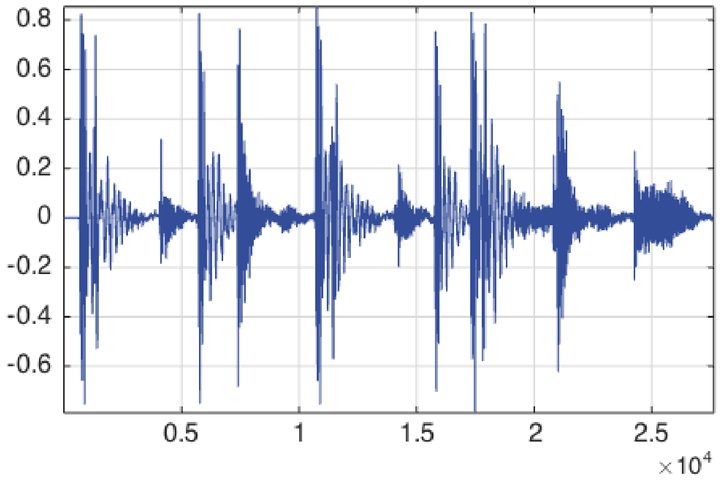


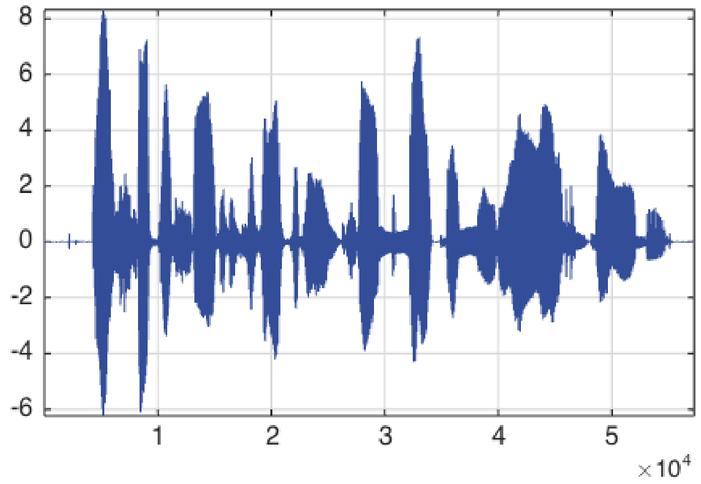
Back to a previous example

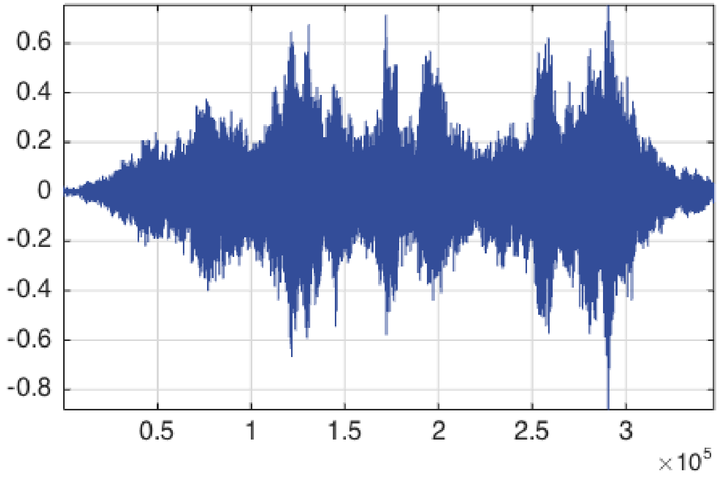
- With the spectrogram we can now see what goes on

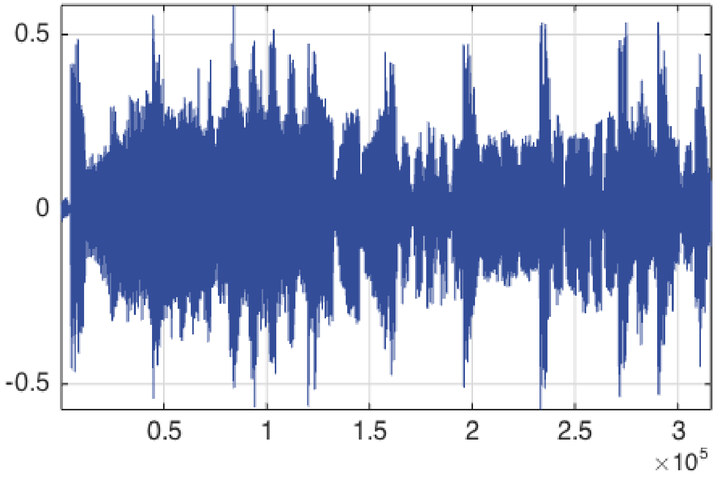


Remember these?

