Eng. 100: Music Signal Processing

DSP Lecture 3

Project 1

Curiosity (animated sheet music):

- http://www.youtube.com/watch?v=Rhv8i0Y08TY

Announcements:

- Lab 2 answers due this week.
- **Homework 1 due start of class next week.**
- Read Project 1 before lab section this week. (reading questions...)
- project1_example.m on Canvas (Files/projects)
- Schedule next two weeks:
  - Tue. Sep. 29: Tech Comm (!)
  - Thu. Oct. 1: Lab 3 lecture
  - Tue. Oct. 6 & Thu. Oct. 8: Lab 3 lectures continued
  - 3 DSP lectures related to Lab 3 before you begin...
Office hours

- Faculty
  - J. Fessler, Thu. 9:30-10:30AM, 4431 EECS
  - P. Kominsky, Thu 12-1PM, 311 GFL
  - T. Bowden, Thu 1:30-2:30PM, 316 GFL

- Lab instructors
  - Sydney Williams: Tue. 12-1PM, EECS atrium CAEN lab
  - Izzy Salley: Tue. 1-2PM, Shapiro undergrad library 1st floor

Go to any / all! (cf. high school)

History:
- 8192 Sample Second: default in Matlab sound command.
- Only option for some old (Sun) computer sound cards.
- Why? Perhaps: speech frequencies [wiki]
Calvin: I think we've got enough information now, don't you?
Hobbes: All we have is one “fact” you made up.
Calvin: That’s plenty. By the time we add an introduction, a few illustrations, and a conclusion, it will look like a graduate thesis. Besides, I’ve got a secret weapon that will guarantee me a good grade! No teacher can resist this! A clear plastic binder! Pretty professional looking, eh?
Hobbes: I don’t want co-author credit on this, OK?
Not one but two facts

- We can make basic music out of sinusoidal signals.
- We can determine the frequency of a sinusoidal signal using:

\[
    f = \frac{S}{2\pi} \arccos\left( \frac{x[n+1] + x[n-1]}{2x[n]} \right).
\]

(Pure sinusoid sampled at \( S \text{ Sample Second} \), using just 3 consecutive samples.)

Other key points from previous lecture:
- \( \omega = 2\pi f / S \) is the digital frequency of a sampled sinusoidal signal
- Models that arise frequently in music (and in perception, nature, engineering):

<table>
<thead>
<tr>
<th>Model</th>
<th>Formula</th>
<th>Log Formula</th>
<th>Graph Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponential</td>
<td>( y = ba^x )</td>
<td>( \log(y) = \underbrace{\log(a)}<em>{\text{slope}} x + \underbrace{\log(b)}</em>{\text{intc.}} )</td>
<td>semi-log</td>
</tr>
<tr>
<td>Power</td>
<td>( y = bx^p )</td>
<td>( \log(y) = \underbrace{p}<em>{\text{slope}} \log(x) + \underbrace{\log(b)}</em>{\text{intc.}} )</td>
<td>log-log</td>
</tr>
</tbody>
</table>
Outline

Part 0: Lab 2 summary

Part 1: Music notes and notation

- Musical notes: frequencies and relations
- Basic musical staff and MIDI numbers

Part 2: Project 1

- Musical tone synthesizer and transcriber:
- Specifications for the two deliverables;
- Two new Matlab commands for graphics
Part 0: Lab 2 summary
Lab 2 summary

What you learned (hopefully)
• Frequencies of musical tones in “The Victors”:
  392, 440, 494, 523, 587, 659 Hz (rounded to nearest integer)
• Semi-log plot revealed missing frequencies:
  415, 466, 554, 622, 698, 740 Hz (mostly accidentals).
• 12 frequencies with common ratio about 1.06

The exact ratio is $2^{1/12} \approx 1.059463094359295$
Twelve such “half steps” in an “octave” leads to a frequency ratio of 2, i.e., the frequency doubles each octave. More later...

What model relates piano key number to frequency?
(exponential or power) ??

Next: How should we represent music symbolically?
Part 1: Music notes and notation
Music communication / representation

How does one person convey musical ideas to others?

A (technical?) communications problem!

