Eng. 100: Music Signal Processing

DSP Lecture 6

Lab 3: Spectra/Spectrogram (Continued)

- Curiosity:
  - https://youtu.be/IvUU8joBb1Q
  - http://www.soundhound.com Song from humming
  - http://dx.doi.org/10.1109/MSPEC.2015.7049439 [1]
    Rain forest logging sound sensors via cell phones
  - aliasing video!
- Lab 3, HW 3 due this week

Outline

- What: The spectrum of a signal (first class)
  - Part 1. Why we need spectra
  - Part 2. Periodic signals $x(t) = c_0 + \sum_{k=1}^{K} c_k \cos\left(2\pi \frac{k}{T} t - \theta_k\right)$
  - Part 3. Band-limited signals, $K = \lfloor BT \rfloor$
- How: Methods for computing spectra (second class)
  - Part 4. Sampling rate $S > 2B$
  - Part 5. By hand by solving systems of equations
  - Part 6. Using general Fourier series solution
  - Part 7. Using fast Fourier transform (FFT), *e.g.*, in `Julia`
- Why: Using a signal's spectrum (third class)
  - to determine note frequencies: $f = \frac{k}{N} S$
  - to remove unwanted noise
  - to visualize frequency content (spectrogram)
  - Lab 3

Learning objectives

- Deeper understanding of spectra
- Find pitch (fundamental frequency) from FFT output
- Remove noise from signals using FFT (and inverse FFT)
- Understand whys and hows of spectrograms
- Understand aliasing and $S > 2B$ requirement

Summary of previous lecture(s)

T-periodic, band-limited signals have finite Fourier series with $K = \lfloor BT \rfloor$:

- Sinusoidal form:

$$x(t) = \underbrace{c_0}_{\text{DC}} + \underbrace{c_1 \cos\left(2\pi \frac{1}{T} t - \theta_1\right)}_{\text{fundamental}} + \underbrace{c_2 \cos\left(2\pi \frac{2}{T} t - \theta_2\right) + \cdots + c_K \cos\left(2\pi \frac{K}{T} t - \theta_K\right)}_{\text{harmonics}}$$

- Trigonometric form:

$$\begin{aligned} x(t) = a_0 \; &+ \; a_1 \cos\left(2\pi \frac{1}{T} t\right) + a_2 \cos\left(2\pi \frac{2}{T} t\right) + \cdots + a_K \cos\left(2\pi \frac{K}{T} t\right) \\ &+ \; b_1 \sin\left(2\pi \frac{1}{T} t\right) + b_2 \sin\left(2\pi \frac{2}{T} t\right) + \cdots + b_K \sin\left(2\pi \frac{K}{T} t\right) \end{aligned}$$

- Coefficients in these two forms are related by

$$\begin{array}{lll} a_0 = c_0 & a_k = c_k \cos\theta_k & c_k = \sqrt{a_k^2 + b_k^2} \\ b_0 = 0, & b_k = c_k \sin\theta_k, & \tan\theta_k = b_k / a_k \end{array}$$

because $$a\cos(\phi) + b\sin(\phi) = \sqrt{a^2+b^2}\cos(\phi - \arctan(b/a))$$

and $$c\cos(\phi - \theta) = (c\cos\theta)\cos\phi + (c\sin\theta)\sin\phi$$

- We can use the fast Fourier transform (FFT) method (`Julia`'s `fft` function) to compute these coefficients very quickly from signal samples if $\boxed{S > 2B.}$

FFT summary

For an array `x` of length N with signal samples $\texttt{x} = \left[x\left(\frac{0}{S}\right) \; x\left(\frac{1}{S}\right) \; \cdots \; x\left(\frac{N-1}{S}\right) \right]$
taken from a signal $x(t)$ that is periodic and band-limited with $S > 2B$:

`(2/N)*real(fft(x))` gives us the "a_k" coefficients in the trigonometric form:

$$[2a_0 \; \underbrace{a_1 \; a_2 \; \ldots \; a_{N/2-2} \; a_{N/2-1}} \; a_{N/2} \; \underbrace{a_{N/2-1} \; a_{N/2-2} \; \ldots \; a_2 \; a_1}]$$

`(-2/N)*imag(fft(x))` gives us the "b_k" coefficients in the trigonometric form:

$$[0 \; \underbrace{b_1 \; b_2 \; \ldots \; b_{N/2-2} \; b_{N/2-1}} \; 0 \; \underbrace{-b_{N/2-1} \; -b_{N/2-2} \; \ldots \; -b_2 \; -b_1}]$$

`(2/N)*abs.(fft(x))` gives us the "c_k" coefficients in the sinusoidal form (usually use this one!):

$$[2c_0 \; \underbrace{c_1 \; c_2 \; \ldots \; c_{N/2-2} \; c_{N/2-1}} \; c_{N/2} \; \underbrace{c_{N/2-1} \; c_{N/2-2} \; \ldots \; c_2 \; c_1}]$$

`angle.(fft(x))` gives us the phase values in the sinusoidal form:

$$[0(\text{or } \pi) \; \underbrace{-\theta_1 \; \ldots \; -\theta_{N/2-1}} \; 0(\text{or } \pi) \; \underbrace{\theta_{N/2-1} \; \ldots \; \theta_2 \; \theta_1}]$$

Frequency associated with kth term (`Julia` index `l` $= k+1$) is $\boxed{f = \frac{k}{N} S.}$

FFT: Why the mirrored coefficients? (read)

Complex-exponential form of Fourier series of any T-periodic signal: $(\imath = \sqrt{-1})$

$$x(t) = \sum_{k=-\infty}^{\infty} X_k \mathrm{e}^{-\imath 2\pi \frac{k}{T} t}.$$

(EECS 215 uses $j = \sqrt{-1}$.) For a T-periodic signal band-limited to $B = K/T$:

$$x(t) = \sum_{k=-K}^{K} X_k \mathrm{e}^{-\imath 2\pi \frac{k}{T} t}.$$

For an array `x` of length N with signal samples $\texttt{x} = \left[x\left(\frac{0}{S}\right) \; x\left(\frac{1}{S}\right) \; \cdots \; x\left(\frac{N-1}{S}\right) \right]$ taken from a signal $x(t)$ that is periodic and band-limited with $S > 2B$:

`(1/N)*fft(x)` gives

$$[X_0 \underbrace{X_1 \; X_2 \; \ldots \; X_{N/2-2} \; X_{N/2-1}}_{\text{positive } k} \; X_{-N/2} \underbrace{X_{-N/2+1} \; X_{-N/2+2} \; \cdots \; X_{-2} \; X_{-1}}_{\text{negative } k}]$$

because

$$\cos(t) = \frac{1}{2}\mathrm{e}^{\imath t} + \frac{1}{2}\mathrm{e}^{-\imath t}.$$

We will not use the complex-exponential form in ENGR 100. (See EECS 216.)

FFT: Why $f = \frac{k}{N} S$? (read)

Recall sinusoidal form of Fourier series of T-periodic signal with band-limit B:

$$x(t) = c_0 + \sum_{k=1}^{K} c_k \cos\left(2\pi \frac{k}{T} t - \theta_k\right)$$

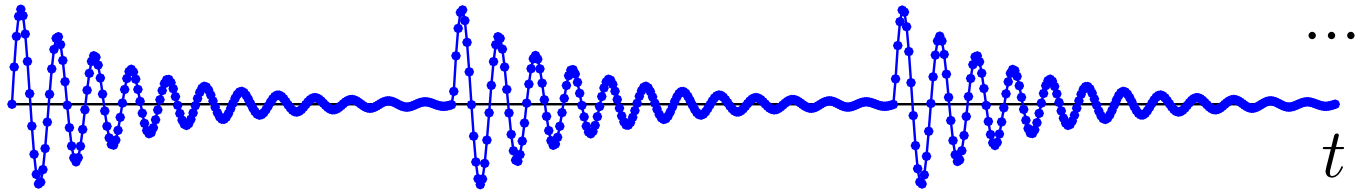
Frequency of kth term in sum is $\boxed{f = \frac{k}{T}.}$

But for unknown pitches we do not know the fundamental period T!

So the formula $f = \frac{k}{T}$ is not immediately useful with the FFT.

FFT: Why $f = \frac{k}{N} S$?

Imagine repeating a segment of audio over and over:



This thought experiment yields a periodic signal with period $T = N\Delta = N/S$ where N is the number of samples.

Substituting $T = N/S$ into the formula $f = \frac{k}{T}$ and simplifying yields

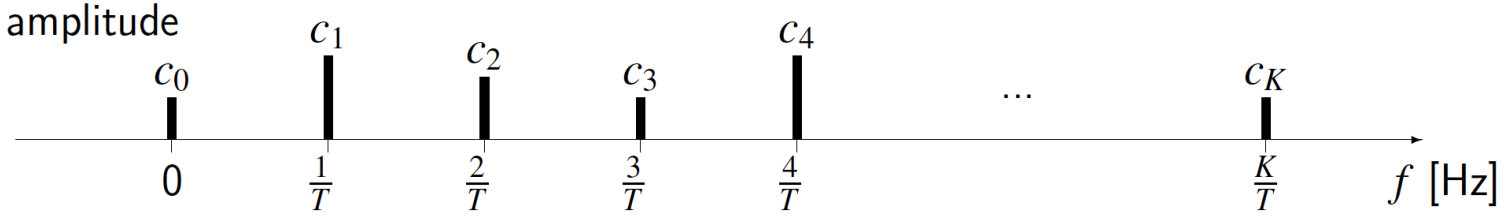
$$\boxed{f = \frac{k}{N} S.}$$

- This formula works perfectly when the unknown frequency f is an integer multiple of S/N.
- If f is not exactly an integer multiple of S/N, then we will see a "bump" in the FFT spectrum instead of a single line.

