Eng. 100: Music Signal Processing
DSP Lecture 7
Project 2: Touch-tone synthesizer / transcriber

Curiosity: http://supermegaultragroovy.com/products/capo/mac

Announcements:
• Read Project 2 before lab this week! (Last set of reading questions.)
• Lab 3 due this week (upload pdf to Canvas)
• HW 4 on Canvas (Matlab), due Thu. Nov. 6 in class
• Exam 1 DSP solutions on Canvas. DSP regrade requests by email memo to JF.
  (Discuss answers at end of class if time permits.)
• Midterm course evaluations (11/22)
F15 Exam 1: scores on DSP part

22 students. median = 46.2/50, median = 47.5/50, std. dev. = 4.0

A 50 50 50 50 50 50 50 48 48 48 47 46 45 45
B 44 44 42 41 41 41
C 36

DSP portion of Exam 1 is 50 / 1302 = 3.8% of the course grade

Homework / reading question scores:
per syllabus: 25+60+30+65 + 8+4+4+4+2 = 202, out of 1302 = 15.5%
Previous year: 90/1200 = 7.5% (educational...)
Proposal: \[
\max \left( \frac{\text{hw} + \text{other}}{1302} \times 100\%, \frac{\text{hw}}{202} \times 7.5\% + \frac{\text{other}}{1100} \times 92.5\% \right)
\]
Outline

• Part 0. Spectrogram examples

• Project 2: Touch-tone phone signals
  ○ Part 1. Analyze spectra of touch-tone phone signals
  ○ Part 2. Synthesize (in Matlab) a touch-tone keypad
  ○ Part 3. Transcribe touch-tone signals to phone number
  ○ Part 4. Analyze transcriber performance for noisy signals
Part 0. Spectra and Spectrogram examples
Violin spectrogram

Recall
- horizontal axis: time segment
- vertical axis: frequency (Hz)
- color intensity: amplitude
Nature’s spectrum analyzer (Audio)

Different sound frequencies are sensed at different positions along the cochlea!

http://www.ifd.mavt.ethz.ch/research/group_lk/projects/cochlear_mechanics

(from Encyclopaedia Britannica Inc.)
Nature’s spectrum analyzer (Light)

A glass prism separates light into individual wavelengths (electromagnetic waves with different frequencies)

FFT separates sounds into individual (audio) frequencies
- White light contains “all” visible wavelengths (frequencies)
- Q: What audio signals contain “all” audio frequencies?
  A: ?? - later in this lecture
Spectrogram application: Spirometer

Low-cost spirometer designed by WUSTL students (in biomedical engineering and mechanical engineering) using microphone and spectrogram to determine air flow.

Note: “Short-Time Fourier Transform” = spectrogram

Why “short time?”

Overview of Project 2 / Outline

Four parts:

● Part 1. Analyze spectra of touch-tone phone signals.
  ○ Reverse engineer tone frequencies
  ○ What method will you use? ??
  ○ Determine pattern of frequencies for touch-tone keys

● Part 2. Synthesize (in Matlab) a touch-tone keypad.
  ○ Straightforward GUI: similar to Project 1 keyboard.

● Part 3. Transcribe touch-tone signals to phone number.
  ○ Can look for specific frequencies; do not need fft.
  ○ New method for detecting frequency components: correlator

● Part 4. Analyze your transcriber performance for signals that are degraded by white noise

This project is a “prelude” for a typical Project 3 involving music synthesis and music transcription.
Part 1: Analyze touch-tone signals ("reverse engineering")
Analyze touch-tone signal spectra

- Tones from 12 keys on phone keypad in file project2.wav
  - Each sampled at 8192 \( \frac{\text{Sample}}{\text{Second}} \); durations 0.5 second.
  - Use \( [x, S] = \text{audioread('project2.wav')}; \text{soundsc}(x, S) \)

- Use \( \text{abs(fft(\ldots))} \) to analyze spectrum of each tone.
  - Fact: each touch-tone signal is a sum of one or more sinusoids.
  - Determine \# of sinusoids and frequencies (in Hz) for each tone.
  - How will you determine the \# of sinusoids? ??

- Relate frequencies to touch-tone keypad
  - Look for patterns.
  - (The frequency assignments were not random; e.g., perhaps all odd buttons use a certain frequency?)
  - Tabulate which button produces which frequencies.

