Eng. 100: Music Signal Processing
DSP Lecture 11
DSP topics / P3 help

Curiosity:

http://www.wired.com/2014/08/gyroscope-listening-hack

https://www.youtube.com/watch?v=hSCObIXDCJc (8-bit synth of So What)

Announcements:

- Course evaluations: submit receipt to Gradescope

Outline

- Part 1. SNR measurements
- Part 2. CDR feedback
- Part 3. DSP courses
- Part 4. Pitch and tempo shifting
- Part 5. Auto-tune
- Part 6. P3 help

Part 1. SNR measurement

SNR measurement

Review of Additive White Gaussian Noise (AWGN):

$$\underbrace{y(t)}_{\text{measured signal}} = \underbrace{x(t)}_{\text{ideal signal}} + \underbrace{\varepsilon(t)}_{\text{additive noise}}$$

$$\text{SNR} = 10\log_{10}\left(\frac{\sum_{n=1}^N x_n^2}{\sum_{n=1}^N \varepsilon_n^2}\right) = 10\log_{10}\left(\frac{\frac{1}{N}\sum_{n=1}^N x_n^2}{\frac{1}{N}\sum_{n=1}^N \varepsilon_n^2}\right)$$

Peak SNR (PSNR)

$$\textbf{PSNR} = 10\log_{10}\left(\frac{x_{\max}^2}{\frac{1}{N}\sum_{n=1}^N \varepsilon_n^2}\right), \quad x_{\max} = \max_n |x_n|.$$

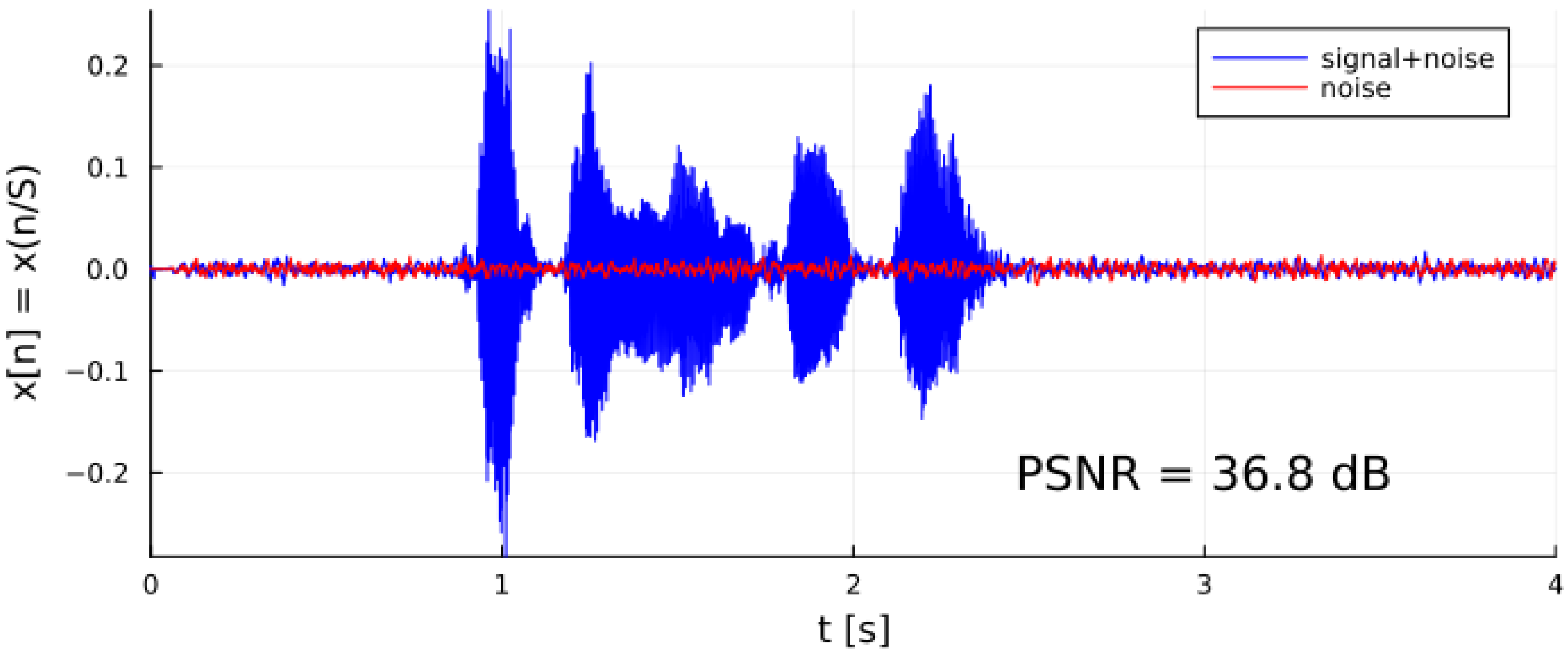
In `Julia`:

```
PSNR = 10*log10(maximum(abs2, x) / (sum(abs2, noise) / length(noise)))
```