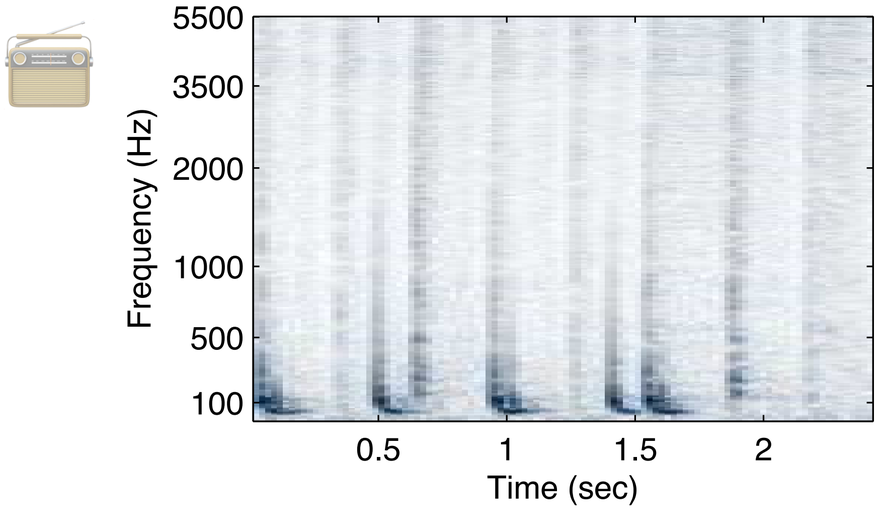


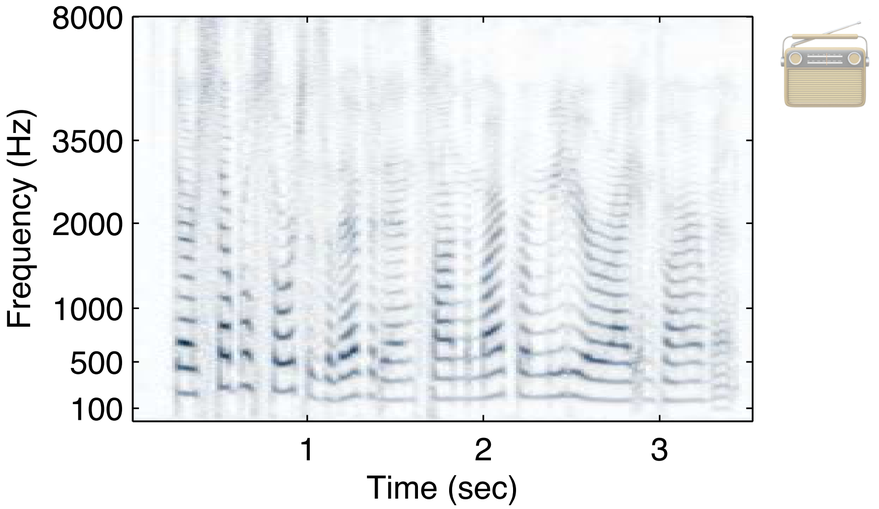


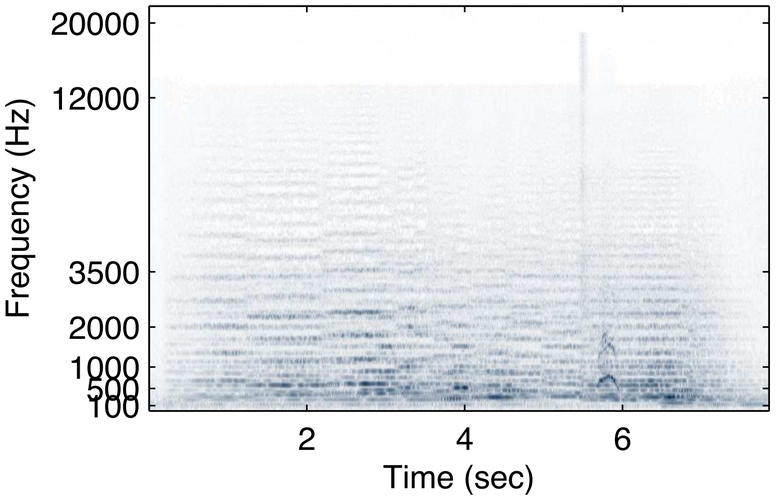


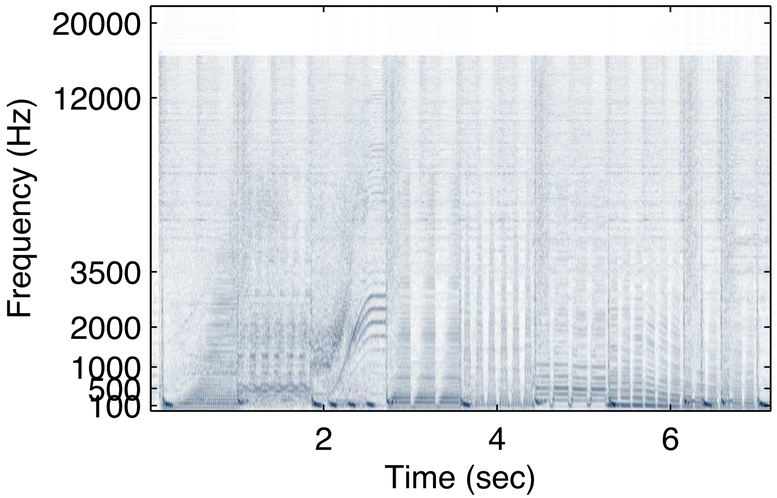


