Eng. 100: Music Signal Processing F15 J. Fessler
Project 3: Music synthesizer and transcriber (etc.)

0 Prologue

This document contains the "classic" Project 3 DSP specifications. This year, student teams are allowed and encouraged to propose and execute a music signal processing project of their own design. Regardless of the signal processing component, all student teams must meet all technical communications deliverables (proposals, progress reports, team documents, final report, presentations, etc.), summarized at the end of this document and described in more detail elsewhere. The final oral presentation must include a live demonstration of the project.

The project description below gives student teams perspective on the scope of the project work expected for these final projects. The "project 3 classic" itself has several open-ended aspects, including designing one or more unique instrument sounds and choosing the transcription algorithm. Variations on "project 3 classic" are perfectly acceptable. For full credit, teams projects must be at least as sophisticated as "project 3 classic."

1 Abstract

This is the main project for the course. It has two parts: (1) Programming a simple music *synthesizer* in `Matlab` using a `Matlab` musical GUI similar to the one you developed in Project 1, now using snippets of real instrument sounds; (2) Programming and evaluating a simple music *transcriber* in `Matlab` using the `Matlab` stem-based staff-like notation in Project 1, but now including note duration information.

2 Background

Music synthesizers use a variety of techniques. A simple approach to music synthesis is to record the sounds of actual musical instruments and play them back at variable speeds to produce the desired pitches, called *wavetable synthesis*. Another simple approach is to filter basic signals like square waves or sawtooth waves to achieve a desired effect, called *subtractive synthesis*. Another approach (that requires more circuitry if implemented with analog electronics) is to produce each harmonic of a note separately and add these sinusoids together, called *additive synthesis*.

Musical transcription that generates sheet music (or the equivalent) directly from a musical recording is much more challenging, particularly for polyphonic music. It is not a completely solved problem, and modern techniques involve signal processing concepts such as hidden Markov models that are graduate-level material, because real music is more complicated than the periodic signals whose periods change abruptly that we have considered in this course.

Nonetheless, transcription of simple music can be accomplished using the techniques you have learned in this course. You saw in Lab 3 that the spectrogram could itself function as a type of musical notation. The correlation approach from Project 2 is another option, as are the autocorrelation method and harmonic power spectrum method that will be described in lecture. Your team should consider more than one approach and select the one that you think is the best for this project.

The results of this project will be two main `Matlab` m-files. One file implements a music synthesizer that accepts on-screen keys clicked with a mouse and outputs a melody using any of several musical instruments (guitars, clarinets, and others) selected from an on-screen menu. Optionally, your synthesizer can add ("mix") multiple sounds together to achieve the effect of several instruments played simultaneously.

The other m-file implements a music transcriber that accepts a `.wav` file of music from your synthesizer and generates a musical-staff-like transcription using `Matlab`'s `stem` command. Details are specified below. Some

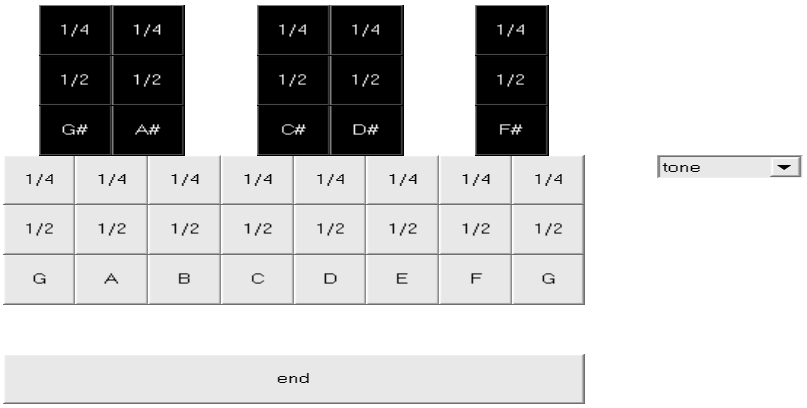
suggested approaches for the transcriber will be discussed in lecture and it may be helpful to refer to those slides.

3 Project 3: What you have to do

- Download the file `proj3.wav` from Canvas. This file contains snippets of length 32768 samples each from an electric guitar, a clarinet, a trumpet, and a single tone, all sampled at $44100 \frac{\text{Sample}}{\text{Second}}$. Each instrument played the 13 notes of a full octave in succession. Your synthesizer will include these four instruments played back using these samples, which is called "sample-based synthesis."
- Design at least one additional instrument of your own using additive synthesis (adding together harmonics with amplitudes selected by you) and label it with your team name.
- Also create a "trumpet section" sound by reverberating each trumpet note. Do this by adding copies of trumpet to itself, with each copy delayed slightly from the previous copy.

3.1 Synthesizer specifications

- Your synthesizer should have a keyboard with note durations, like those shown below.



- Use the following note durations.