Some options:
- Oral
- Pictures
- Video (?) (cf. animated sheet music example)
- Numbers
- Letters
- Words
- Diagrams / Symbols
Music communication: Pictures

http://furulya.hu/historikus-ujjrendek/ujjrendek/Virgilliano.jpg

“The Victors” in pictures
Music communication: Video?
Music communication: Numbers

List the frequencies, durations, and volume of each note?

“The Victors” as a list of frequencies, durations, volumes:

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Duration</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>494</td>
<td>4</td>
<td>extra loud</td>
</tr>
<tr>
<td>392</td>
<td>2</td>
<td>loud</td>
</tr>
<tr>
<td>440</td>
<td>2</td>
<td>loud</td>
</tr>
<tr>
<td>494</td>
<td>2</td>
<td>extra loud</td>
</tr>
<tr>
<td>392</td>
<td>2</td>
<td>loud</td>
</tr>
<tr>
<td>440</td>
<td>2</td>
<td>loud</td>
</tr>
<tr>
<td>494</td>
<td>2</td>
<td>loud</td>
</tr>
<tr>
<td>523</td>
<td>4</td>
<td>loud</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Very “physics” representation.
Reading this while performing would be hard.
Transposing (playing in another key) would be very hard.
Music communication: Numbers

- MIDI = Musical Instrument Digital Interface (defined 1982) [wiki]
- Musical frequencies represented by integers (Happy computer!)
- \[\text{MIDI number} = 69 + 12 \log_2(\text{frequency in Hertz}/440)\]
- “The Victors” represented in a computer via MIDI numbers:
  \[
  \{71, 67, 69, 71, 67, 69, 71, 72, \ldots\}
  \]
- MIDI files also include amplitude (volume) and duration

Transposing (playing in another key) is easy:
just add or subtract an integer from all numbers.
MIDI: Designed by engineers!

- Defined “recently” (1982) by an engineer
- Designed by Dave Smith, “Father of MIDI”
- He chose this formula for the MIDI “pitch number:”
  \[
  \text{MIDI} = 69 + 12 \log_2(\text{frequency in Hertz/440})
  \]
  - Why?
    - Allowed MIDI pitch numbers are 0,1,...,127
    - Lowest note on a piano A0 is 27.5 Hz, MIDI=21
    - A440, aka A4, just above middle C is 440 Hz, MIDI=69
    - Highest note on a piano C8 is 4186 Hz, MIDI=108
- What is the lowest possible MIDI frequency? ??
- Highest? ??
- Why 0 to 127? ??

Project 1 transcriber would be easier if the specification were to produce MIDI output...
Music communication: Letters

- 7 natural notes are named A, B, C, D, E, F, G.
- These include the 6 notes that appeared in “The Victors”
- These 7 occur the most frequently in many songs.
  cf. Morse code
  Aside: “CQD” abbreviation for “sécu” (from the French word sécurité) distress call, [wiki] about 100 years before “OMG”
- These are the “white notes” on a piano
- 5 “accidentals” labeled A♯, C♯, D♯, F♯, G♯
- 5 equivalently labeled B♭, D♭, E♭, G♭, A♭
- A♯ (“A-sharp”) same as B♭ (“B-flat”): between A and B

“The Victors” in letters: B G A B G A B C …
Transposing between different keys requires different sequence of letters - not easy.

More than 1000 years ago (!), relative frequencies were named; this communication system is called solfège or solfeggio.

*cf.* “The Sound of Music” score by Rogers and Hammerstein

In the key of G, a “dictionary:”

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F#</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do</td>
<td>392</td>
<td>440</td>
<td>494</td>
<td>523</td>
<td>587</td>
<td>659</td>
<td>740</td>
<td>784</td>
</tr>
</tbody>
</table>

In theory, by learning songs in solfège, you can play in any key.

“The Victors” in solfège: Mi Do Re Mi Do Re Mi Fa Do Re Mi ...
Music communication: Diagrams / Symbols

Musical staff representation:

- The “natural” notes have the simplest symbols.
- Script “G” encircles line where “G” note is. “Treble clef”
- (Accidentals are adorned with ♯ or ♭.)
- A professional-grade transcriber must look much much better!

“The Victors” in (crude) musical staff notation:

Higher frequencies are put higher. (Computer unfriendly format.)
Music communication: Matlab plots

Matlab was designed for engineering, not for music notation!