We can now “see” what goes on



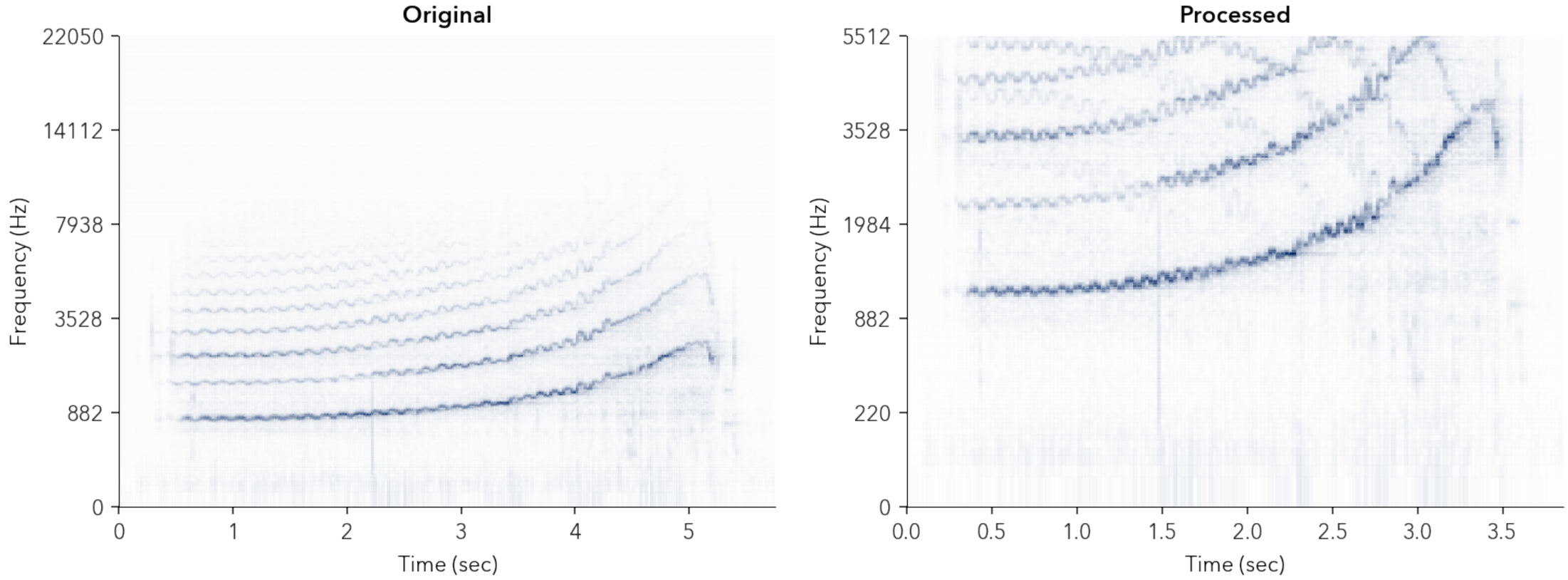




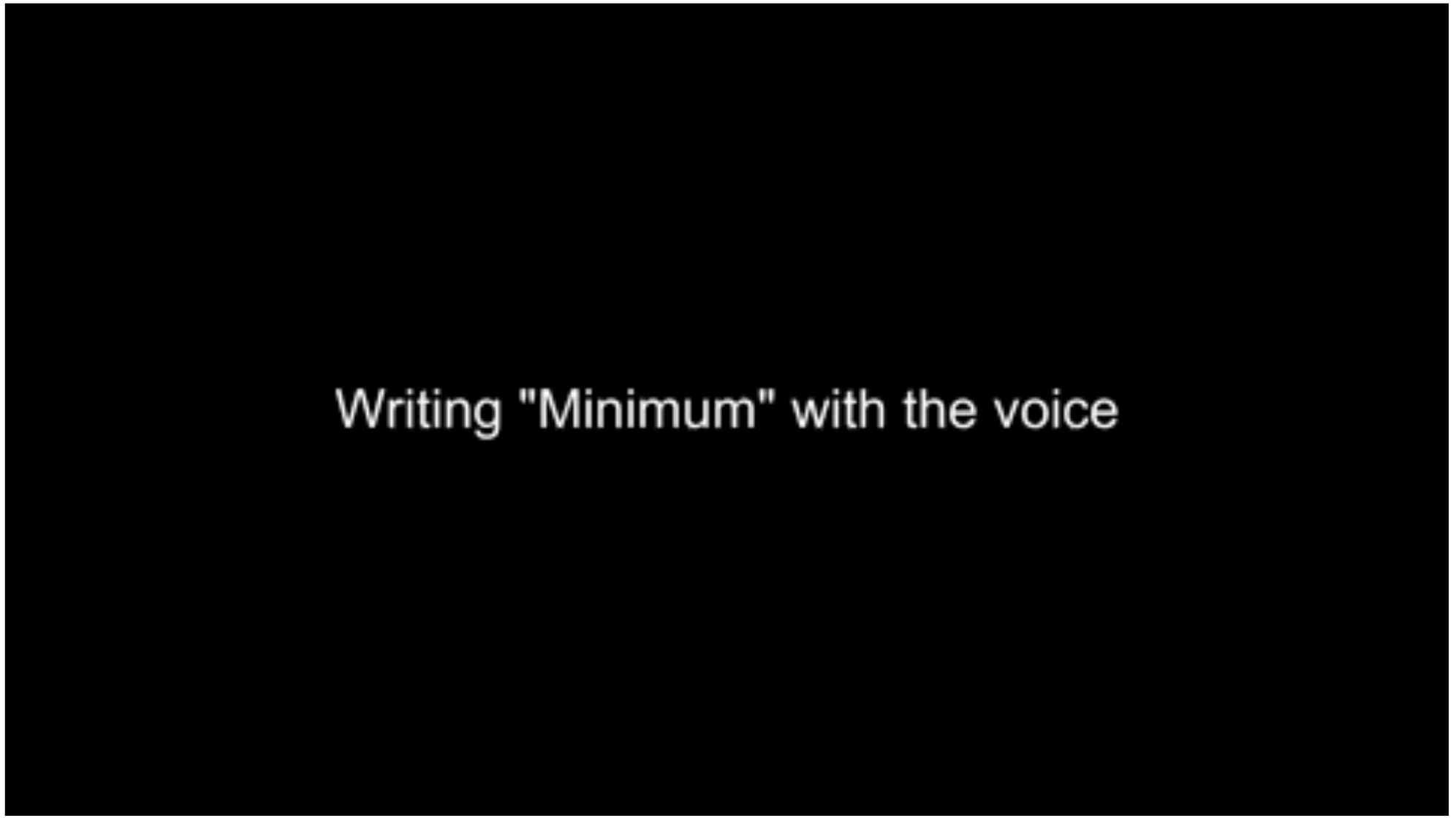


Very useful diagnostic tool!

