LAB 2: AUDIO SPECTROGRAMS

ECE180: Introduction to Signal Processing

OVERVIEW

Spectrograms provide a valuable graphical tool to visualize the time-domain and frequency-domain behavior of audio signals, and are widely used to study speech signals. In this lab project you will learn how to interpret spectrograms by studying a variety of basic signals including the sine tone, chirp, band-limited pulse, and noise. After establishing some experience you will use the spectrogram to study several speech clips.

OUTLINE

1. Create the spectrogram of a single sine tone
2. Study the chirp signal
3. Study the spectrogram of a band-limited pulse
4. Create the spectrogram of noise
5. Create a time-localize signal with a window
6. Investigate the spectrogram of speech

PREPARATION – TO BE COMPLETED *BEFORE* LAB

Study these tutorial videos:

1. Audio I/O -- http://youtu.be/-PqHK3lHV4o (5:21) (this is new)
2. Script efficiency -- http://youtu.be/td09_g89X9w (2:18) (review these tips to minimize typing while developing a script)
3. Arrays -- http://youtu.be/qEKXgVZN2ts?t=2m41s (refresh your memory on the colon operator, 2:46 to 3:30)
4. Multiple plots -- video http://youtu.be/F_2YMcT8BrI (0:00 to 2:15, directing plots to specific figure window numbers)
5. Scripts -- http://youtu.be/ykEUJBnLDbE (refresher as needed)

Ensure that you have installed the ECE180 MATLAB folders (see Lab 1).

Remember to trade scribe duties.

LAB ACTIVITIES

1. *Create the spectrogram of a single sine tone:*

1.1. NOTE: You will create a single script for the entire lab by extending the script for each part of the lab. Use the double percent-style comment `%%` with a suitable label that includes the part number to delineate each major part of the lab.

1.2. Begin your script with a single sinusoidal tone, display the spectrogram of the tone, and listen to the tone:

(a) Create two variables, one for the sampling rate f_S in hertz and another for the tone duration D in seconds; begin with 8000 Hz and 2 seconds:
```
fs=8000;
D=2;
```
(b) Set up a column-vector for the sampled time array t based on the duration and sampling rate:
```
tt=(0:fs*D-1)'/fs;
```
IMPORTANT: Don't forget the single quote that transposes this 1-D array.

(c) Create the sinusoidal tone $x(t) = \sin(2\pi f_0 t)$ using a variable for the frequency f_0. Begin with 100 Hz:
```
f0 = 100;
x = sin(2*pi*f0*tt);
```
(d) Study `helpwin` (or `doc`) for the ECE180 MATLAB function `spectro` that plays the audio signal and displays its spectrogram; add this line to your script. Omit the `TRES` (time resolution) argument for now. See the next step before you test this part of your script.

1.3. Turn your laptop computer's volume all the way down to zero, and then gradually increase the volume to a comfortable level as you repeatedly run the script by pressing F5.

WARNING: Protect your hearing! Always start at a low volume level so that you do not accidently blast your ears with sound that is too loud. Hearing damage is cumulative during your life, and you need to avoid subjecting yourself to sound levels that can cause damage.

1.4. Run your script to create the spectrogram figure window. Dock the figure window to keep it visible (click the button on the top right of the figure). Press F5 to save and re-run the script each time that you make a change.

1.5. The spectrogram vertical axis shows the frequency content of the signal at any given moment while the horizontal axis plots the time evolution of the signal. The color at each frequency-time coordinate represents the signal intensity; refer to the colorbar to translate a color to its numerical value. The scale is calibrated in *decibels* with +0 dB representing the maximum intensity and increasingly negative values representing lower intensities.

1.6. Try several values of frequency f_0. Confirm that the high-intensity stripe's frequency position matches the value of your selected f_0.

1.7. Try values near $f_0 = 4000$ Hz, both a little below and a little above (where "a little" is about 100 Hz), and then try some higher values up to 8000 Hz. Discuss the behavior that you observe. What is the maximum value of f_0 that produces an unambiguous spectrogram display for a given f_0? In other words, if you desire to know the value of f_0 that you originally typed by looking *only* at the stripe on the spectrogram, what is the highest value of f_0 that allows you to do this? How is this value related to the sampling rate?

2. *Study the chirp signal:*

2.1. Create a *chirp* (a swept-frequency sinusoid) that begins at frequency 40 Hz at time $t = 0$ and ends at 4000 Hz at time $t = D$:
```
x=chirp(tt,40,D,4000);
```

Note: New lines that you add to create the signal `x` simply update the variable workspace variable, therefore it is not necessary to comment out earlier lines. However, be sure to comment out previous `spectro` function calls so that you only have one active `spectro` call in the script. Tip: Highlight one or more lines and press Ctrl+R to convert them to comments; press Ctrl+T to accomplish the reverse.

2.2. ✎ Change the duration to five seconds, and then to ten seconds. What visual change do you observe in the spectrogram?

2.3. ✎ Set the duration back to five seconds, and then add `'lo'` (logarithmic) as the last argument to `chirp` (include the single quotes). Describe the *visual* difference between this spectrogram and that of the previous step. Describe the *aural* difference between this sound and that of the previous step. Which of the two chirp types would you use when you want the frequency to sound like it is steadily increasing at a *constant* rate?

2.4. Refer to `helpwin graph3d` and scroll down to the "Colormaps" section. Experiment with these colormaps beginning with `colormap(gray)`. Note that the spectrogram numerical values always remain the same; the different colormaps simply apply different rules to map intensity onto color.

2.5. Try `colormap(spectral);` this colormap is located in the ECE180 "matlab" subfolder. This map uses distinct color bands to make it easier to associate a particular color with its numerical value on the colorbar. Also, extreme values are presented as white and black which makes it easy to spot the highest and lowest values.