Line spectra

- Spectrum of a T-periodic, band-limited signal with $K = \lfloor BT \rfloor$:

$$x(t) = c_0 + c_1 \cos\left(2\pi \frac{1}{T} t - \theta_1\right) + c_2 \cos\left(2\pi \frac{2}{T} t - \theta_2\right) + \cdots + c_K \cos\left(2\pi \frac{K}{T} t - \theta_K\right)$$

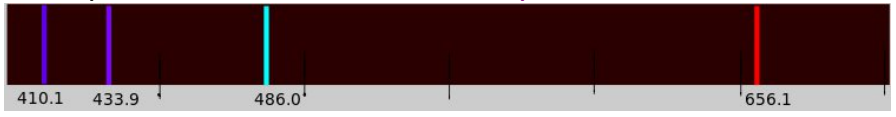


- More generally, line spectra of band-limited signals look something like this:

Formula: $x(t) = c_0 + c_1 \cos(2\pi f_1 t) + \cdots + c_4 \cos(2\pi f_4 t)$

(We assumed $\theta_k = 0$ in this formula because phase is unspecified in spectrum.)

- In physics, often only the frequencies of atomic line spectra are shown, not the amplitudes:



Exercise

Draw the spectrum of this signal:

$$x(t) = 20\sin^2(2\pi 50t) + 30\cos(2\pi 200t)\,.$$

Hint: $\sin^2(\phi) = \frac{1}{2} - \frac{1}{2}\cos(2\phi)$ "power reduction formula"

Spectrum of a live recorded signal

Basic audio recording with `Julia`:

```
using Sound: record, sound
using Plots: plot, default; default(label="")
using FFTW: fft
record(0.001) # warm-up
println("begin")
x, S = record(5)
x = x[1:2:end]; S ÷= 2 # reduce memory
Nx = length(x)
t = (1:Nx)/S
p0 = plot(t, x, xlabel="t [s]", ylabel="x(t)")
```

This records `5` seconds of monaural audio sampled at `S` Hz and stores the results in vector `x`. Requires a microphone.

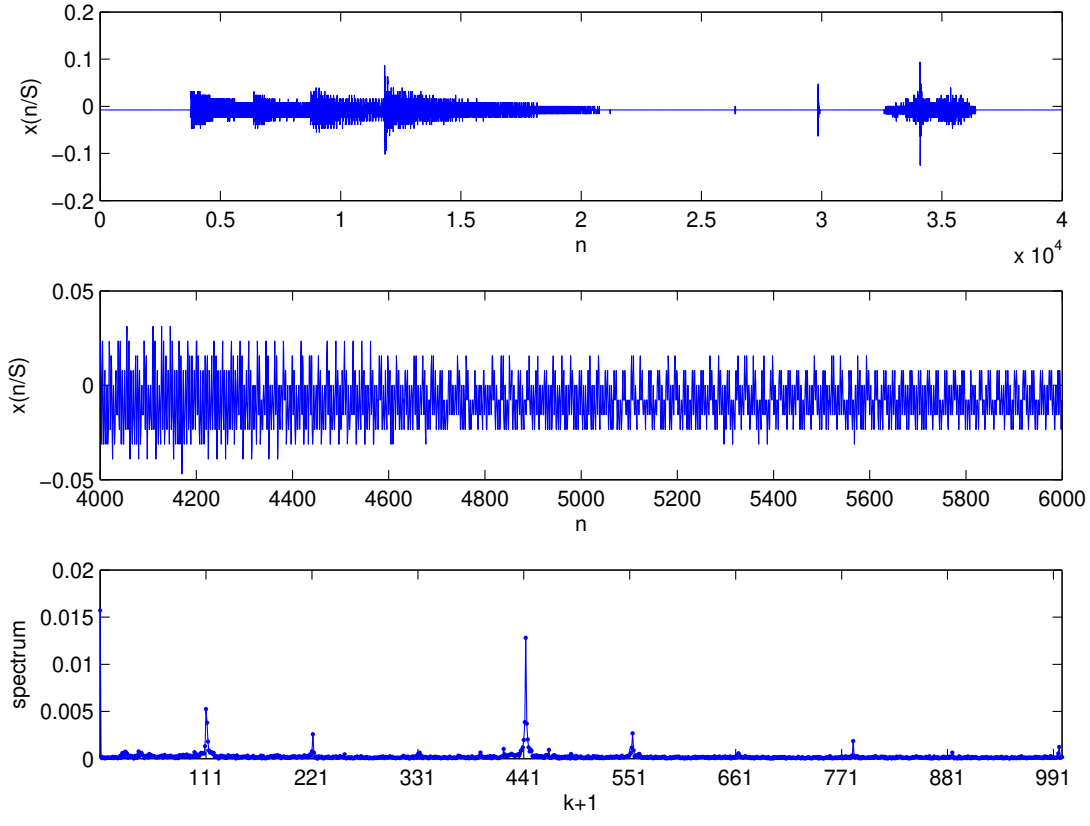
Key formula: $\boxed{f = \frac{k}{N} S}$ where `l` $= k + 1$ [wiki]

(guitar "transcription" demo here using `fig/l05/record2-fft1.jl` `fig/l05/record2-fft2.jl`)

Plotting code

```
using Plots: plot!, default; default(label="")
n = Int[2.0 * S] .+ (1:2000)
p1 = deepcopy(p0)
plot!(p1, t[n], x[n], color=:magenta)
y = x[n]; Ny = length(y) # segment
p2 = plot(y, xlabel="n (sampling rate $S Hz)", ylabel="y[n]")
lmax = 101
p3 = plot(2/Ny * abs.(fft(y)), line=:stem,
    title="spectrum of y (zoomed), fmax=$((lmax-1)/Ny*S) Hz",
    xaxis=("frequency index l=k+1", (0,lmax), 0:5:lmax))
plot(p1, p2, p3, layout=(3,1))
```

DSP, FFT and "Stairway to heaven"



play

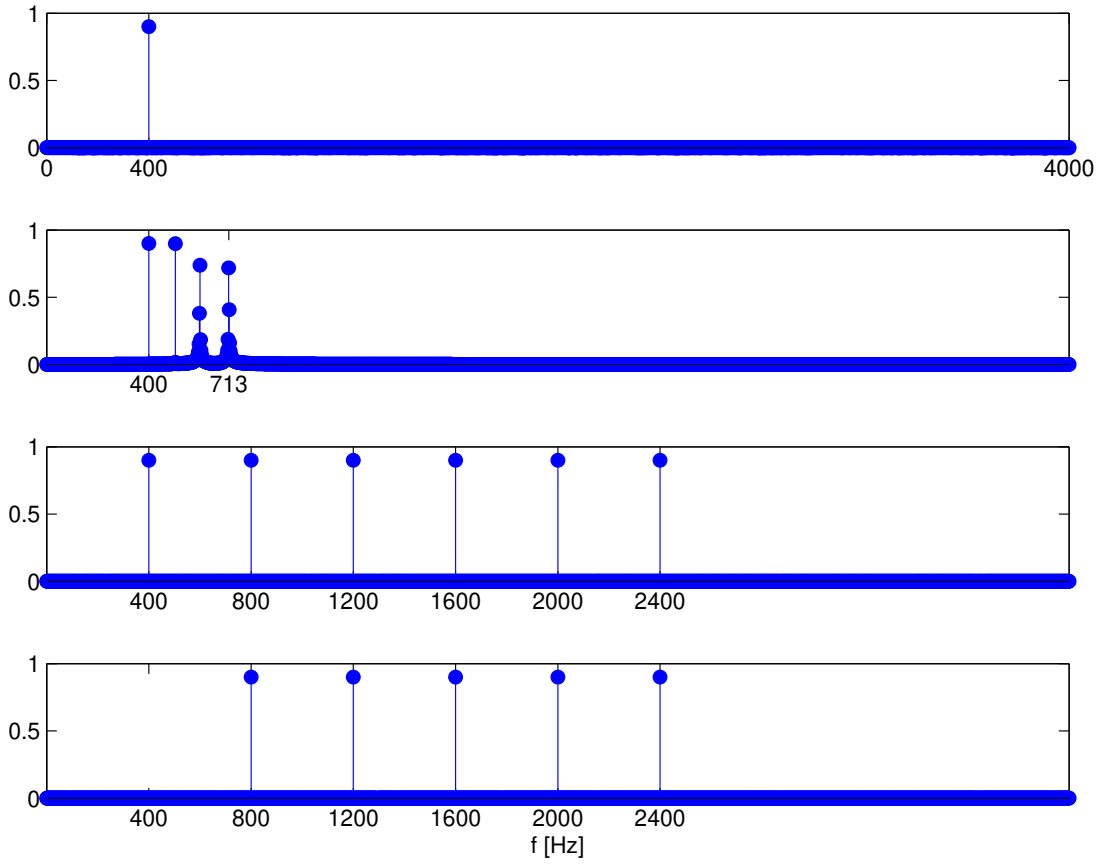
play

Fundamental frequency of first note: $f = \frac{k}{N} S = \frac{110}{2000} 8000 = 440 \text{Hz}.$

Q0.1 What is the lowest (nonzero) frequency (in Hz) we could find here?

??

Listening and seeing



play play play play

Why is the lowest note on a tuba (or large pipe organ) C0 = 16.35 Hz? ?? [wiki]

What is the fundamental period (in sec) of the last signal?

Application: Removing noise from signals (Lab 3)

Removing noise from signals

- Given: signal = desired_signal + noise_signal

$$y(t) = x(t) + \varepsilon(t)$$

- Goal: reduce ("filter out") as much noise as possible
- Assume/know: signal is band limited to B Hz
- Idea: eliminate frequencies above B Hz (must be noise)
- Analog approach: use electrical circuit (*e.g.*, treble control)
  (See EECS 215/216)
- Digital approach:
  - sample signal
  - compute spectrum using FFT
  - set to zero portions of the spectrum that are just noise
  - use inverse FFT (`ifft`) to synthesize improved signal
  - (Many more ways described in DSP course EECS 351.)