- Always look at the spectrogram!!
  - Best way to debug audio glitches!



Minimum!!



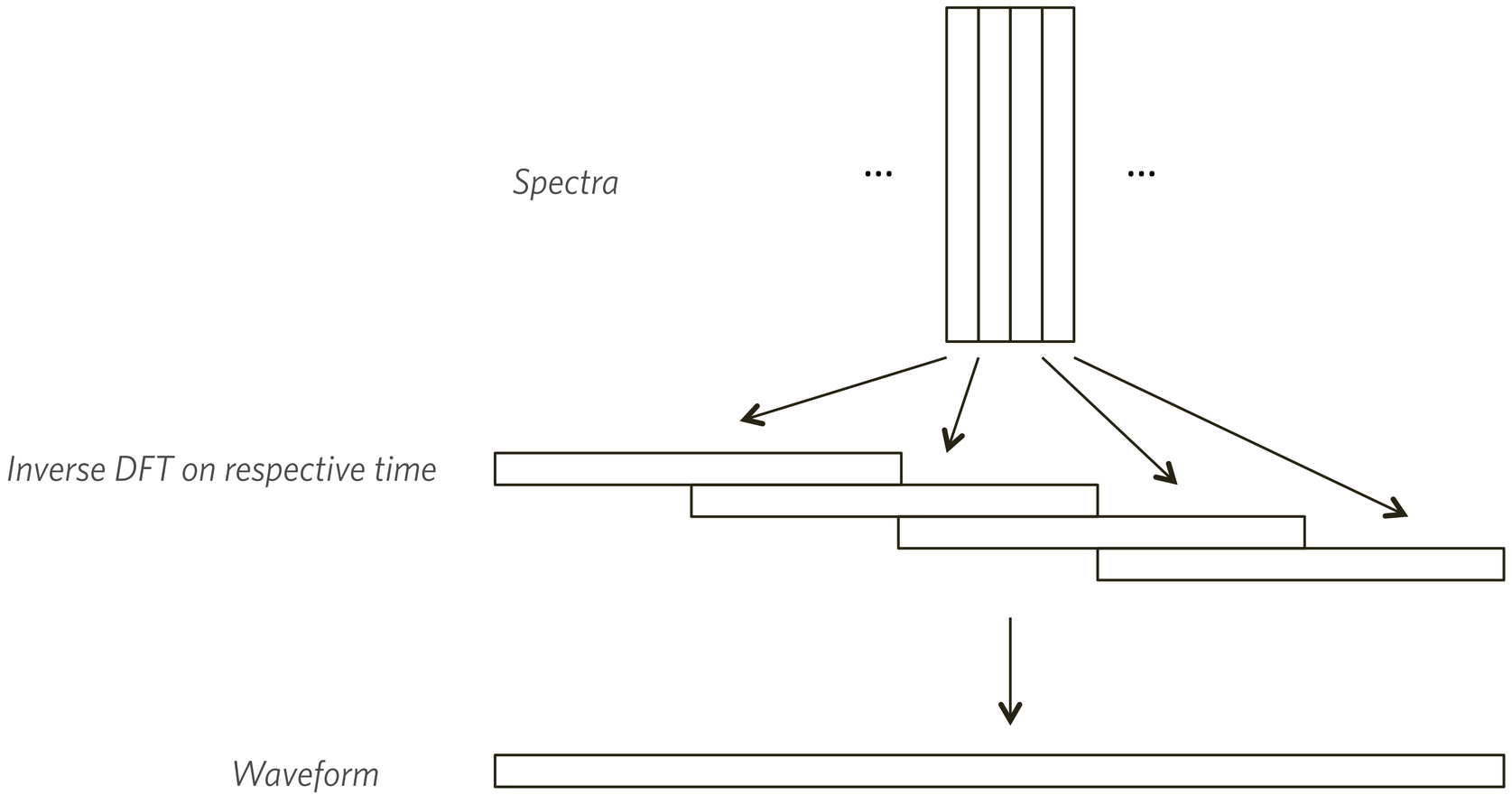
https://www.youtube.com/watch?v=faBFiEfPxUU

