Lab 1
Speech Processing

1. Objectives

- To become familiar with the matlab basic commands for speech processing.
- To read a recorded audio file and plot the time/frequency domain characteristics and also plot the spectrogram.
- To separate English phonemes from specific words and plot the phonemes.

2. Reading an audio file into Matlab and playing it back

Okay, assuming we have a speech or audio file "speech.wav" saved in C:\temp, let's load it into the Matlab workspace:

[s,fs,nbits] = wavread('C:\temp\speech.wav');

The semicolon at the end of the command means that we do not wish to print out the values. (Sidenote: If you have the file in "au" file format instead, you can load it using "auread" command, with similar output arguments. Reading of lossy-compressed audio files like MP3 does not seem to be supported by Matlab but there are probably freely available toolboxes for that, try Google search if you are interested).

By the way, you don't need to give the full path if you add the directory to the Matlab search path:

addpath C:\temp

[s,fs,nbits] = wavread('speech.wav');

By now you should have the signal loaded into the workspace. It is stored in the vector "s", its sampling rate is "fs" Hertz and the quantization resolution is "nbits" bits per sample. You can display the values of these variables just by typing the variable name, or by using the "disp"-command. Matlab supports the standard sprintf-functions in "C- language style", so you can try, for instance:

disp(sprintf('The sampling rate is %i Hz and bit resolution %i bits per sample',fs,nbits)); If you want to play back the signal in Matlab, you can use the "soundsc" command. This command needs the vector of the signal samples, and the sampling rate as its arguments:

soundsc(s,fs);

Let's try something fun. Try what happens with these commands:

soundsc(s,fs/2);

soundsc(s,fs*2);

Explain the results.

3. Plotting the waveform

You can plot the signal waveform by typing

plot(s);

The x-axis shows the indices to vector "s" and the y-values are the amplitude values of the signal, corresponding to air pressure variations which are converted to voltage variations using a transducer (microphone). Try zooming the figure (magnifying tool in the figure), **what are the minimum and maximum values of the signal.**

The x-axis are sample numbers and not in seconds – what to do?

Note that, the plot shows the indices to the vector "s" in the x-axis. If we want to plot the time in seconds, we have to first create another vector which contains the labels for x- axis. In order to do so, let's first create a vector which has equal length with the vector "s" and contains linearly spaced values between 0 and 1. This can be done as follows:

xval = linspace(0,1,length(s));

Alternatively, you can use the ":" operator. You are encouraged to learn using the ":" operator, we will be using it every now and then. First, define the step size (increment) between the values as

step = 1/length(s);

Now, you can use ":" operator to give a range of values:

xval = 0:step:1-step;

which says that we want to create a vector whose values start from value zero, end to value 1-step and the increment is defined by "step".

Whether you created the "xval" vector with linspace or the ":" operator, the length of this vector should be the same as your signal. (You can check your variables with the "whos" command, or check the lengths of the vectors by the length-command).

Okay, so we have a vector containing values from 0 to 1. By multiplying each values of this vector with the length of the sample in seconds, we will get a vector which indices the time in seconds, cool! The length of the signal "s" in seconds is length(s)/fs where "fs", again, is the sampling rate. Therefore, we write:

xval = xval * (length(s)/ fs);

Let's plot "s" using the physical time units on the x-axis:

plot(xval,s);

If you want to be less clear, you can do everything by one command as well plot((0:1/length(s):1-1/length(s))* (length(s)/ fs),s);

Adding axis labels and legend

If you want to labels to your axis, this can be done as follows:

xlabel('Time [seconds]');

ylabel('Amplitude'); Try also:

grid on;

legend('This is my first signal');

If you want to save your figure into an image file, you can also do that from the command line:

figure(1), print -dpng 'C:\temp\mysignal.png'

4. **Plotting the spectrogram**

What are spectrum and spectrogram, anyway?

Some features of the signal are visible in its waveform. More insight into the signal's time frequency characteristics can be obtained by studying its spectrum, i.e. the distribution of energy across different frequencies. Speech and audio signals are highly nonstationary signals, meaning that the signal statistics and spectrum are functions of time. Therefore, the signal must be splitted into small "subsignals" what we call frames. Then, we can calculate the spectrum of each frame separately, as if it was an independent signal, and plot it. A representation that is obtained in this way is called spectrogram and it tells the relative energy of the signal at each time and frequency position. To be precise, each frame must be also multiplied by a window function before computing the spectrum1.

How to display a spectrogram in Matlab?

The Matlab function for producing spectrogram is "spectrogram" ("specgram" also works in some older Matlab versions but has slightly different input arguments)2. You can call this function (as usually in Matlab) with a different number of input arguments. From the help file, we will adopt the following command format:

spectrogram(x,window,noverlap,NFFT,fs) ;

Here:

x Signal samples

window Window function (or the length of the default window)

noverlap Overlap between the successive frames in samples

NFFT Number of frequency points in the Fourier spectrum

fs Sampling rate in Hertz

Assume we have a signal "s" loaded to workspace. As before, the sampling rate of the signal is given in variable "fs". First, the typical frame length for analyzing speech is about 20 milliseconds. Let's first convert this into sample count:

framelen_samples = (20/1000)*fs;

Usually, the overlapping region between successive frames is 30% to 50% of the frame length, depending on the application. Therefore, we continue by writing:

noverlap = 0.3* framelen_samples;

Next, we note that the size of the spectrum (number of points in FFT) have to be selected larger than the frame length, and it should be a power of two (assumed by the FFT algorithm). Thus, as a rule of thumb, we will select NFFT to be the next power of two from the frame length.