Removing noise digitally: Details

- sample using $S > 2B$, where signal is band-limited to B
- Recall that $f = \frac{k}{N} S$ and `(2/N)*abs.(fft(x))` gives:

$$[2c_0 \underbrace{c_1 \ \dots \ c_K}_{\text{low} \uparrow} \underbrace{c_{K+1} \dots c_{N/2-1}}_{\text{high} \uparrow} \underbrace{c_{N/2}}_{\text{highest}} \underbrace{\underbrace{c_{N/2-1} \dots c_{K+1}}_{\text{high} \downarrow} \underbrace{c_K \dots c_1}_{\text{low} \downarrow}}_{\text{mirror}}]$$

- Find smallest K such that $KS/N \geq B$, (*i.e.*, $K = \lceil NB/S \rceil$). Then we want:

$$[2c_0 \underbrace{c_1 \ \dots \ c_K}_{\text{low}} \underbrace{0 \dots 0}_{\text{high}} \underbrace{0}_{\text{highest}} \underbrace{\underbrace{0 \dots 0}_{\text{high}} \underbrace{c_K \dots c_1}_{\text{low}}}_{\text{mirror}}]$$

- remove frequency components above B using `fft`; synthesize "improved" signal using `ifft` (review `a[k] .= 0`)

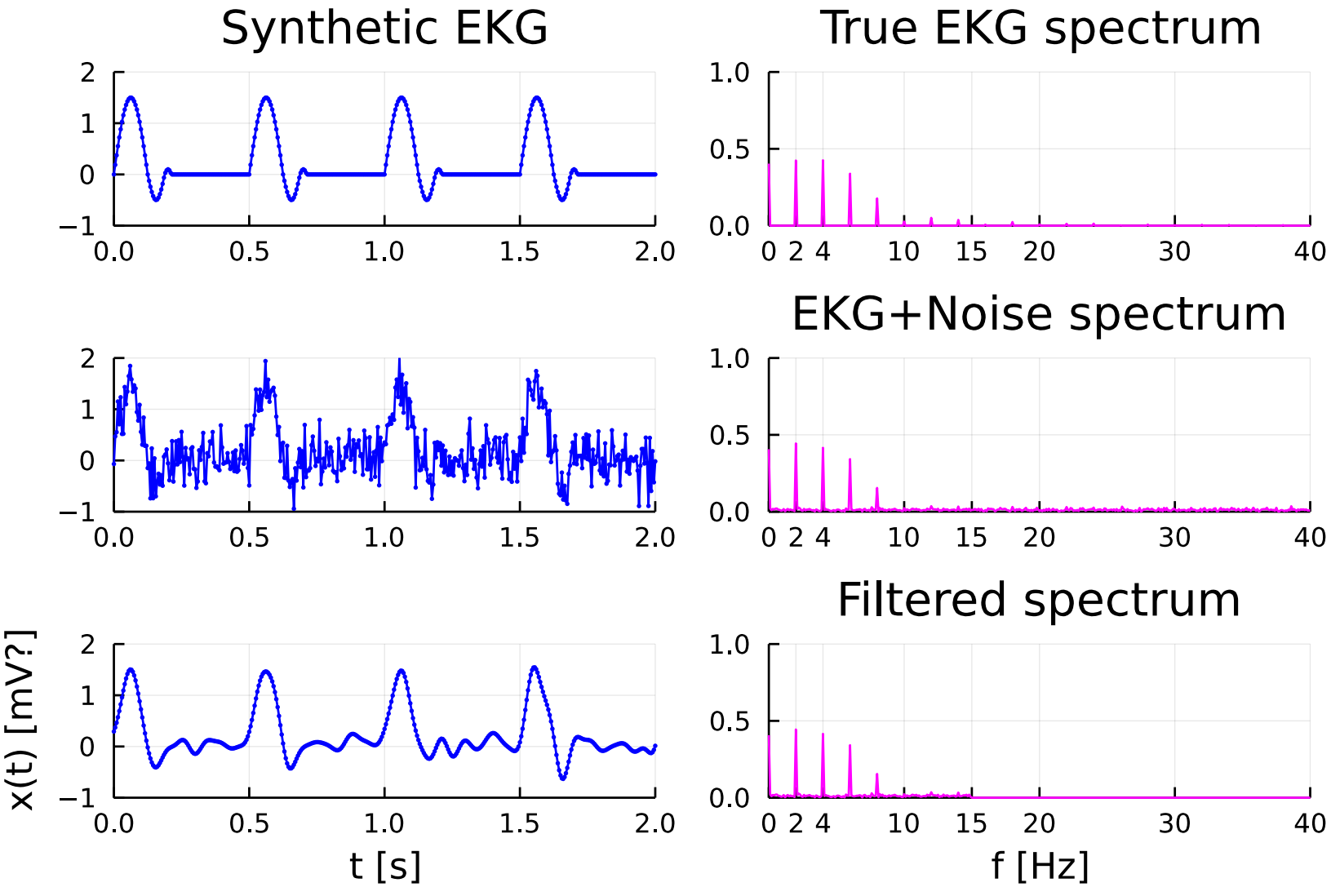
```
Yf = fft(y)
Zf = copy(Yf)
Zf[K+2:N-K] .= 0 # remember l=k+1
z = real(ifft(Zf))
```

- This is one form of signal processing operation called low pass filtering (via FFT).

Example: Low-pass filtering a noisy EKG signal

- EKG = Electrocardiogram (heart) electric signal.
- Periodic (we hope!) with period about 0.5 second when working out: (120 beats per minute = 0.5 second per beat).
- Components at $0.5, 1, 1.5, 2, \ldots$ Hz up to about 15 Hz.
- So we eliminate all frequency components above 15 Hz.
- Result: high-frequency noise eliminated, although low frequency noise components remain.
- Real-world application: wearable activity trackers like fitbit devices
- See next slide for signals and their spectra.

Example: Low-pass filtering a noisy EKG signal



Example: Code for EKG denoising

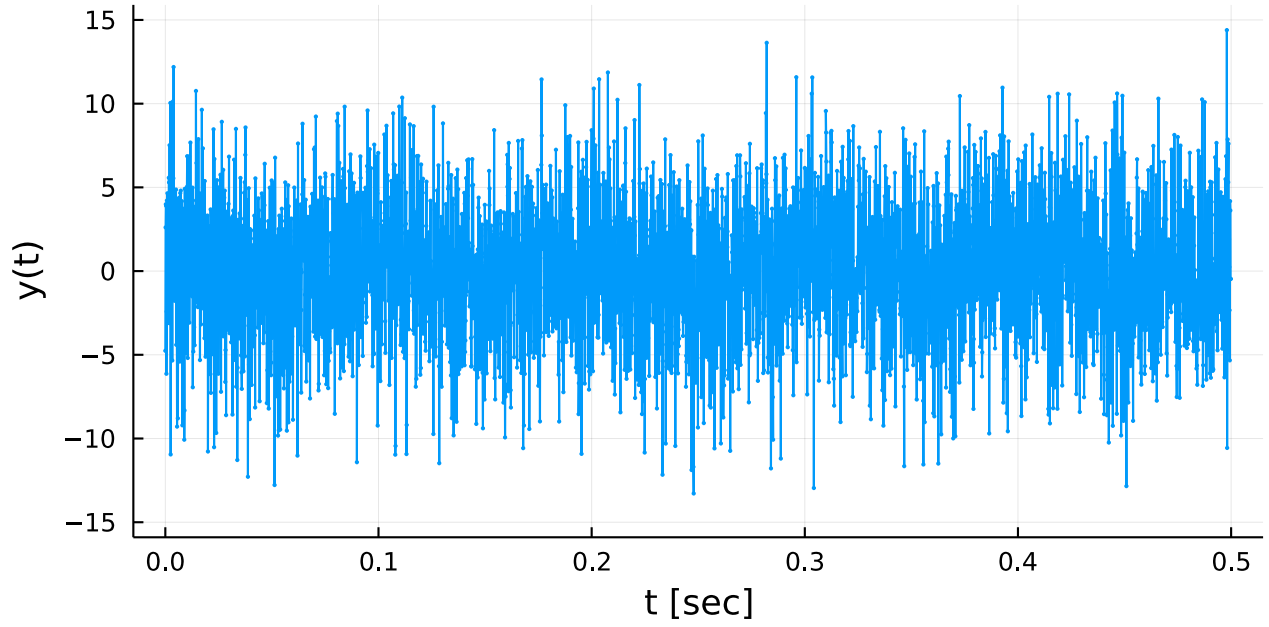
```
using Plots; default(label="", markerstrokecolor=:auto)
using FFTW: fft, ifft
using Random: seed!; seed!(0)

T = 0.5 # 120 BPM ⇒    0.5 sec per beat
ekg1(t) = 0.0 < t < 1.0 ? 1.5*sin(π*t) :
          1.0 < t < 1.5 ? -0.5*sin(2π*t) :
          1.5 < t < 1.7 ? 0.1*sin(5π*(t-1.5)) : 0
ekg2(t) = ekg1(4*mod(t,T)/T) # periodic synthetic EKG signal

def = Dict(:xlims => (0,2), :ylims => (-1,2), :color => :blue,
    :markersize => 2, :xticks => 0:0.5:2)
p1 = plot(ekg2, title="Synthetic EKG"; def...)

N = 2000; S = 200; t = (0:(N-1))/S
xtrue = ekg2.(t)
scatter!(p1, t, xtrue; def...)

y = xtrue + 0.3 * randn(N)
p2 = plot(t, y, marker=:circle; def...)

Xtrue = fft(xtrue)
Y = fft(y)
```

```
K = 150 # N*B/S for B = 15Hz cutoff
Z = copy(Y)
Z[K+2:N-K] .= 0 # remove noise
z = real(ifft(Z)) # recover signal from (complex!) spectrum
p3 = plot(t, z, marker=:circle, xlabel="t [s]", ylabel="x(t) [mV?]"; def...)

# only display 0 to 40 Hz part of spectrum
def = Dict(:xlims=>(0,40), :ylims => (0,1), :color=>:magenta,
   :xticks => [0; 2; 4; 10; 15; 20:10:40], :yticks => 0:0.5:1)
f = (0:(N-1))/N * S
p4 = plot(f, 2/N * abs.(Xtrue), title="True EKG spectrum"; def...)
p5 = plot(f, 2/N * abs.(Y), title="EKG+Noise spectrum"; def...)
p6 = plot(f, 2/N * abs.(Z), title="Filtered spectrum", xlabel="f [Hz]"; def...)

plot(p1, p4, p2, p5, p3, p6, layout=(3,2))
#savefig("denoise-ekg.pdf")
```