SNR measurement example

Practical PSNR formula uses the peak of the noisy signal:

$$\mathrm{PSNR} = 10 \log_{10}\left(\frac{y_{\max}^2}{\frac{1}{N}\sum_{n=1}^N \varepsilon_n^2}\right), \quad y_{\max} = \max_n |y_n|.$$



SNR measurement code

```
using Sound: record, sound
using FFTW: fft, ifft
using Plots: plot, plot!, default, gui, savefig
using WAV: wavwrite
using Measures: mm
default(label="", markerstrokecolor=:auto, widen=false)

if !@isdefined(x1) # || true
    record(0.001) # warm-up
    @info "noise only"
    x0, S = record(4)
    @info "signal+noise"
    x1, S = record(4)
end

Nx = length(x1)
t = (1:Nx)/S
p0 = plot(t, x1, label = "signal+noise", color = :blue,
 xaxis = ("t [s]", (0,4), 0:4), yaxis="x[n] = x(n/S)",
 size = (700,300), left_margin = 3mm, bottom_margin = 4mm,
)
plot!(t, x0, label="noise", color = :red)

psnr = 10 * log10(maximum(abs2, x1) / (sum(abs2, x0) / length(x0)))
psnr = round(psnr, digits=1)
plot!(annotate = (3, -0.2, "PSNR = $psnr dB"))
# savefig(p0, "snr1.png")
# wavwrite(x0 / maximum(abs, x1), "snr1-noise.wav"; Fs=S)
# wavwrite(x1 / maximum(abs, x1), "snr1-signal.wav"; Fs=S)
```

Pythagorean theorem version

$$\underbrace{y}_{\text{measured signal}} = \underbrace{x}_{\text{ideal signal}} + \underbrace{\varepsilon}_{\text{additive noise}}$$

If $x \perp \varepsilon$ then by Pythagorean theorem:
$\|y\|^2 = \|x\|^2 + \|\varepsilon\|^2$ so $\|x\|^2 = \|y\|^2 - \|\varepsilon\|^2$

$$\Longrightarrow \text{SNR} = 10\log_{10}\left(\frac{\sum_{n=1}^N x_n^2}{\sum_{n=1}^N \varepsilon_n^2}\right) = 10\log_{10}\left(\frac{\left(\sum_{n=1}^N y_n^2\right) - \left(\sum_{n=1}^N \varepsilon_n^2\right)}{\sum_{n=1}^N \varepsilon_n^2}\right)$$

```julia
using Random: seed!; seed!(0)
S = 44100
x = cos.(2π*(1:S)/S*528) # ideal
e = 0.1 * randn(S) # noise
y = x + e # noisy measurement
osnr = 10 * log10(sum(abs2, x) / sum(abs2, e)) # original SNR
psnr = 10 * log10(maximum(abs2, y) / (sum(abs2, e) / length(y))) # peak SNR
pythagorean = 10 * log10((sum(abs2, y) - sum(abs2, e)) / sum(abs2, e))
[osnr psnr pythagorean] # [ 16.9889  22.5587  16.9829 ]
```

Part 2. CDR feedback

CDR DSP notes

- Basic requirement of a synth: produce tones of correct pitch and duration (easily testable).
- Accuracy of "duration" in msec or in note lengths?
- Non-specific music terminology
  - "input song" is too vague; signal from .wav file?
    song name (string)? MIDI file (pitches and durations)?
  - "tone" is ambiguous (pitch or timbre?)
- More detail about DSP in final report.
  - E.g., if you used "envelope" explain how it works and what parameters used and show an example or two of the envelope and the identified note segments.
  - If you used "correlation" describe the reference signals
  - If you used "autocorrelation" or "FFT spectrum" describe process for finding the appropriate peak.
  - If it does not work 100% of the time, show an failed example and discuss how you might address it in the future.