(Notice how brief this description is now.)
Part 2: Touch-tone synthesizer
Touch-tone synthesizer: keypad GUI

- Create touch-tone keypad GUI (cf. Project 1)

- User “dials” (!) by clicking on a sequence of buttons

- Simplification: user clicks on “end” button, causing entire signal to be played using sounds and stored to file touch.wav via:
  
  ```
  file = 'touch.wav';
  audiowrite(file, x, S)
  ```

- Simplification: 0.5 seconds (at $S = 8192$ Hz) for each tone

- Synthesizer check:
  
  ```
  [x, S] = audioread('touch.wav'); soundsc(x)
  ```

  should sound like a touch-tone phone “dialing”
Part 3: Touch-tone transcriber
Transcribe touch-tone signals

• Input: \([x, S] = \text{audioread('touch.wav')}\);

• Output: string of numbers: 7631434
  ○ Do not need to include * or # (not in phone numbers)
  ○ No hyphens (dashes) needed

• How to determine # of buttons pressed? ??

• How to determine which button was pressed for each tone?

○ Will arccos method work? ??
○ Will abs(fft(...)) method work? ??
Pattern recognition using correlation

(new signal processing method)

- For this application the possible signal patterns are few:
  - There are only 12 buttons, each with a unique signal.
  - Each signal is sum of a very small number of sinusoids.
  - We know all the possible frequencies.

- Do not need \texttt{abs(fft())}. It is unnecessarily expensive!

- New approach: a type of \textit{pattern recognition}
  - Find specific frequencies using \textit{correlation} operation, also known as a \textit{matched filter}.
  - We try to find the sinusoids that “best match” the signal. \[\text{[wiki]}\]
  - Correlation is quite robust to noise / interference.
  - Correlation is even faster / cheaper than FFT when \# of possible frequencies is much smaller than \# of signal samples.
Correlation example: \( \sum_{n=1}^{N} x[n] p[n] \)
Correlation implementation: Basic

- Correlation method: multiply and sum.
  - Multiply input signal $x$ by (each) candidate pattern signal.
  - Sum the resulting product signal: gives correlation value.
  - Choose candidate pattern with largest correlation.

- Example implementation in Matlab:
  - `freqs = [110 120 130]`  
    (The candidate frequencies in this example.)
  - `corr = x * cos(2*pi*[1:N]'*freqs/S)`  
    (Returns array of 3 correlation values for the candidates.)
  - `[~, index] = max(corr)`  
    (Returns largest correlation value (unimportant here) and its index in `corr` array.)
  - `freqs(index)`  
    frequency of the sinusoid that best matches signal $x$

- Key trick: `sum(x .* pattern)` same as `x * pattern'`
Correlation implementations in Matlab

Given two (row) vectors of same length:
\[x = [x_1 \ x_2 \ \ldots \ x_N]\]
\[y = [y_1 \ y_2 \ \ldots \ y_N]\]

Mathematical formula: \[\text{Correlation}(x,y) = \sum_{n=1}^{N} x_n y_n\]

Fortran/BASIC/C/C++ style implementation in Matlab:
\[
corr = 0;
for \ n=1:\text{numel}(x)
    corr = corr + x(n) * y(n)
end
\]

Faster Matlab “vectorized” style implementation (cf. math):
\[
corr = \text{sum}(x \ .* \ y)
\]

Even less typing Matlab implementation (for row vectors):
\[
corr = x * y';
\]
Correlation implementation: Improved

The preceding implementation works fine for a \( \cos \) signal, but not if it is \( \sin \). We want it to work for both. In fact we want it to work for a sinusoid of \textit{any} phase.

Improved implementation to be used in Project 2:

- \( \text{freqs} = [110 \ 120 \ 130] \)
  (Your frequencies will be different; from Part 1)
- \( N = \text{length}(x); \ S = 8192; \)
- \( \text{corr} = (x \times \cos(2\pi[1:N]'*\text{freqs}/S)).^2 \ldots \)
  + \( (x \times \sin(2\pi[1:N]'*\text{freqs}/S)).^2 \)
- \( [~, \text{index}] = \max(\text{corr}) \)
- \( \text{freqs}(\text{index}) \)

- Basic idea: find which candidate frequencies best match \( x \) either as a \( \cos \) or as a \( \sin \) wave.
- After finding frequencies, decode which button was pressed for each 0.5 second segment of synthesizer output.
Summary of Transcriber Specifications

- Length of each phone digit known: 0.5 sec.
- Sampling rate known: \( S = \frac{8192 \text{ Sample}}{\text{Second}} \).
- Touch-tone signal written to file touch.wav.
- Use correlation to find best match for each segment.
- Do not use \( \text{abs}(\text{fft}()) \): too much computation!
- Do not use numerous \text{if} statements: inefficient.
- Output: String of phone digits without hyphen.
Part 4: Investigating transcriber accuracy in the presence of noise
Transcriber accuracy in noise: Overview

• Phone signals, wired or wireless, have noise present. (Music signals too.)
• So far we have mostly ignored noise.
• Noise: What exactly is noise?
• Performance: How well does your transcriber work when noise is present (as in real world)?
• Figure of merit: Numerical measurement of performance of a system (detector, estimator).
Zero-Mean Additive White Gaussian Noise

\[ y(t) = x(t) + \varepsilon(t) \]

- Measured signal = Ideal signal + Additive noise

- Additive White Gaussian Noise (AWGN) is a good model for many actual sources of noise.