The inverse spectrogram

- We can also go from spectrogram to waveform
  - Inverting the spectrogram procedure

- For each (complex) spectral frame
  - Convert to time segment using inverse DFT
  - Optionally apply a window again
    - To “undo” synthesis window, or to avoid processing artifacts
  - “Overlap and add” segments that coincided

Overlap and add



Careful when inverse windowing!

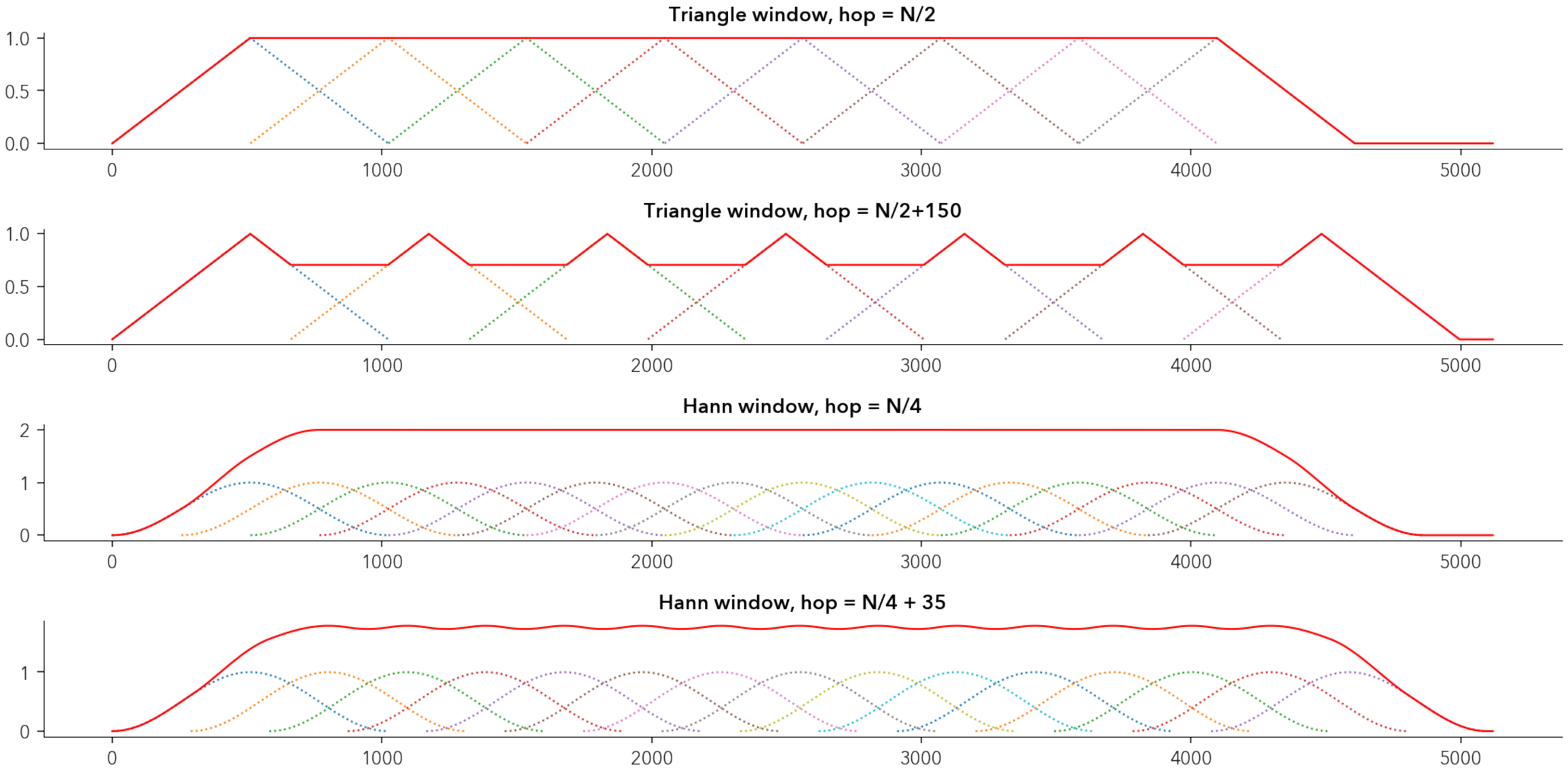
- If you use no windows the output scales with hop size
  - Number of times you overlap-add segments

- If you use windowing you need to satisfy COLA:
  - Constant OverLap Add:

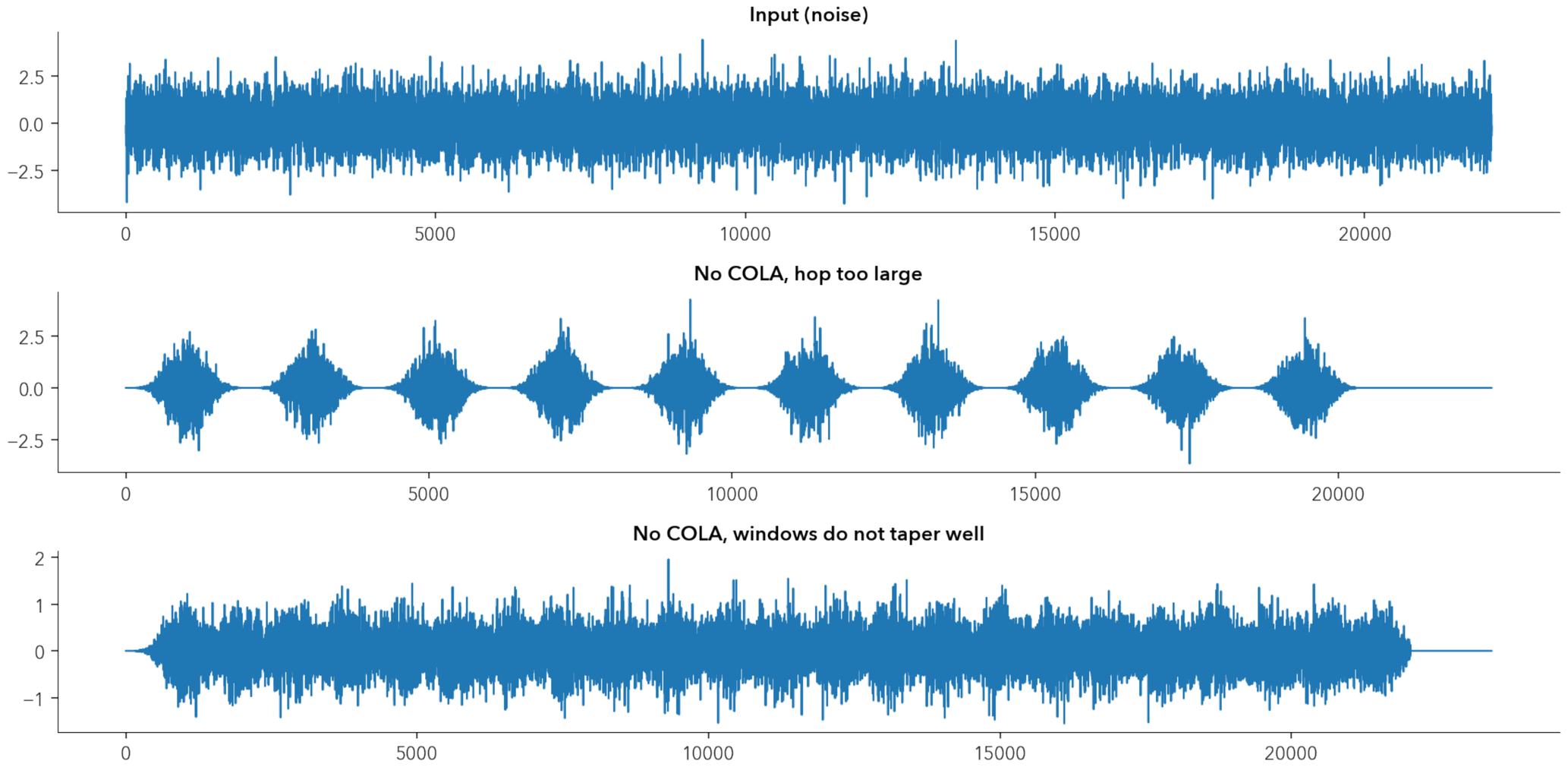
$$\sum_{m=-\infty}^{\infty} w(n - mH) = 1, \forall n \in \mathbb{Z}$$

  - where H is the hop size

What does that mean?