- Technical writing style tips
  - avoid contractions in technical writing (*don't* use them!)
  - punctuate equations as part of sentences (not isolated)
  - Always a comma after 'e.g.' and 'i.e.' (see below)
- English: fewer vs less
  An easier class has
  - *fewer* homework problems (countable)
  - *less* time spent per problem (not countable)

  (Many exceptions, e.g., "500 words or less" in an essay...)
- Grammar
  - The student aced the test, which was awesome. (Correct)
  - The student aced the test that was awesome. (Correct)
  - The student aced the test, which was too long. (Incorrect)

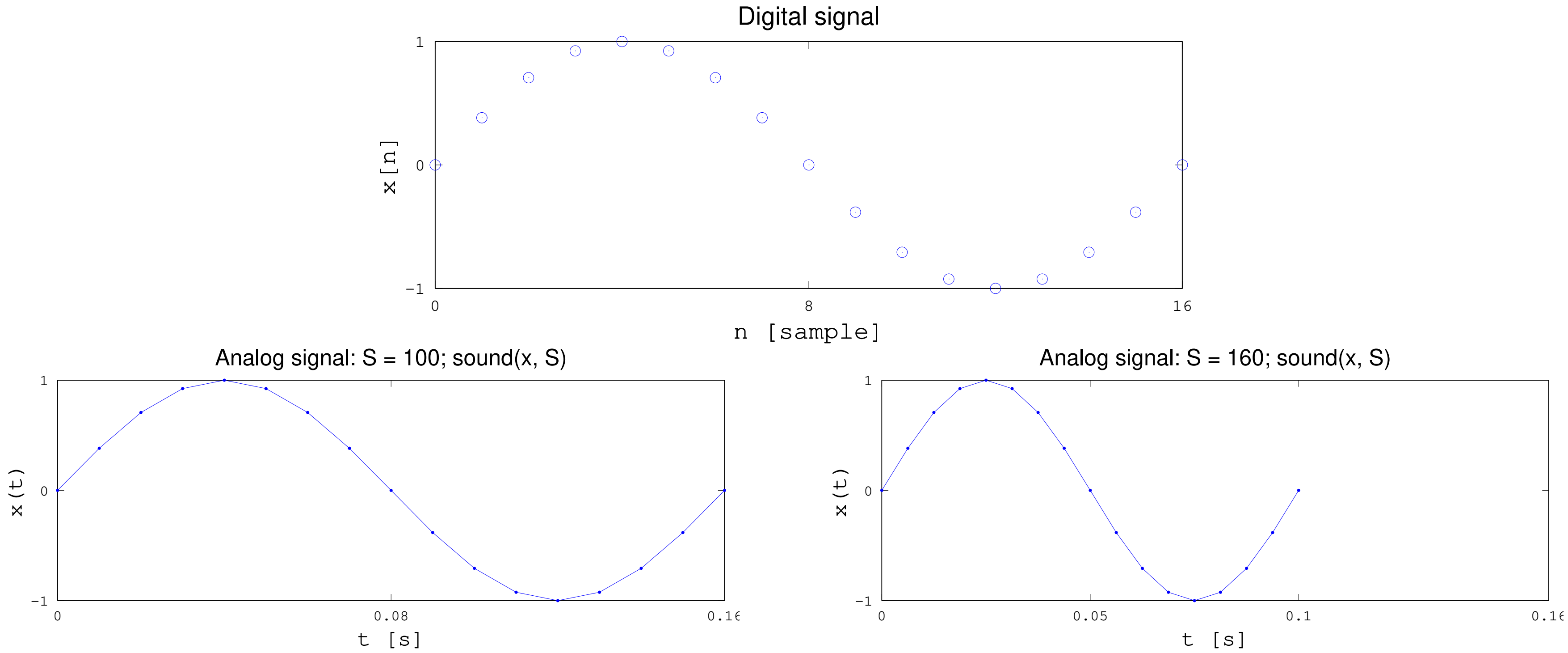
Part 3. DSP courses

Signal Processing Courses

- EECS 216: Introduction to Signals and Systems
- EECS 351: Digital Signal Processing and Analysis
- EECS 452: Digital Signal Processing Design Laboratory
- EECS 453: Principles of Machine Learning
- Linear algebra: Math 214, 217, ...

Part 4. Pitch and tempo shifting

Time scaling via sampling rate



Changing the sampling rate parameter: changes pitch *and* tempo.