2.6. Any of the colormaps may be inverted by simply subtracting them from 1, e.g., `colormap(1-gray)`.

2.7. Select a colormap that particularly appeals to you and then screenclip the spectrogram. State the colormap name, too.

2.8. The colormap remains in effect while the figure window remains open. Use `colormap(spectral)` from now on.

3. *Study the spectrogram of a band-limited pulse:*

3.1. Create a *band-limited pulse* waveform based on the existing workspace variables that you used to create the sine tone; use `spectro` as before, and begin with $f_0 = 500$ Hz and a duration of two seconds :

```
f0 = 500;
n = 1;
x = diric(2*pi*f0*tt,2*n+1);
```

3.2. ✎ The second argument of `diric` (short for "Dirichlet" function, pronounced "deer ih shlay") controls the number of sinusoidal harmonics that are present in the waveform; a harmonic's frequency is an integer multiple of the fundamental frequency. The value of `n` sets the number of harmonics, and at the moment you should see a single sine tone at 500 Hz (you can also see a tone at 0 Hz, but this is not audible). Increase the value of `n` from 1 to 6, re-running your script each time. Now gradually decrease the fundamental frequency to 50 Hz. Explain how the specific value of f_0 manifests itself in the spectrogram.

3.3. ✎ Gradually increase the number of harmonics to 50. Describe the aural effect of increasing the number of harmonics.

3.4. ✎ By looking at the spectrogram you can now understand why this is called a "band-limited" pulse, i.e., the frequency *bandwidth* (range of significant frequency content) has a well-defined upper limit. Express the bandwidth of this waveform as an equation in terms of the number of harmonics and fundamental frequency.

3.5. The spectrogram itself does not reveal why this waveform is called a *pulse*, however. Open a new figure window and plot the band-limited pulse:

```
figure
```

```
plot(tt,x)
```

Click the "Zoom In" tool, right-click on the plot, select "Zoom Options," and then select "Horizontal Zoom." Click on the figure a few times to zoom in on the waveform details. The plot reveals that the band-limited pulse is a periodic train of narrow pulses. Use Shift+Click to zoom out one level, and right-click the plot and choose "Reset to Original View" if necessary.

3.6. Screenclip several periods of the band-limited pulse waveform.

3.7. Screenclip the spectrogram display.

3.8. Close the band-limited pulse waveform figure window so that future calls to `spectro` will continue to use your docked window.

4. *Create the spectrogram of noise:*

4.1. Use the following code to generate a white noise signal that is the same length as your sampled time array; a duration of two seconds works well for this part:

```
n = dsp.ColoredNoise(0,length(tt),1);
x = step(n);
```

Listen to the noise signal and display its spectrogram. You may also wish to plot the noise signal in a separate figure window. White noise contains equal amounts of all possible frequencies.

4.2. Change the first argument of `dsp.ColoredNoise` to 1 to generate a "pink noise" signal. Describe the visual and aural differences between white noise and pink noise.

4.3. Experiment with other values of the first argument in the range –1.5 to +1.5. Describe the visual and aural effect of this parameter.

5. *Investigate the spectrogram of speech:*

5.1. Read the audio clip "s1.wav" into the workspace, and then use `spectro` with the `spectral` colormap:

```
[x,fs] = audioread('s1.wav');
```

Note that `audioread` returns the audio samples in the first variable and the sampling rate in the second variable. What is the sampling rate of this particular audio clip?

5.2. Create a new sampled time array to match the *exact length* of the sampled audio clip. Copy and paste your finished line (or lines) of MATLAB code for this step. Hint: Adapt the code fragments from Parts 1.2(b) and 4.1.

5.3. Open a new figure window and plot the audio clip time-domain waveform using your sampled time array to calibrate the x-axis; remember to label the x-axis, too. Use `grid on` to make it easier to see the time location of each part of the speech.

5.4. Screenclip the audio waveform plot.

5.5. Study `helpwin` for the ECE180 MATLAB function `snippet` that interactively defines a start and stop time to play a short segment of audio. Estimate and record the start time of each word in the plotted speech waveform.

5.6. Now use `snippet` to make the same estimates from the spectrogram window. What do you observe about the spectrogram in the vowel sound regions of the speech? Which of the signals that you created in the previous parts of the lab looks the most similar to these vowel sound regions? Hint: Try zooming in on the waveform plot of the speech clip.

5.7. Experiment with the time resolution parameter of `spectro`. Begin with the default value of 100 milliseconds and confirm that the spectrogram looks the same. Try higher (coarser) values of time resolution, working your way up to 500 ms. What is the overall effect on the spectrogram?

5.8. Return to 100 ms and then begin working your way down to lower (finer) values of time resolution until you reach 1 ms. What is the overall effect on the spectrogram?

5.9. Look for a time resolution value that maximizes the visible details in both the *frequency domain* (horizontal axis) and in the *time domain* (vertical axis). Screenclip the spectrogram display for your best result and report the time resolution value that you found.

5.10. Study the spectrograms of the remaining speech clips "s2.wav" to "s9.wav" and continue to use your optimum time resolution value. Describe the typical features one can observe in the spectrograms of speech signals, and relate these features to the aural features of the speech signals.

5.11. Screenclip the spectrogram display that most appeals to you, and indicate the speech clip number.

6. *Wrap Up:*

6.1. Copy and paste your documented script as it stands at the end of Part 5.

6.2. Ensure that you have named the completed worksheet for your lab team "Lab 2 _ Firstname Lastname.docx" (first and last name of the scribe) and then submit it to Gradscope by the due date.