“The Victors” in a (crude) Matlab plot:

We focus here on the music signal processing...
A graphical dictionary

Without valves (open)
2nd valve a minor 2nd
1st valve major 2nd
1st+2nd valve minor 3rd
(like so 3rd valve alone)
2nd-3rd valve major 3rd
1st+3rd valve perfect 4th
1st+2nd+3rd valve diminished 5th

Bb
Trumpet
Fingering
Chart
Graphical communication in EE

Picture

Diagram with symbols


http://www.epanorama.net/circuits/hybrid-circuit.gif
Frequencies of musical tones

<table>
<thead>
<tr>
<th>Note</th>
<th>A</th>
<th>A♯, B♭</th>
<th>B</th>
<th>C</th>
<th>C♯, D♭</th>
<th>D</th>
<th>D♯, E♭</th>
<th>E</th>
<th>F</th>
<th>F♯, G♭</th>
<th>G</th>
<th>G♯, A♭</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREQ 440Hz</td>
<td>2⁰/₁²</td>
<td>2¹/₁²</td>
<td>2²/₁²</td>
<td>2³/₁²</td>
<td>2⁴/₁²</td>
<td>2⁵/₁²</td>
<td>2⁶/₁²</td>
<td>2⁷/₁²</td>
<td>2⁸/₁²</td>
<td>2⁹/₁²</td>
<td>2¹⁰/₁²</td>
<td>2¹¹/₁²</td>
</tr>
<tr>
<td>Hz</td>
<td>440</td>
<td>466.2</td>
<td>493.9</td>
<td>523.3</td>
<td>554.4</td>
<td>587.3</td>
<td>622.3</td>
<td>659.3</td>
<td>698.5</td>
<td>740.0</td>
<td>784.0</td>
<td>830.6</td>
</tr>
<tr>
<td>MIDI</td>
<td>69</td>
<td>70</td>
<td>71</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
<td>80</td>
</tr>
<tr>
<td>Ratio</td>
<td>1/1</td>
<td>none</td>
<td>9/8</td>
<td>6/5</td>
<td>5/4</td>
<td>4/3</td>
<td>none</td>
<td>3/2</td>
<td>8/5</td>
<td>5/3</td>
<td>16/9</td>
<td>15/8</td>
</tr>
<tr>
<td></td>
<td>440</td>
<td>?</td>
<td>495</td>
<td>528</td>
<td>550</td>
<td>586.6</td>
<td>?</td>
<td>660</td>
<td>704</td>
<td>733.3</td>
<td>782.2</td>
<td>825</td>
</tr>
</tbody>
</table>

Frequency ratios are called *intervals* in music:

- **half step** $2^{1/12}/1$
- **whole step** $9/8 \approx 2^{2/12}$ ratio (2 half steps)
- **octave** $2/1$ frequency ratio (e.g., middle C to next C) (12 half steps): $(2^{1/12})^{12} = 2$
- **fifth** $3/2 \approx 2^{7/12}$ ratio (7 half steps, logically?)

The 3/2 ratio comes from modes of vibrating string (or air column); used as “drones” in bagpipes, hurdy gurdy, guitar “power chords.”

Using the $2^{1/12}$ ratio is called *equal temperament* (cf. small integer ratios).

J.S. Bach’s “The Well-Tempered Clavier” (1722)

Some Arabic music uses $2^{1/24}$ ratio (24 notes per octave)
Part 2: Project 1
Project 1: Specification summary

- Pure sinusoidal-tone music **synthesizer**:
  - Matlab-generated virtual keyboard via GUI
  - Click on key with mouse generates that note
  - Writes audio signal for sequence of played notes into .wav file
  - Diagram: Music notes → keypress → Sythesizer generates sinusoidal tone sequence signal $x[n] → file.mat$

- Pure sinusoidal-tone music **transcriber**:
  - Input: a .mat file of music played on synthesizer
  - Output: pseudo-musical-staff notation of music
  - Diagram: File.mat → Transcriber → Music staff

- Detailed specifications on subsequent slides.
- Suggestion: output MIDI notation first, then make staff.
Project 1: Tone Synthesizer GUI