- Why called white? ??

- At each time \( t \): \( \varepsilon(t) \) has a Gaussian distribution (bell curve).

- At any two times \( t_0 \) and \( t_1 \), no matter how close:
  - \( \varepsilon(t_0) \) and \( \varepsilon(t_1) \) are completely uncorrelated:
  - knowing \( \varepsilon(t_0) \) will not help one predict \( \varepsilon(t_1) \)
  - In words, \( \varepsilon(t) \) is “completely random”

- Zero-mean implies the DC value is zero.
White Gaussian Noise Example

To hear it:

\[
z = \text{randn}(8000,1); \ \text{soundsc}(z);
\]

Why does it sound like the wind or the ocean surf?
Touch-tone transcriber performance measure

- Noise level rises $\implies$ transcriber gets more digits wrong.
- We want to investigate and quantify this error rate.
- Key factor: signal level relative to noise level.
- Signal-to-noise ratio (SNR) in decibels (dB):
  Mathematically: $\text{SNR} = 10\log_{10}\left(\frac{\sum x^2[n]}{\sum \varepsilon^2[n]}\right)$
  In Matlab:
  $\text{SNR} = 10*\log10(\text{sum}(x.^2) / \text{sum}(\text{noise}.^2))$
  where:
  - $x$ is vector of ideal signal values
  - noise is vector of noise values.
Transcriber error rate plot

- SNR is on horizontal axis;
- error rate (percentage) is on vertical axis.
Touch-tone performance investigation

- Transcriber gets some digits wrong.
- Error rate: Fraction of wrongly decoded digits.
- Need to “survey” many digits to get accurate measure.
- For each SNR: use 100 digits, count # decoded incorrectly.
- Each call to \texttt{randn} generates new noise values.
- Random digits versus same digit each time?
- Plot: error rate vs SNR for several SNR levels.
Transcriber Performance: Comments

• Any potential transcriber customer will want to see your plot of error rate vs SNR.
• Below some *threshold SNR*, error rate will rise rapidly
• What transcriber error rate is acceptable?
• What noise level can your transcriber tolerate?
• How to achieve even better performance?
  Error-correction. Digital communications: EECS 455
Outline of Matlab Program for Noise Study

Investigating transcriber performance in noise (pseudo-code)

clear; x = "signal_for_button_1";
errors = zeros(10,1); snr = zeros(10,1);
for level=1:10
    nsum=0; err=0;
    for trial=1:100
        noise = 5 * level * randn(size(x));
        y = x + noise;
        nsum = nsum + sum(noise.^2);
        % apply your transcriber to signal "y" here
        if "transcriber_does_not_output_1"
            errors(level) = errors(level) + 1;
        end
    end
    snr(level) = 10*log10(sum(x.^2)/(nsum/100);
end
plot(snr, errors, '-o')
Code explanation

- Outer loop over 10 different SNR levels: different noise strengths because of $5*\text{level}*\text{randn}()$
- Inner loop over 100 trials; each trial with different random noise realization
- Use the signal for button "1" each time; makes things easier.
- Count # of errors in 100 trials; By using 100 trials, this will be error rate as a percentage.
- $\text{nsum}/100$ is *average* noise power over 100 trials
- $\sum(x.^2)$ is signal power.
- $\text{snr}$ and $\text{errors}$ are both arrays of 10 values.
Note the *threshold* at an SNR of about -20 dB.
- Below this, error rate increases dramatically
- For any SNR below 0, the noise level exceeds the signal level, yet your transcriber still works down to about -20 dB!
  This is because *correlation* is very *robust* to noise.
Summary of Project 2

1. Reverse engineer touch-tone frequencies using `fft`
2. Simple GUI for touch-tone synthesizer
3. Touch-tone transcriber based on correlation
   - Your transcriber will work perfectly in absence of noise
4. Investigate how your transcriber error rate increases as noise level increases

Correlation is one of several “pattern recognition” techniques used in numerous signal processing applications, including SONAR, RADAR, ultrasound imaging, fingerprint recognition, retina scans, ...

Read Project 2 before lab this week!

(Discuss Exam 1 solutions if time permits.)