Key lines:

```
Z[K+2:N-K] .= 0
z = real(ifft(Z))
f = (0:(N-1))/N*S
```

Example: Removing noise from mystery signal

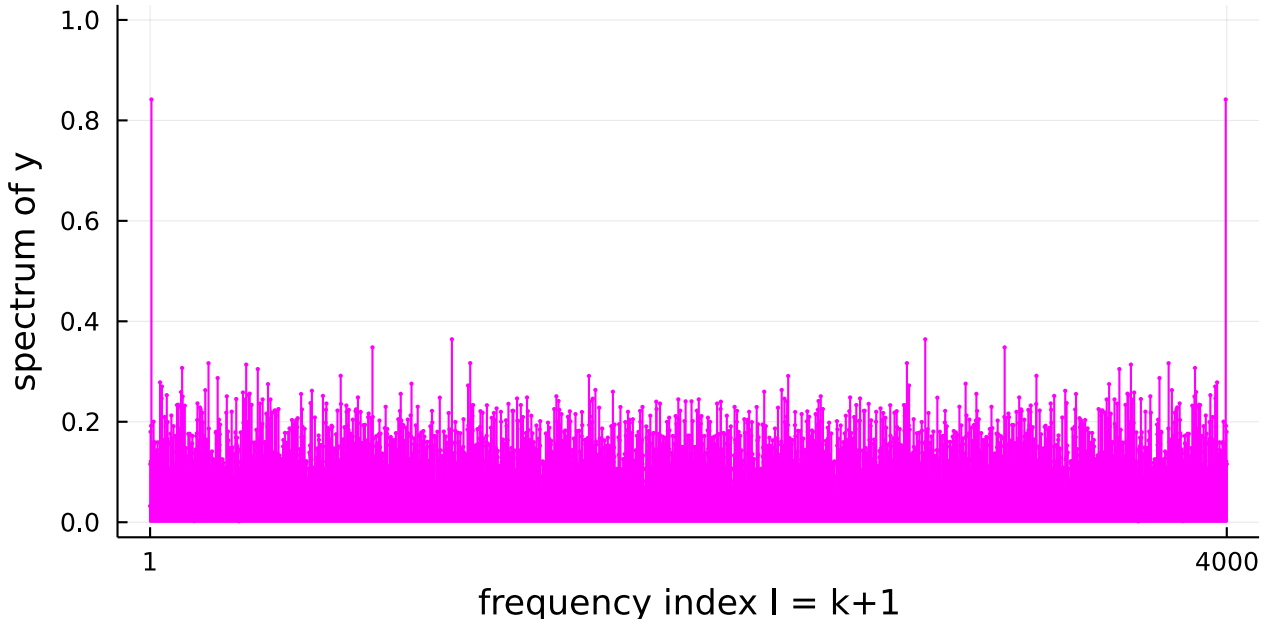


- Known: $N = 4000$ samples at $S = 8000$ Hz
- What is the signal $x(t)$ buried in this noise?
- How do we recover it from $y(t)$?
- (*cf.* restoring old analog recordings, movies, etc.)

"All static, no noise... turn the radio down. Those bandwidth signals can't reach this far."

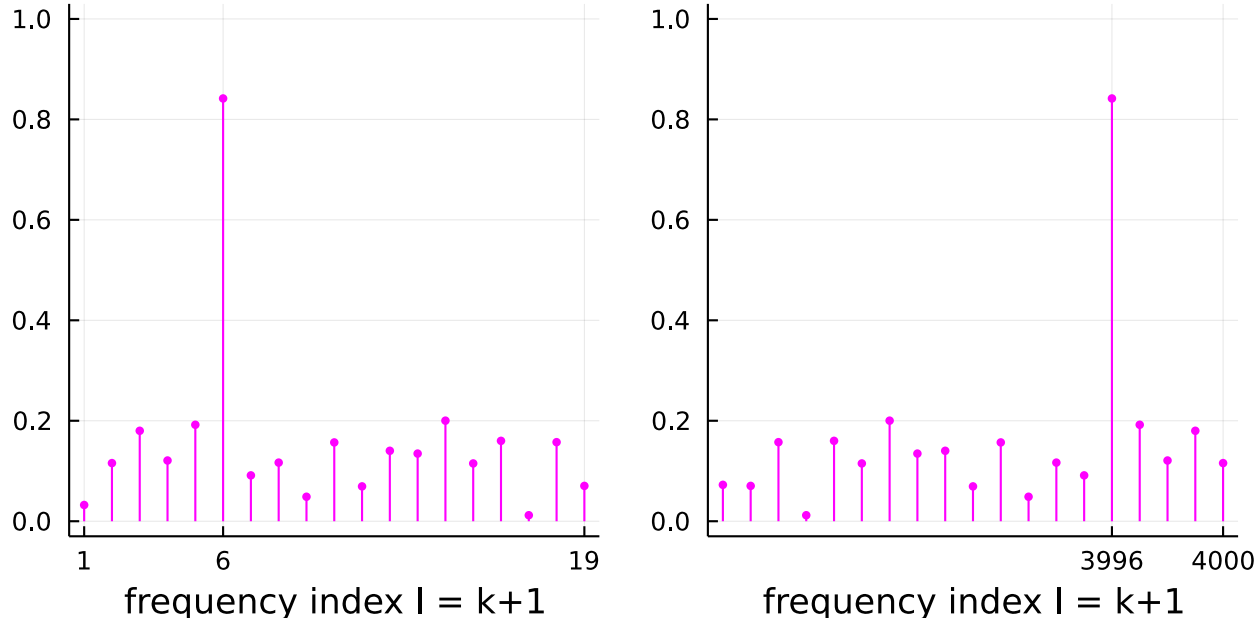
Death Cab For Cutie, song "Lowell, MA"

Spectrum of mystery signal



- `plot(2/N*abs.(fft(y)), line=:stem)`
- This spectrum also looks like random noise!?
- Look at vertical scale: large peaks near 1 and 4000.
- Zoom in to see exactly where the peaks are (next slide)

Spectrum of mystery signal: Zoomed



- Large peaks at `Julia` array index `6` and `3996` $= 4000 + 1 - (6 - 1)$
- (If we counted from 0, then peaks at $k = 5$ and $N - k = 8000 - 5 = 7995$.)
- Sinusoid frequency $(6 - 1)S/N = 5(8000)/4000 = 10$ Hz
- Now filter out all other frequencies (all but elements 6 and 3996).

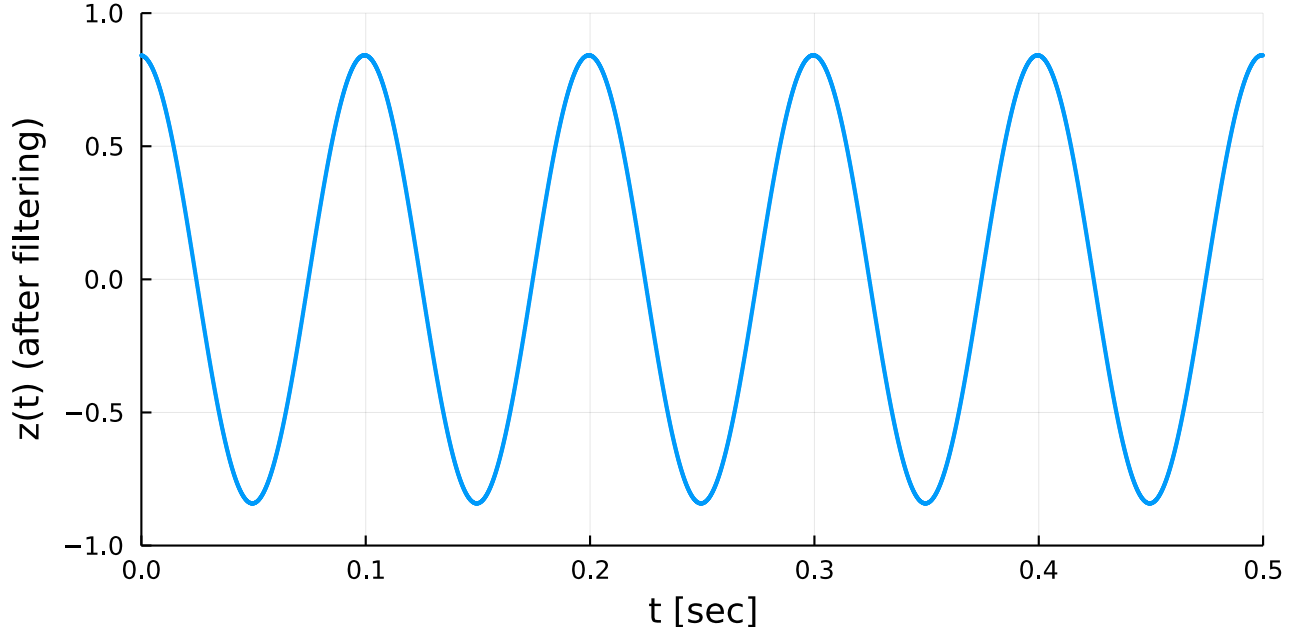
Filtering noisy mystery signal

Generating data:

```
N = 4000; S = 8000; n = 0:N-1; t = n/S
x = cos.(2π*10*t); y = x + 4 * randn(size(x))
```

Processing data:

```
using FFTW: fft, ifft
l = 6 # = k + 1
Y = fft(y); Z = copy(Y)
Z[[1:(l-1); (l+1):(N-(l-1)); (N+2-(l-1)):N]] .= 0
z = real(ifft(Z))
```



Q0.2 Did our noise removal method work perfectly?