Examples of bad windowing

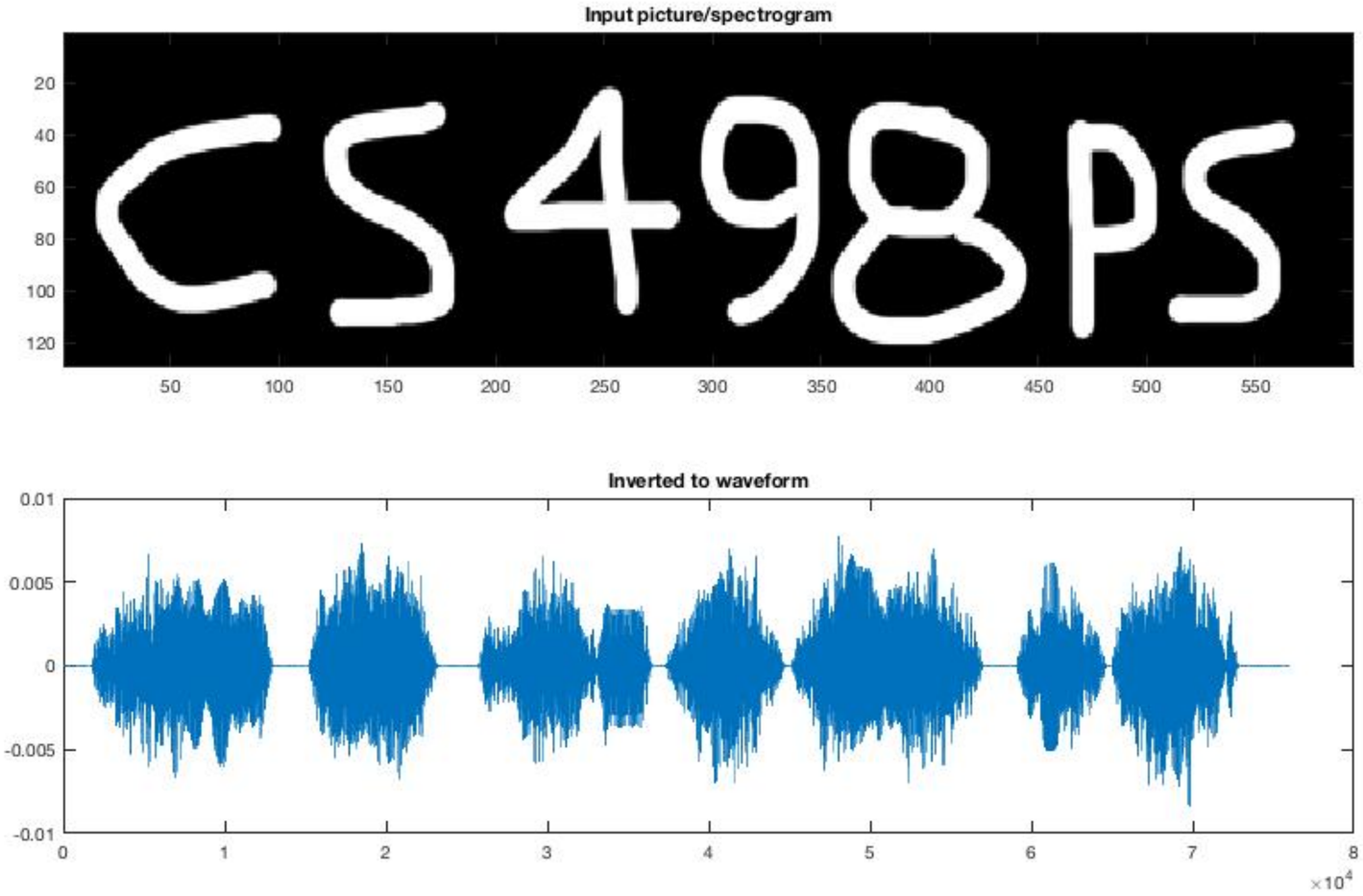


Some uses of inverse spectrograms

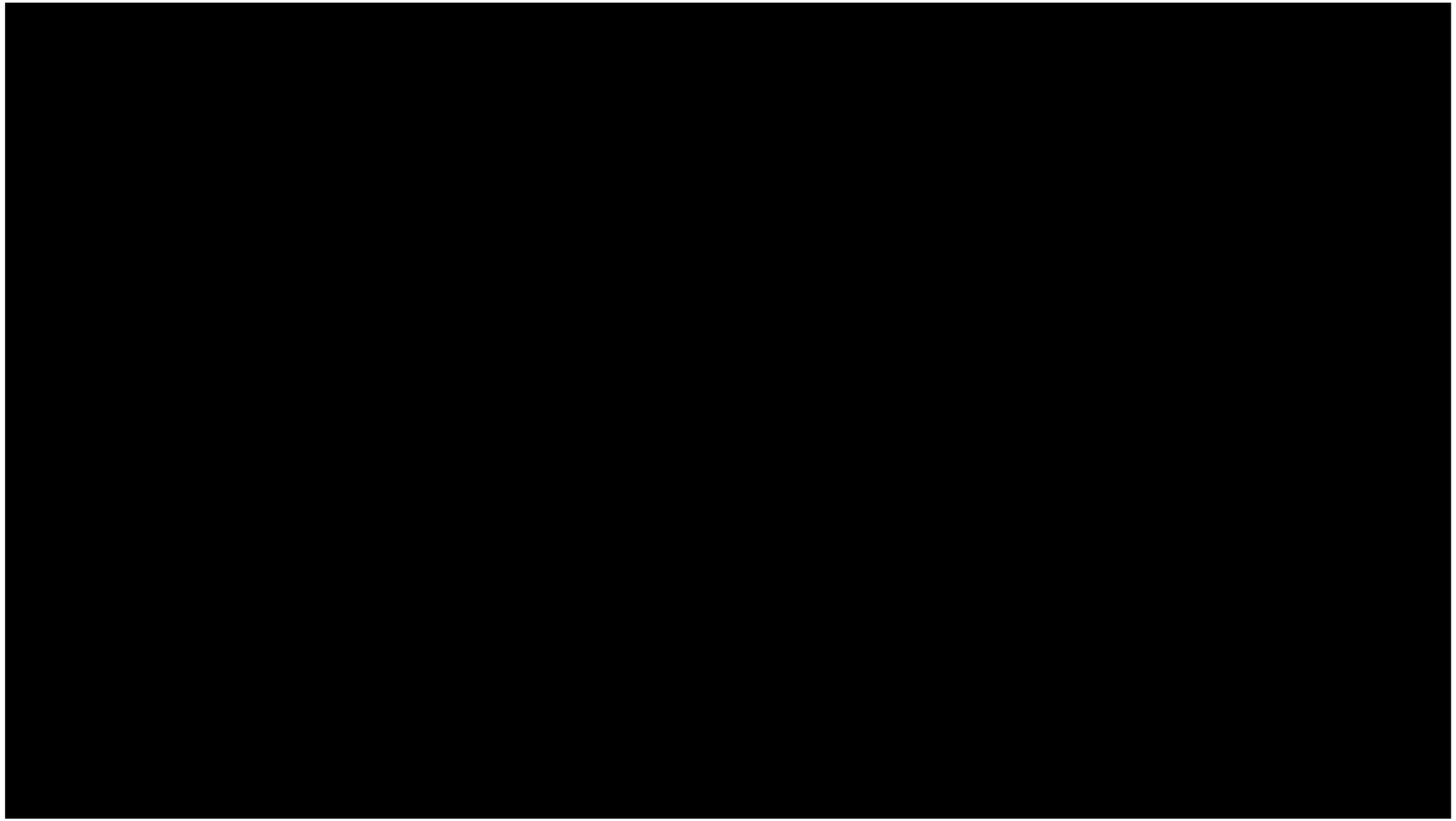
- Useful for spectral editing!
  - Demo
- We will use later for:
  - Denoising
  - Time stretching/compression
  - Spectral manipulations
  - Fast convolutions
  - And many more …