- Mimics single octave of a piano keyboard: black (above) and white (below) keys.
- You can jazz this up (add colors, labels) if you so desire.
- Clicking on key with mouse plays that note using sound.
- Clicking on “end” causes your song to be played and its signal written to a file using save.
Project 1: Tone transcriber output

signal file.mat $\rightarrow$ transcriber $\rightarrow$

- Crude musical staff notation of “The Victors.”
- Musical notes: uses stem plot for simplicity.
- No duration information: all notes have same length.
- Heights against 5-line background “staff” must be correct.
- Determining the notes / pitches / frequencies and computing these heights from sampled music signals generated by your synthesizer is the main DSP point of Project 1.
Matlab GUI command for Project 1

The most (only?) important GUI command:

• `uicontrol('style', 'pushbutton', 'position', [100 250 40 80], 'string', 'Db', 'callback', '***')`

• Creates button 40 pixels wide and 80 tall, located 100 pixels from left and 250 above bottom of screen, with “Db” label

• When “pushed” by left-clicking with mouse, Matlab executes the command(s) “***” (you must provide),

• e.g., the command string `'load train; sound(y)'` causes the train signal to be loaded and played.

• Use many of these commands to create your keyboard GUI.

• One of the few “generic programming” commands in course; most other commands are related to music signal processing.

Example (remarkably concise): (note character/string arguments vs numerical arguments)

```
uicontrol('style', 'pushbutton', 'position', [100 250 40 80], 'string', 'choo', 'callback', 'load train; sound(y)'
```

26
Matlab plotting commands for Project 1

• To make a stem plot of the numbers in vector \( y \) that mimics musical staff notation:

\[
\text{subplot}(311), \text{stem}(y), \text{axis}([0 \text{ length}(y) 0 4])
\]

For filled-in circles, use: \text{stem}(y, 'filled')

• To create the five musical staff lines:

\[
\text{set(gca, 'ygrid', 'on', 'ytick', [0:4])}
\]

For solid staff lines:

\[
\text{set(gca, 'ygrid', 'on', 'ytick', [0:4], 'GridLineStyle', '-')}
\]

Example: \( y = [2 \ 1.5 \ 1 \ 1.5 \ 2 \ 2 \ 2 \ 1.5 \ 1.5 \ 1.5 \ 2 \ 3 \ 3] \);

Name that tune ??
Musical tone transcriber: Plotting accidentals

- If using MIDI notation, this would be easy.
  - Find frequencies of each note in Hz; put in vector \( F \).
  - Compute MIDI numbers using formula:
    \[
    \text{midi} = 69 + 12 \times \log_2(F / 440);
    \]
  - Just stem-plot the result: \text{stem(midi)}
  - But this plot would be poorly suited to musicians

- Challenge: musical staff notation treats natural notes differently from accidentals (sharps & flats)

- Solution:
  - Plot natural notes at 0, 0.5, 1, 1.5, ...
  - Plot accidentals “half way in between” at 0.25 0.75 etc.
  - Use “table lookup” via 
    \text{Matlab} indexing, shown next.
Using a Matlab vector for “table lookup”

<table>
<thead>
<tr>
<th>Note</th>
<th>Frequency</th>
<th>MIDI number</th>
<th>vertical position</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>64</td>
<td>64</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>65</td>
<td>65</td>
<td>0.5</td>
</tr>
<tr>
<td>F♯</td>
<td>66</td>
<td>66</td>
<td>0.75</td>
</tr>
<tr>
<td>G</td>
<td>67</td>
<td>67</td>
<td>1</td>
</tr>
<tr>
<td>G♯</td>
<td>68</td>
<td>68</td>
<td>1.25</td>
</tr>
<tr>
<td>A</td>
<td>440</td>
<td>69</td>
<td>1.5</td>
</tr>
<tr>
<td>A♯</td>
<td>466</td>
<td>70</td>
<td>1.75</td>
</tr>
<tr>
<td>B</td>
<td>494</td>
<td>71</td>
<td>??</td>
</tr>
<tr>
<td>C</td>
<td>523</td>
<td>72</td>
<td>??</td>
</tr>
<tr>
<td>C♯</td>
<td>554</td>
<td>73</td>
<td>??</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Vertical positions of the stem circles are unequally spaced!