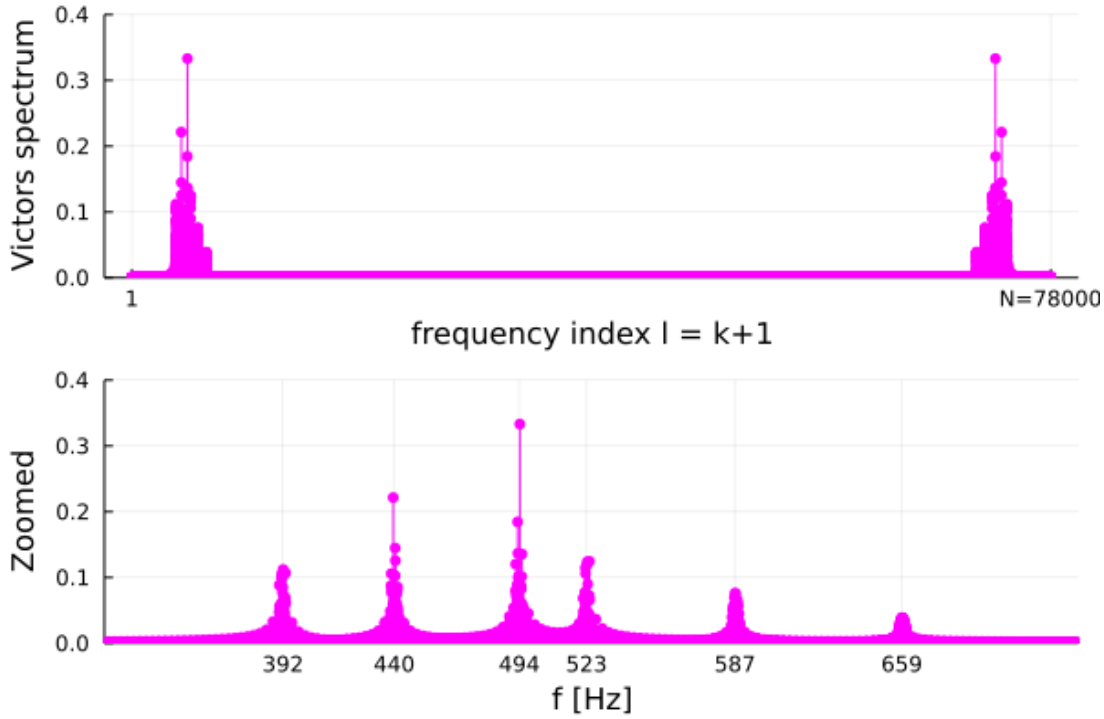
A: True B: False ??

Application: Pitch tracking (transcription) for songs Spectrogram (Lab 3)

Music transcription: Songs?

Spectrum of (sinusoid version of) "The Victors"

```
x = vec(matread("victors_tone.mat")["x"]); S = 8192; N = length(x)
plot(2/N*abs.(fft(x)), line=:stem)
```

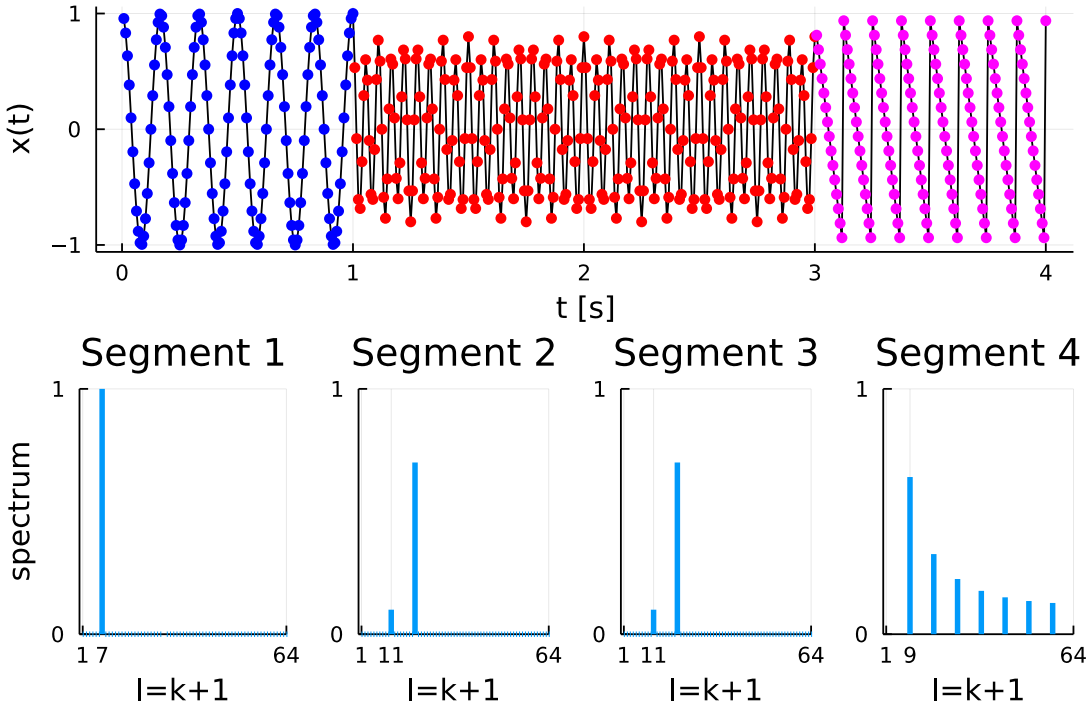


More versatile than arccos, but what note order? What song?

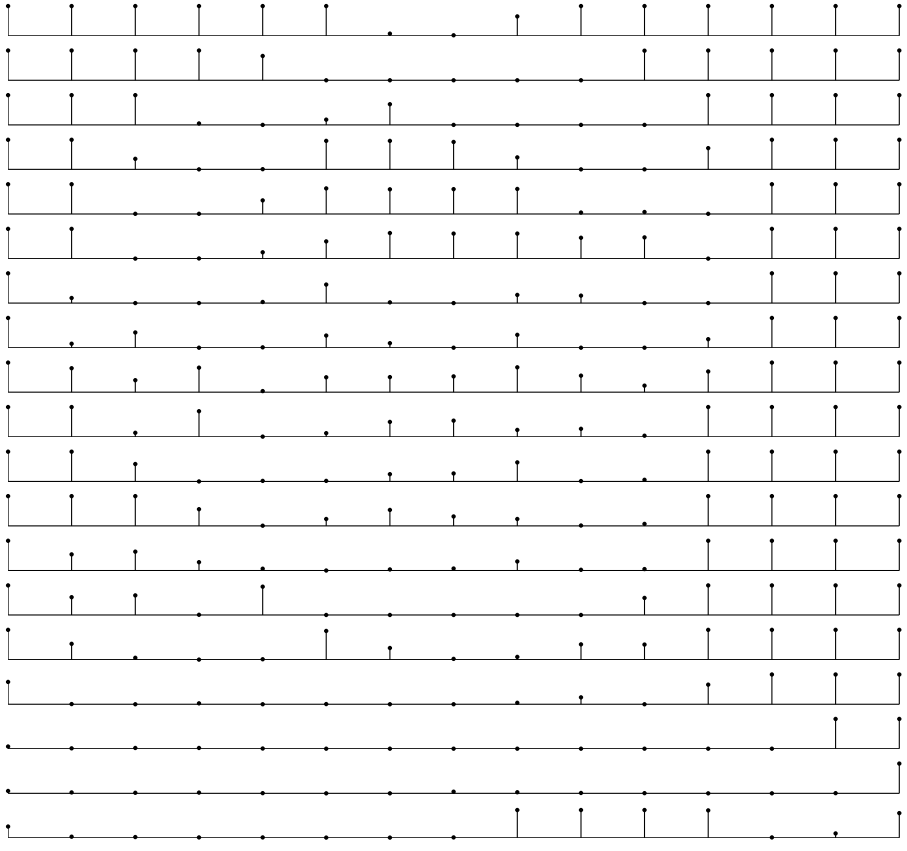
Need a *time varying* spectrum to transcribe songs!

Time-varying spectral analysis

- Idea: Subdivide long signal into short time segments.
- Hopefully notes do not change during most time segments.
- Apply `fft` to each signal segment.
- Examine spectra for each segment to identify note(s) played.
- If we make many short segments, then we get many spectra.
  How to best display many spectra?