`sound(x,S)`

play

`sound(x, S * 2^(6/12))`

play

Tempo changes

How to play a recorded song faster or slower *without changing pitch*?

Phase vocoder [1] url [2] url [3] url [4] url [5] url url [6]

Basic idea:

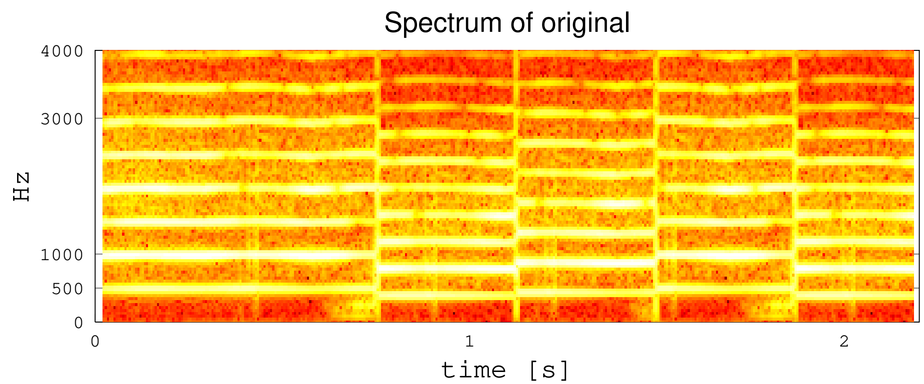
- Use "short-time Fourier transform" (STFT) to make spectrogram (*i.e.*, FFT of overlapping segments)
- Modify spectrogram using interpolation, being careful with phase
- Synthesize signal from modified spectrogram using inverse STFT (Inverse FFT via `ifft` of each segment, carefully combining.)

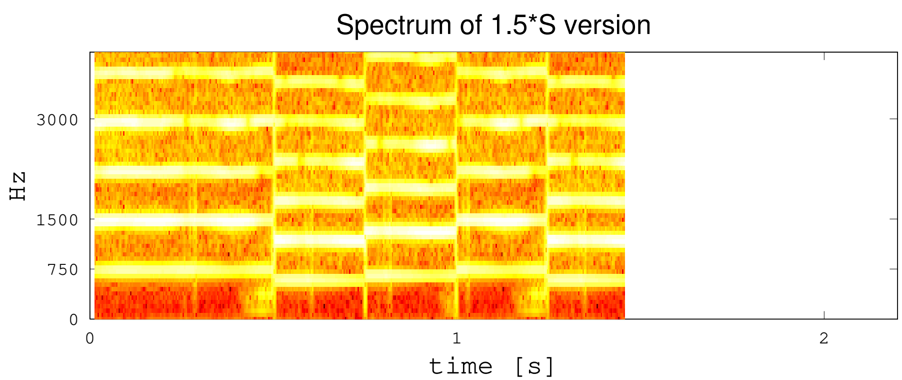
Example.

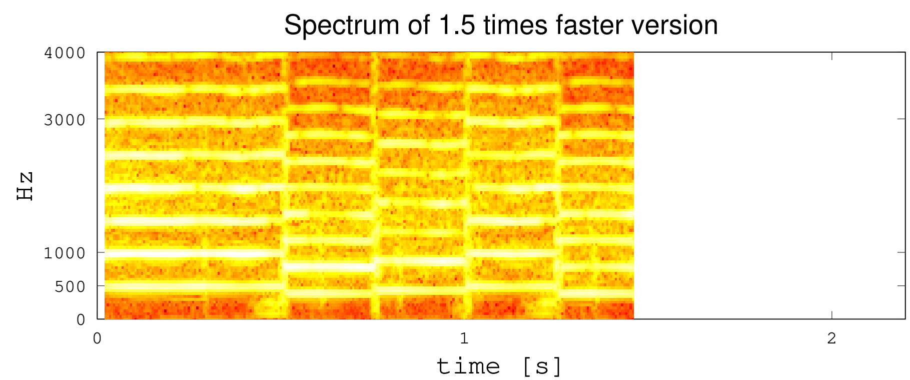
original: play 3/4 speed: play 3/2 speed: play