Wrong way (works for E, G only): stem((midi-64)/3)

We need a simple automatic way to specify vertical position given MIDI number.
Solution by table lookup

Define a table (array) of vertical positions (here for E to D only):
\[ V = [0 \ 0.5 \ 0.75 \ 1 \ 1.25 \ 1.5 \ 1.75 \ 2 \ 2.5 \ 2.75 \ 3]; \]

Use Matlab indexing to “lookup” vertical positions:
\[
\text{stem}(V(\text{midi}-63)) 
\]
(discuss indexing, next slide)

Example (for B A G A B B B A A A ...):
\[
\text{midi} = [71 \ 69 \ 67 \ 69 \ 71 \ 71 \ 69 \ 69 \ 71 \ 74 \ 74];
\]
\[
\text{disp}(V(\text{midi}-63))
\]
\[
2.0 \ 1.5 \ 1.0 \ 1.5 \ 1.5 \ 2.0 \ 2.0 \ 2.0 \ 1.5 \ 1.5 \ 1.5 \ 2.0 \ 3.0 \ 3.0
\]
\[
\text{stem}(V(\text{midi}-63), \ 'filled') ,
\]
\[
\text{set(gca, \ 'ygrid', \ 'on', \ 'ytick', \ [0:4], \ 'ylim', \ [0 \ 4])}
\]
Matlab array indexing examples

%%
x = [10:5:30]
x = 10 15 20 25 30

>> x(2)  
15

>> x(2:4)  
15 20 25

>> x(3:end)  
20 25 30

>> x([4, 1:2:3])  
??
Musical tone transcriber: Outline

- Synthesizer output is a signal with $M$ tones concatenated.
- Simplification: transcriber “knows” each tone has length: $N=2000$ samples
- Transcriber must determine number of tones $M$. How? ??
- Reshape 1D signal into array of $M$ tones of length $N=2000$:
  \[ y = \text{reshape}(x, 2000, M); \]
- Apply simple arccos frequency estimation method:
  \[
  f = \frac{S}{2\pi} \arccos\left( \frac{x[n+1] + x[n-1]}{2x[n]} \right).
  \]
  
  \[ n = 1000; \quad \text{ratio} = (y(n+1,:) + y(n-1,:)) ./ (2 * y(n,:)); \]
  \[ F = (S/2/pi) * \text{acos(ratio)}; \quad \% \text{hail to the vectors!} \]
  
  (Use Matlab to explain about $y(3,:)$ etc.)

- Generate MIDI number array from frequency vector $F$.
- Use table lookup to make final stem plot.
Project 1: Process

- Create Tone Synthesizer and Tone Transcriber (two .m files).
  p1_yourteamname1.m  p1_yourteamname2.m
  Have your lab instructor confirm that both of these work.
  Upload your two m-files in a zip file to Canvas.

- Your team will present your project results orally.
  You will learn how to do this in tech comm part.
  All 50 points of your Project 1 grade based on presentation.

- Save your presentation as a .pdf, not .pptx, file.
  (Windows: Select “print”; print to “Adobe” at top of menu.)
  Google slides may be convenient for collaborative editing.
  The file name should be just your team name.

- Upload pdf of your presentation to Canvas.
  (Further instructions from discussion leaders.)
Matlab/ DSP in practice - more ways than you realize

How do you create and back test your trading strategies?

Are you developing algorithmic or automated trading systems?

Whether you are new to MATLAB, or you’re a long-time user, there are many options that can help you generate trading ideas, test them, and deploy them to traders and automated trading systems.

» View a short video on connecting MATLAB to Interactive Brokers with Trading Toolbox.

» Watch a technical presentation on developing and testing algorithmic trading strategies.

» Learn about cointegration and watch a presentation to see how it can be used for pairs trading.

» Explore user-contributed files about real-time trading in MATLAB on MATLAB Central.

Get Started with Trading Toolbox: Connect to Interactive Brokers 7:22

Automated Trading with MATLAB 64:54

Cointegration and Pairs Trading with Econometrics Toolbox 61:27

Download demo and presentation

(return to end of l02 if time permits)
Summary for Project 1

- Read Project 1 before lab section this week.
- `project1_example.m` on Canvas (Files/projects)