Note	Whole	Half	Quarter	1 second
Length	$32668 + 100$	$16284 + 100$	$8092 + 100$	$S = 44100$

  The final 100 samples of each note should be zeros, to provide duration information.
- The pull-down menu should offer the user a choice of instruments, selected prior to use of the keyboard.
```
h = uicontrol('style', 'popup', 'position', [500 250 100 50], ...
'string', 'guitar|clarinet|trumpet|tone|TeamSound1|trumpetsection');
```
- To determine which instrument the user selects use: `pause; v=get(h, 'value');`
  This generates v=1,2,3,4,5 or 6 depending on the user's selection.
  Alternatively you can incorporate the `v = get(h, 'value')` command as part of the `'callback'` option of the `uicontrol` command.
- When the `end` button is pressed, the song should be played back to the user and the signal should be written to a `song.wav` file. Do not use a `.mat` file here; `.wav` files can be played on many computers with many types of software, but `.mat` files are only useful with the (expensive) `Matlab` software so it would not be very useful to musicians!
- We encourage you to be creative and add desirable features to the basic GUI shown above. You may modify the layout as long as your approach can generate (at least) all 13 notes and all 3 durations, and is at least as "easy to use" as the layout above.

3.2 Transcriber specifications

- The transcriber should output a musical staff and notes like in Project 1, except that here the separation between transcribed notes should indicate the length of each note:
  a whole note is followed by 3 extra spaces; a half note by 1 extra space; a quarter note by none. Hints:
  - Use `reshape` and look for columns ending in zeros.
  - Let `t` be the indices of those columns.
  - Then you can use `stem(t,v)` where the vertical positions `v` come from Project 1.
- The transcriber need not be able to handle the electric guitar (lower bass scale).
- The transcriber must be able to handle your additive synthesis instrument (or the first such instrument if you make more than one).
- If you add extra features to your synthesizer such as polyphonic music or additional instruments, your transcriber is not required to handle those.
- Perform *noise investigations*, like in Project 2.
  - Use quarter-note durations, including the 100 zeros at the end.
  - Show the error rate versus SNR for at least 5 instruments on one plot (with 5 curves).
  - Vary the SNR enough that the error rates range from 0 to at least 50% for every instrument.
  - Use `Matlab`'s `legend` command to show clearly which curve goes with which instrument.
  - As always, label your axes.

3.3 Hints

- Start testing your transcriber before your synthesizer is finished. Instead of using `x = audioread(...)` to read in the audio file created by the synthesizer, create simple test tones:
  `S = 44100; N = 32768; x = cos(2*pi*440*[1:N]/S); x(end-99:end) = 0;`
  The last line sets the last 100 values of `x` to zero just as the synthesizer must do.
- After you get the transcriber working for individual tones you can use notes from the `project3.wav` file, and sequences of tones and notes.

3.4 Deliverables

This section of the specifications applies to all projects, not just project 3 classic. You will present your results in both written form (a technical report) and orally to the class and instructors, who will function here as your co-workers and bosses, respectively. Your deliverables will be graded by both your DSP professor and your TC instructor.

Because this is the main design project in the course, there will be several deliverables along the way. See the course schedule and Canvas for due dates and points.

Deliverable	where
P3 proposal	pdf to Canvas
P3 progress report	pdf to Canvas
P3 draft oral presentation	with discussion leader
P3 peer evaluation	pdf to Canvas
P3 oral final presentation	ppt or pdf to Canvas with one slide per page and 3 paper copies (4 slides per page)
P3 final report	pdf to Canvas
P3 mfiles (and data)	.zip to Canvas

Do not upload any `.doc` files. Your TC instructors may want hard copy of some or all components.

3.4.1 Proposal

Your proposed project must be at least as sophisticated as project 3 "classic" for full credit. It is wise consult with Prof. Fessler about your project idea before finalizing and submitting your written proposal.

3.4.2 Progress report

Be sure to consider all the feedback from your graded proposal when writing your progress report.

3.4.3 Peer evaluation

The form is provided in the Assignment on Canvas.

3.4.4 Oral presentation

The final oral presentation must include a live demonstration (with audio) of the project. Be sure to test your `Matlab` and audio output well before the day of the presentation!

Your presentation must include at least one spectrum or spectrogram to illustrate the nature of a sound you synthesized or processed. The spectrum axes should be labeled in Hz!

For project 3 classic, the demonstration must include your additive synthesis "addsyn" sound and you must show its spectrum for the low G note.

For projects involving a synthesizer, your presentation must include a screen shot of your synthesizer GUI and a brief audio example from your synthesizer. For projects involving a transcriber, include a screen shot of your transcriber output corresponding to a non-trivial audio example, a description of your transcription method, and an error-rate plot.

3.4.5 Final report

Your report must include the same elements specified for the oral presentation, except that the audio example is optional. (Additional aspects of the presentation and report will be specified in your discussion section.) You will be graded on both the presentation and the results. This is very realistic—great inventions will flourish only if you can explain to bosses and customers what it does and why it is valuable.

Do not include your entire m-files as an appendix in your report. However, you should include at least one block of code in an Appendix of your report that shows how you implemented something interesting about your project, and you should refer to that Appendix in the body of the report. The block of code included in the Appendix should include comments (using the % syntax in `Matlab`) that would help another ENGN 100 student be able to understand what the code block does. You might also include other graphs or images as Appendix material if appropriate.

3.4.6 Software

Upload your project m-files for your project to Canvas by the report deadline. Combine the files into one `.zip` file and upload only that one file. Do not include the original `project3.wav` file in your zip file. If for some reason you modified that file (unlikely), then give it a new name to avoid confusion with the original. Include a readme.txt file (no Word files) if needed to explain how to run your code.