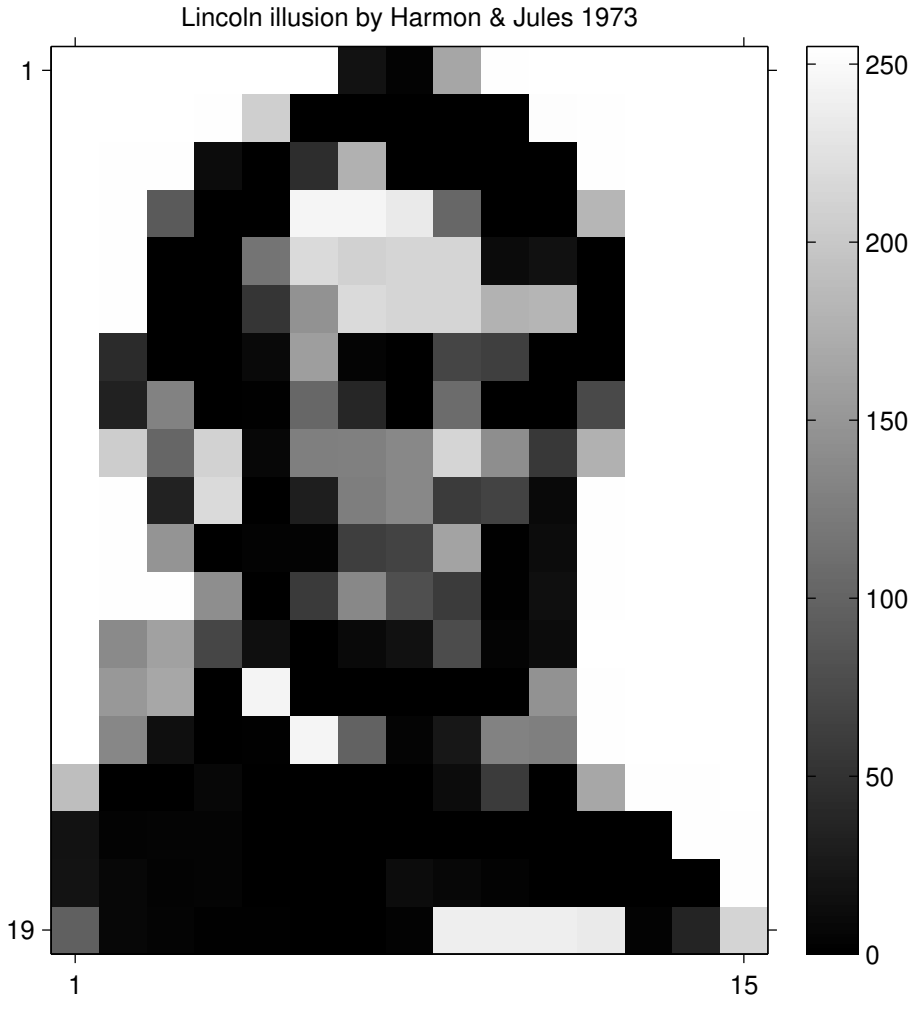
Displaying many 1D arrays: stem plots?



$\rightarrow f$ [Hz]

But a picture is worth 1000 words (or plots)...

Displaying many 1D arrays: A grayscale image



http://www.michaelbach.de/ot/fcs_mosaic

We will use grayscale (or color) images for sequences of spectra.

Spectrogram

A grayscale (or color) display of spectra of many sequential short signal segments is called a spectrogram.

-gram: from Greek root meaning something that is drawn.

- Horizontal axis is time (segment number)
- Vertical axis is frequency
- Grayscale intensity (or color) for the amplitude coefficients $\{c_k\}$

Useful for signals whose spectra vary with time, *e.g.*, songs!

Spectrogram overview

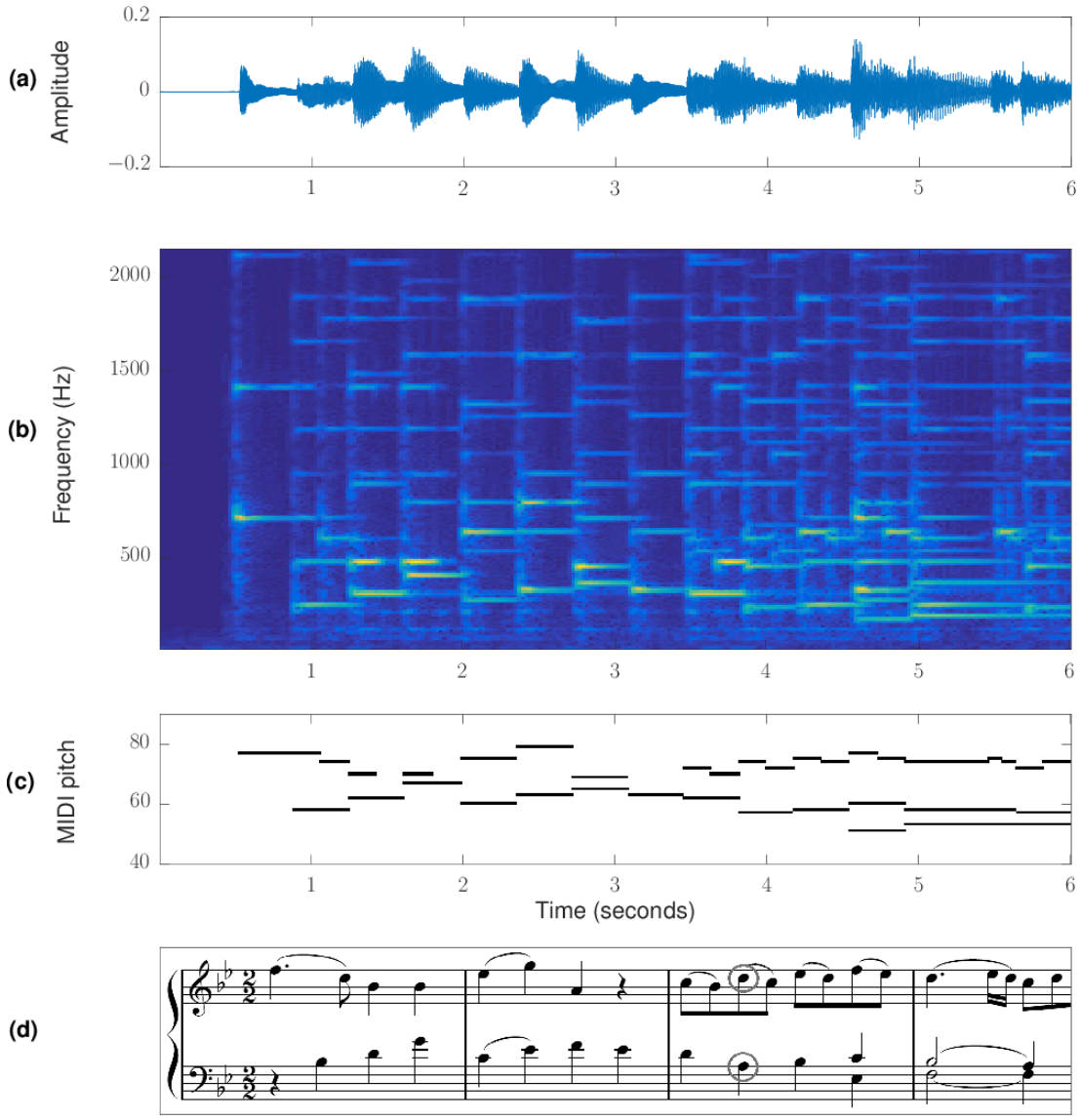


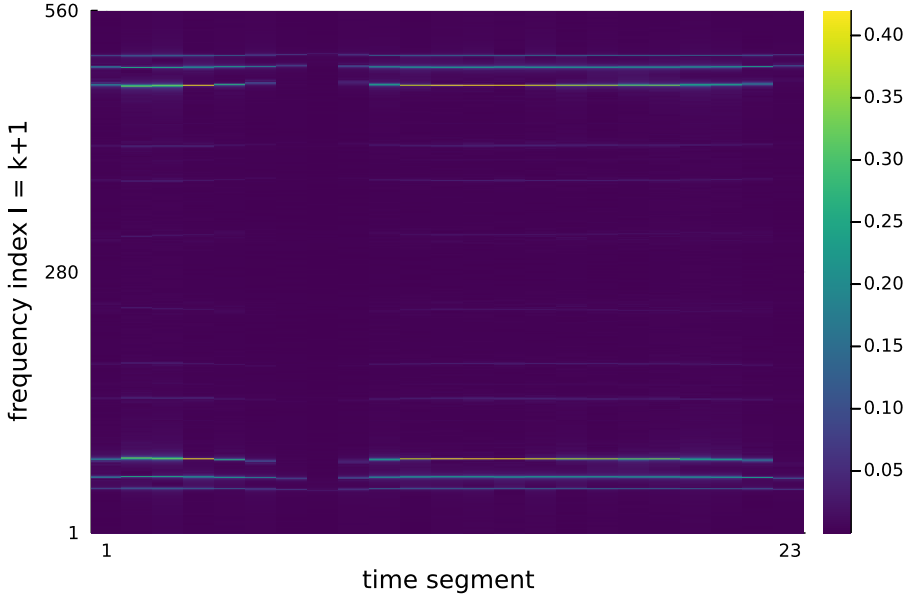
Figure from survey paper on music transcription (Spotify) [2].

(Thanks Team Banana!)

Why are first 3 lines spaced widely apart?