Fun applications

- Pictures to sound

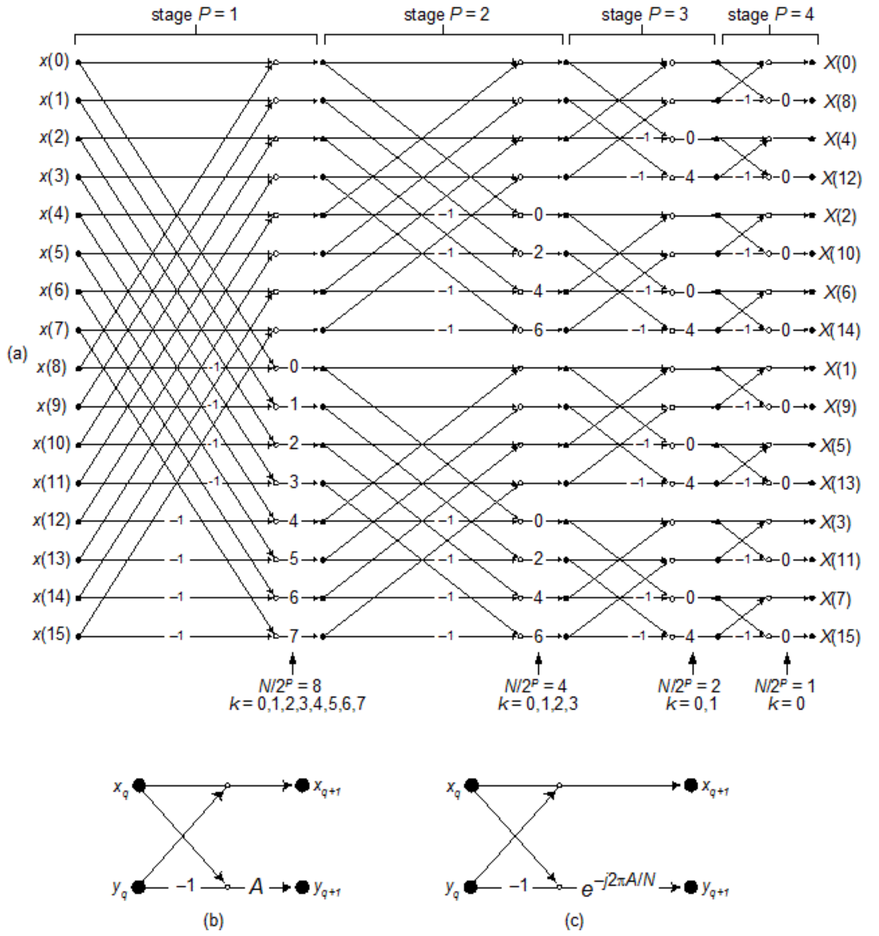


A commercial example



The Fast Fourier Transform (FFT)

- Efficient DFT algorithm
  - Huge speedup! (always use it!)
- Most routines you will find and use will be FFT routines
  - These return full complex spectrum
    - Some are specifically for real inputs
  - You might have to modify for real inputs



Recap

- Digitizing and discretizing audio
  - Basic things to remember to represent sound best
- Frequency analysis and the DFT
- Time-frequency analysis and the spectrogram
  - Also its inverse

Reference material

- Overview of DSP:
  - http://www.dspguide.com/pdfbook.htm

- Spectral analysis of audio:
  - http://www.dsprelated.com/dspbooks/sasp/

Thursday is lab day

- First graded lab
  - Implementing a forward/inverse spectrogram
  - Examining sounds using your code

- Labs administrivia
  - Released on Thursdays, submit solutions within two weeks
  - Send your notebooks via email to me (attached or linked)
    - Use your @illinois email so that I know who you are!!
    - Use subject: “CS498 Lab #” (where # is the lab’s number)
    - Late submissions get zero grade (worst two grades thrown out)