Note: can combine phase vocoder with sampling rate parameter change

Spectrograms







Using `sound(x, 1.5*S)` Using phase vocoder

original: play

play play

Song duration

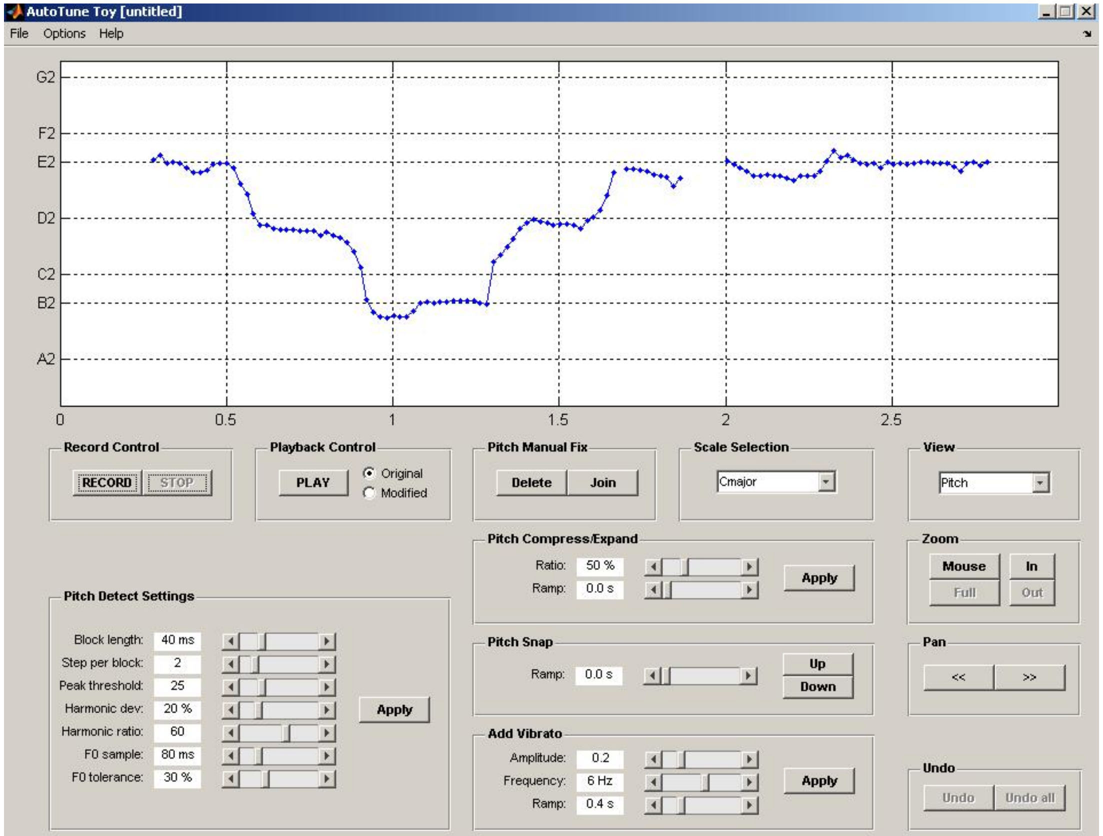
```julia
using Sound: soundsc
x = 1000*cos.(2π * (1:2000)/8000 * (200:10:290)')
soundsc(vec(x), 4000)
```

Q0.1 What is the duration (in seconds) of the song generated by this code?

$\boxed{??}$

Part 5. Auto-tune

Auto-tune demo



before: play

after: play

The auto-tune method was popularized by Cher (!) in 1998 hit "Believe"

`http://www.mathworks.com/matlabcentral/fileexchange/26337-autotune-toy`

`tex/course/100-engin/demo/auto-tune-toy/AutoTuneToy.m`

2015 example: `https://www.youtube.com/watch?v=eq1FIvUHtt0`

Part 6. P3 help

References

[1] J. L. Flanagan and R. M. Golden. Phase vocoder. *Bell Syst. Tech. J.*, 45(9):1493–509, November 1966.

[2] M. Portnoff. Implementation of the digital phase vocoder using the fast Fourier transform. *IEEE Trans. Acoust. Sp. Sig. Proc.*, 24(3):243–8, June 1976.

[3] M. Dolson. The phase vocoder: A tutorial. *Computer Music Journal*, 10(4):14–27, 1986.

[4] J. Laroche and M. Dolson. New phase-vocoder techniques for pitch-shifting, harmonizing and other exotic effects. In *IEEE Workshop on Appl. of Signal Processing to Audio and Acoustics*, pages 91–4, 1999.

[5] D. P. W. Ellis. A phase vocoder in Matlab, 2002.

[6] W. A. Sethares. *Rhythm and transforms*. Springer, 2007.