Spectrogram in Julia

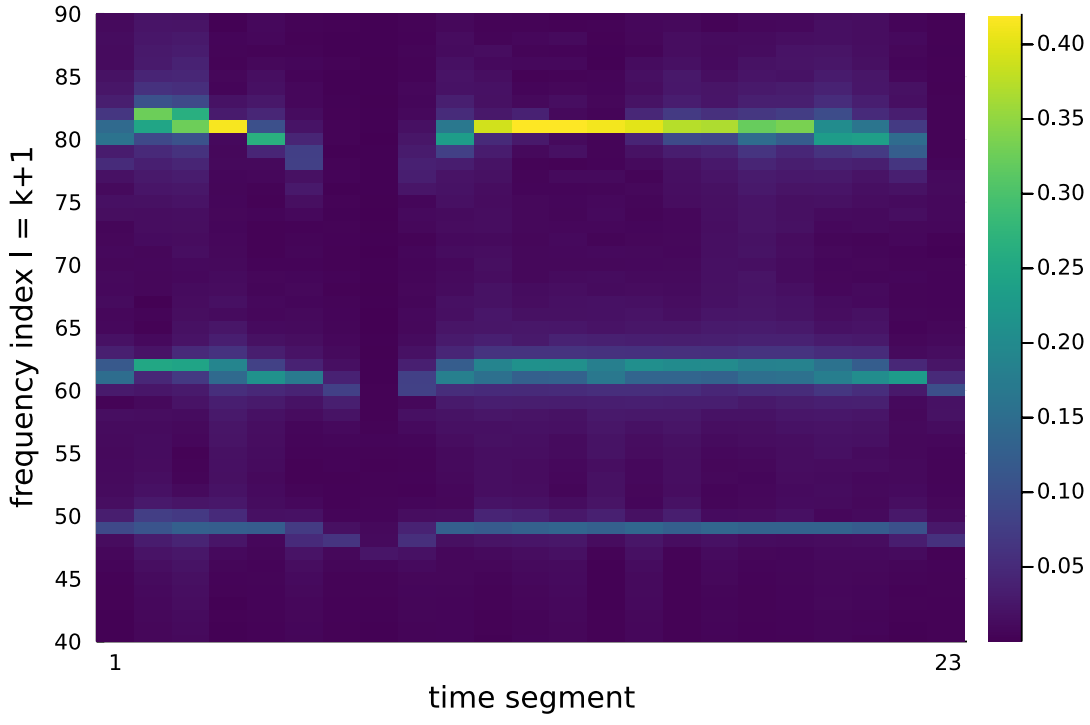
```
using WAV: wavread
using FFTW: fft
using Plots: plot!, heatmap, default; default(markerstrokecolor=:auto, label="")

(y, S) = wavread("../synth/train-whistle.wav") # 12880 = 560 × 23
M = 560; z = reshape(y, M, :) # 23 sound segments of 560 samples each
Z = 2/M * fft(z, 1) # 1D fft of each column of z
p1 = heatmap(abs.(Z), color=:cividis, # spectrogram
    xticks = [1, 23], xlabel="time segment",
    yticks = [1, M/2, M], ylabel="frequency index l = k+1")
```



Spectrogram of train whistle

Make spectrogram look better by zooming in:

```
plot!(ylims = (40,90), yticks=40:5:90)
```



play

Q0.3 What is pitch of lowest note of chord (in Hz, round to nearest integer)?

??

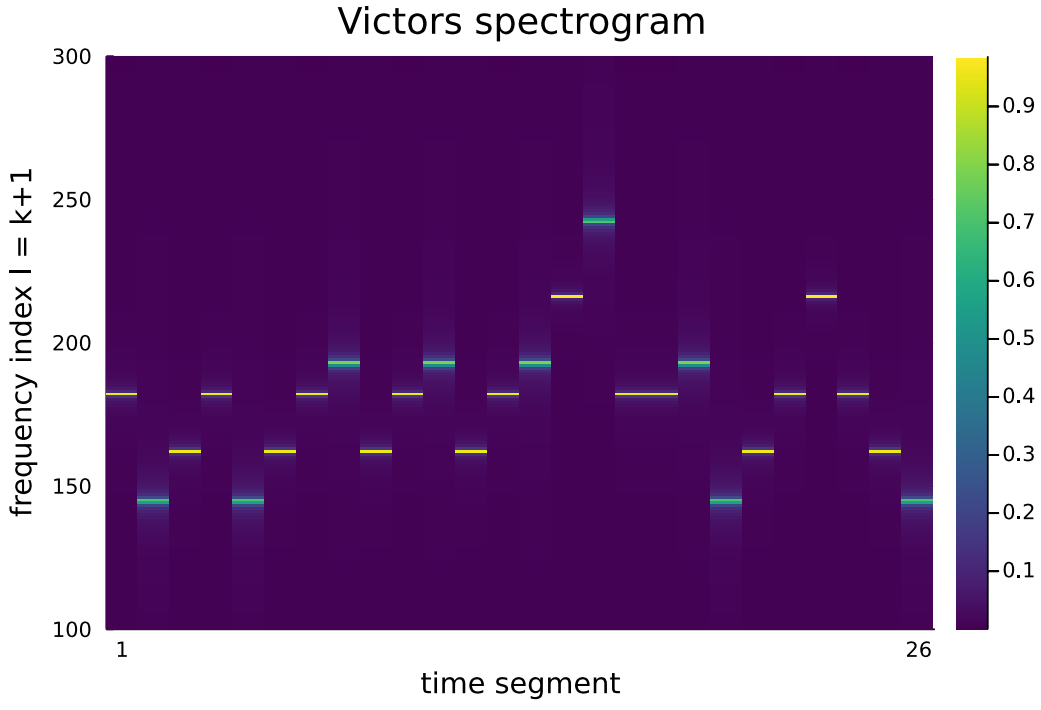
Spectrogram of a song

`victors_tone.mat` contains the signal of a 26 note song (Lab 2).
Each note has 3000 samples for sampling rate $S = 8192 \frac{\text{Sample}}{\text{Second}}$.

Here is Julia code for displaying its spectrogram:

```
using MAT: matread
using Plots: heatmap
using FFTW: fft

file = "../../../data/victors_tone.mat"
x = vec(matread(file)["x"])
S = 8192 # sampling rate
N = 3000 # samples per note
z = reshape(x, N, :) # 26 notes of 3000 samples each
Z = 2/N*abs.(fft(z, 1)) # 1D FFT of each column of z
p1 = heatmap(Z, color = :cividis,
    ylabel = "frequency index l = k+1",
    ylims = (100, 300), title = "Victors spectrogram",
    xticks = [1,26], xlabel = "time segment")
#savefig("victors1.pdf")
```

Spectrogram of "The Victors"



play

http://haostaff.com/store/images/pianoroll.gif

Each column of this spectrogram is one spectrum shown in gray scale.

What kind of signal is each note? ??

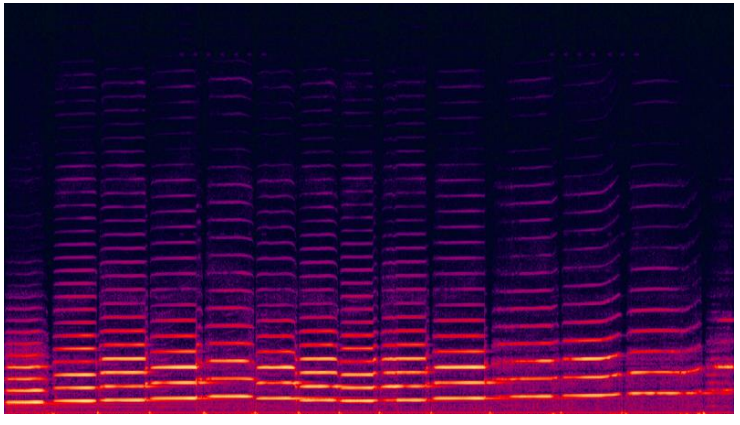
This representation (time horizontal, frequency vertical) is a lot like music!

Frequency of highest tone? $f = \frac{k}{N} S = \frac{242}{3000} 8192\text{Hz} = 660.8\text{Hz}$ (E)

Real music spectrograms

Bizarre "hidden" spectrogram from song by Aphex Twin [wiki]

Violin playing: [wiki]



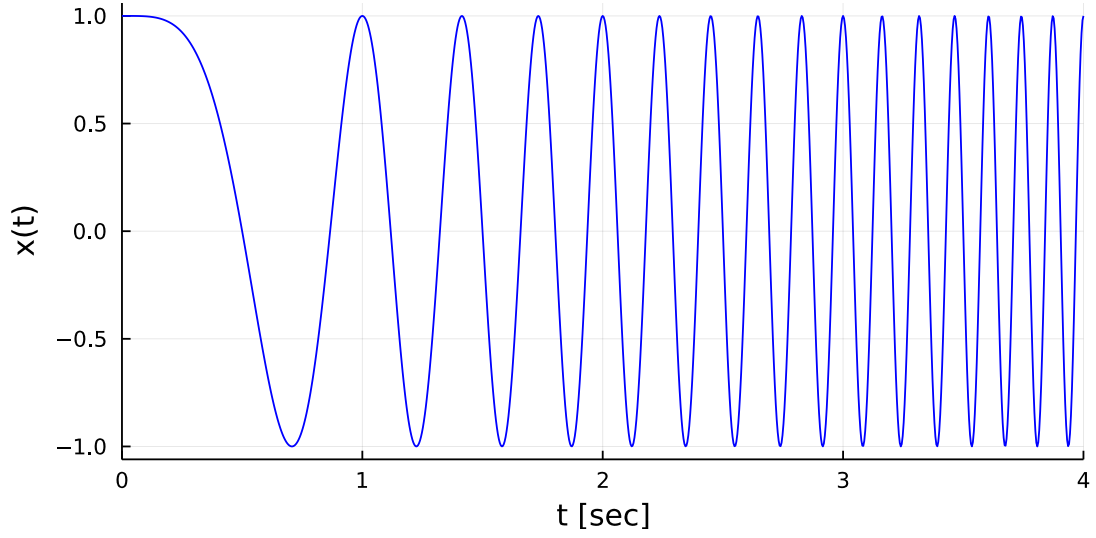
play

Example: Spectrogram of chirp signal

Chirp signal: $x(t) = \cos\left(2\pi\alpha t^2\right)$

- birds, dolphins, some forms of RADAR, slide trombone?
- Instantaneous frequency increases linearly with time
- Instantaneous frequency $= 2\alpha t$ (not αt) Hz.

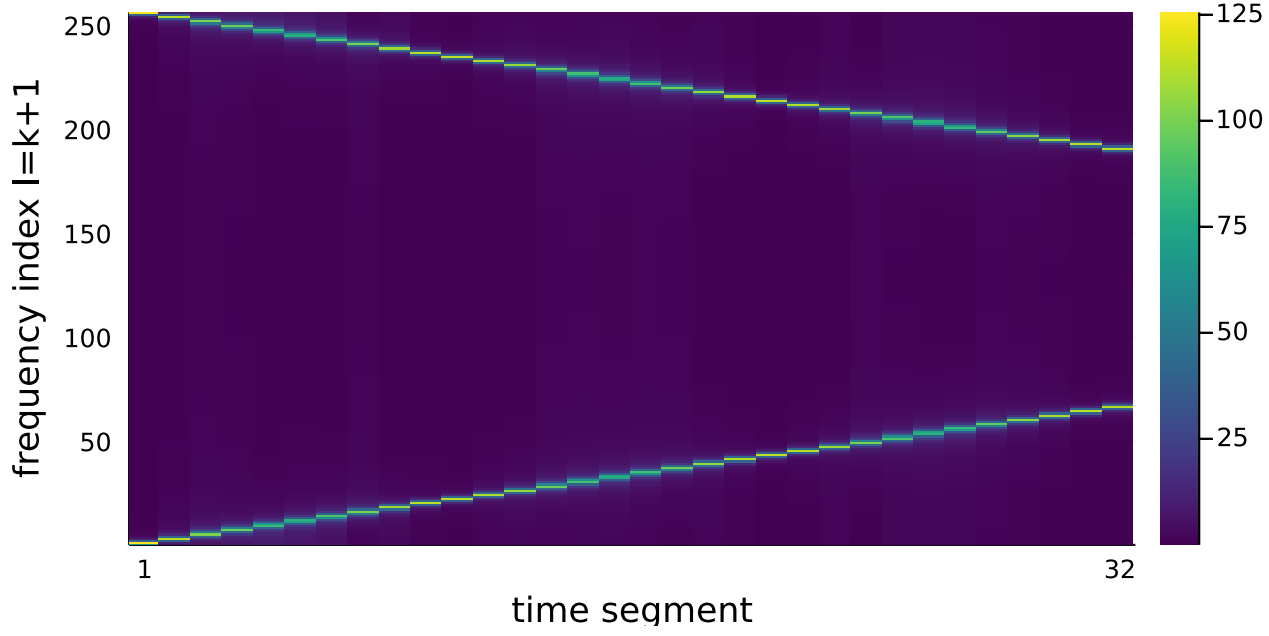
```
t = (0:8191)/250; x = cos.(2π*t.^2); plot(t, x)
```



Instantaneous frequency increases from 0 to $2 * 8191/250 = 65.5$ Hz

Example: Spectrogram of chirp signal

```
heatmap(abs.(fft(reshape(x,256,32), 1)), color=:cividis)
```



Final segment has frequency index `l = 68`.

Corresponding frequency $f = \frac{k}{N} S = \frac{68-1}{256} 250 = 65.4$ Hz.

This is around the *average* of frequencies in the final time segment.

Sound of a higher-frequency chirp: play

Black holes and chirps

The gravitational waves that were detected in early 2016 as a signature of two black holes colliding billions of years ago were detected because the black holes emitted chirps!

To hear the sound, see:
https://www.youtube.com/watch?v=TWqhUANNFXw

For more details, see:
http://www.ligo.org/science/GW-Inspiral.php

Summary of spectra/spectrogram

- With spectrum, we can determine frequency components of one or more notes played *simultaneously*
- Need $S > 2B$
- Can remove or reduce some forms of noise / interference
  - use `fft` to *analyze* spectrum
  - can modify spectrum
    - Example: set some values to zero
    - Example: shift some values to other frequencies (*e.g.*, auto-tune)
  - use `ifft` to *synthesize* modified signal (*e.g.*, signal with reduced noise or interference)
- Spectrogram lets us analyze sequences of notes (songs)

Sampling rates and maximum frequency

Q0.4 What is the highest frequency we could find by arccos method? (HW1 "challenge" problem.)

??

Q0.5 What is the highest frequency we can find by the FFT method?

`plot((2/N)*abs.(fft(x)))` gives:

$$[2c_0 \underbrace{c_1\ c_2\ \ldots\ c_{N/2-2}\ c_{N/2-1}}\ c_{N/2}\ \underbrace{c_{N/2-1}\ c_{N/2-2}\ \ldots\ c_2\ c_1}]$$

??

Q0.6 What is the maximum frequency we can find "by eye" from a digital signal $x[n]$, assuming no aliasing has occurred?

??

Q0.7 What is the maximum frequency we can find "by eye" from an analog periodic signal $x(t) = x(t+T)$?

??

Why $S > 2B$ is crucial to avoid aliasing

- Consider $x(t) = \cos(2\pi f t)$ with $f = S/2$
  Plot its samples $x[n]$
  ??

- Consider $y(t) = \sin(2\pi f t)$ with $f = S/2$
  Plot its samples $y[n]$
  ??

Q0.8 Would $S \geq 2B$ suffice to avoid aliasing?
??

- For FFT approach, the highest *reliable* frequency is really for $k = \frac{N}{2} - 1$,
  i.e., $f = \frac{(N/2-1)}{N} S = \left(\frac{1}{2} - \frac{1}{N}\right) S < S/2$

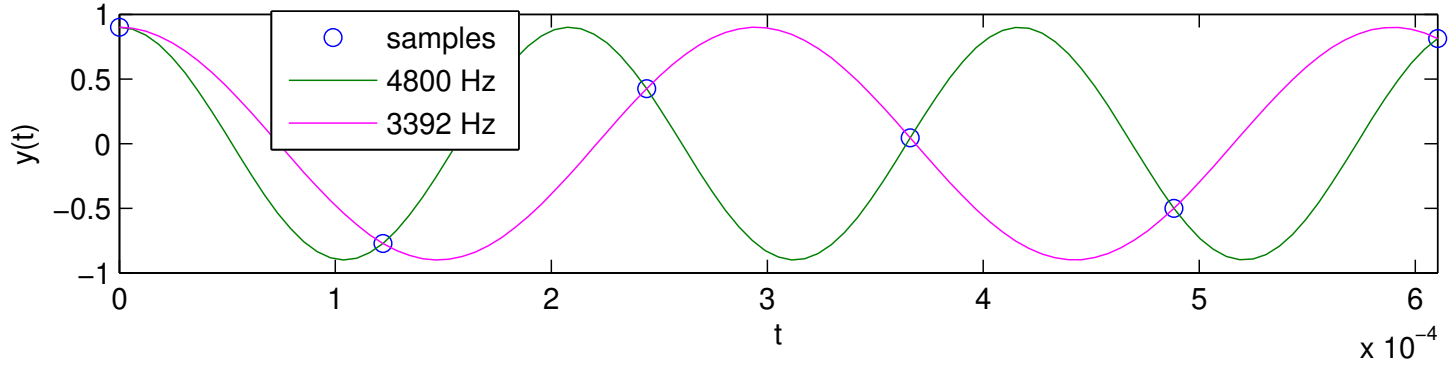
Aliasing: audio example

```
S = 8192; t = 0:1/S:0.3
x = 0.9*[cos.(2pi*2800*t); cos.(2pi*3800*t)]
```

play

```
y = 0.9*[cos.(2pi*3800*t); cos.(2pi*4800*t)]
```

play



arccos method says 3392 Hz, not 4800 Hz for last part of this example

Q0.9 Is $S > 2B$ here?

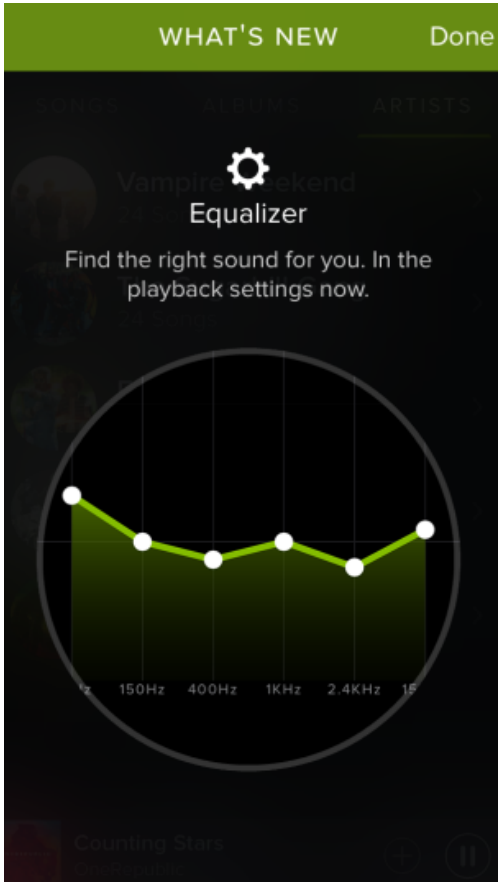
A: Yes B: No ??

Audio spectrum equalizers: Analog and digital

Audio equalization

http://commons.wikimedia.org/wiki/File:Graphic_equalizer.jpg

spotify equalizer (2014)



Another possible component of a music signal processing project.

References

[1] E. Upton, M. Veloso, A. Gadgil, T. White, B. Monks, and R. Malkin. Today's Engineering Heroes: The Rain Forest's Defender. *IEEE Spectrum*, 52(3):46–7, March 2015.

[2] E. Benetos, S. Dixon, Z. Duan, and S. Ewert. Automatic music transcription: an overview. *IEEE Sig. Proc. Mag.*, 36(1):20–30, January 2019.