Wait wait - what is “the next power of two”, anyway? You will find answer from the help of the function “nextpow2”. We will adopt that command and choose:

NFFT = 2^nextpow2(framelen_samples);

Finally, we note that the so-called Hamming-window is the most commonly used window function in speech and audio processing. This can be produced in with the command “hamming” which needs the window length in samples as its input argument. To see how the Hamming window looks like, type:

plot(hamming(framelen_samples));

Now, we are ready to view the spectrogram with,
spectrogram(s,hamming(framelen_samples),noverlap,NFFT,fs);

(Sidenote: Usually spectrograms are plotted so that the time increases from left to right, and frequency increases from down to up. The Matlab spectrogram plotting seems bit non-standard in this sense, time goes from down to up and frequency from left to right).

Spectrogram as a data structure: a complex-valued matrix

In the spectrogram, the color indicates the energy. You can add the legend with the command “colorbar” which shows how the color values are mapped between different energy values. When plotting a spectrogram, the values are usually shown in logarithmic scale to match better human perception.

Note that if you give an output argument to spectrogram, you will get the result into a matrix:

SG=spectrogram(s,hamming(framelen_samples),noverlap,NFFT,fs);

If you type “whos”, you will see that “SG” is a complex-valued matrix of size NFFT x NFRAMES, where NFRAMES is the number of frames. In other words, the Fourier spectra of different frames are stored as column vectors. Let’s take out the spectrum of a randomly selected frame as an example:

spectrum = SG(:,178);

which means that we pick the column number 178 from the matrix. The “:” operator means simply “all” – select all the rows such that the column index is 178. Note that the spectra are complex-valued, this is by definition of the Fourier spectrum. What the spectrogram command plots is the magnitude spectrum of each frame. In order to get the magnitude spectrum, we need to take the absolute value (modulus) of the complex variable:

magnitudespectrum = abs(spectrum); Let’s plot it. You can plot it simply like,

plot(magnitudespectrum);

Or, if you want to use the decibel (dB) scale, then use:

plot(20*log10(magnitudespectrum+eps));

Here, a small number (eps, built-in constant in Matlab) is added to spectrum before taking log, just in case the spectrum magnitude would be exactly zero at some frequency. Usually we prefer the logarithmic plots since they match better our auditory, as well as visual, perception.

How can I relabel the x-axis to be in Hertz ?

As you can notice from the plot of your magnitude spectrum, the x-axis shows the values from 1 to NFFT/2+1. We would like to show the physical units in Hertz in the x-axis. Recall that according to the sampling theorem, if we sample with sampling rate fs, the highest proper frequency in the signal will be limited to fs/2. Therefore, the point NFFT/2+1 in the spectrum corresponds to physical frequency fs/2. Following the same idea as before, we get the frequency labels as follows:

freqvaluesHz = ((0:NFFT/2)/(NFFT/2+1))*(fs/2); Let's plot in dB scale:
plot(freqvaluesHz, 20*log10(magnitudespectrum+eps));
Let's add axis labels and grid:
xlabel('Frequency [Hz]');
ylabel('Magnitude [dB]'); grid on;

5. American English Phonetics

antimoon.com *Articles and products for serious English learners*

Phonetic alphabets reference

The *IPA* column contains the symbol in the International Phonetic Alphabet, which is the alphabet used in many English dictionaries.
The *ASCII* column shows the corresponding symbol in the Antimoon ASCII Phonetic Alphabet, which can be used to type the pronunciation of words on a computer without the use of special fonts.
For a full description of the alphabets + audio recordings of the sounds, visit **www.antimoon.com/ipa**

vowels

IPA	ASCII	examples
ʌ	^	cup, luck
ɑ:	a:	arm, father
æ	@	cat, black
ə	..	away, cinema
e	e	met, bed
ɜ:	e:	turn, learn
ɪ	i	hit, sitting
i:	i:	see, heat
ɒ	o	hot, rock
ɔ:	o:	call, four
ʊ	u	put, could
u:	u:	blue, food
aɪ	ai	five, eye
aʊ	au	now, out
oʊ əʊ	Ou	go, home
eə	e..	where, air
eɪ	ei	say, eight
ɪə	i..	near, here
ɔɪ	oi	boy, join
ʊə	u..	pure, tourist

consonants

IPA	ASCII	examples
b	b	bad, lab
d	d	did, lady
f	f	find, if
g	g	give, flag
h	h	how, hello
j	j	yes, yellow
k	k	cat, back
l	l	leg, little
m	m	man, lemon
n	n	no, ten
ŋ	N	sing, finger
p	p	pet, map
r	r	red, try
s	s	sun, miss
ʃ	S	she, crash
t	t	tea, getting
tʃ	tS	check, church
θ	th	think, both
ð	TH	this, mother
v	v	voice, five
w	w	wet, window
z	z	zoo, lazy
ʒ	Z	pleasure, vision
dʒ	dZ	just, large

special symbols

IPA	ASCII	meaning
ˈ	'	' is placed before the stressed syllable in a word. For example, the noun *contract* is pronounced /ˈkɒntrækt/, and the verb *to contract* is pronounced /kənˈtrækt/.
ʳ	(r)	/kɑːʳ/ means /kɑːr/ in American English and /kɑː/ in British English.
i	i(:)	/i/ means /iː/ or /ɪ/ or something in between. Examples: *very* /ˈveri/, *ability* /əˈbɪlɪti/, *previous* /ˈpriːviəs/.
ᵊl	.l	/ᵊl/ shows that the consonant /l/ is pronounced as a syllable. This means that there is a short vowel (shorter than the /ə/ sound) before the consonant. Examples: *little* /ˈlɪtᵊl/, *uncle* /ˈʌŋkᵊl/.
ᵊn	.n	/ᵊn/ shows that the consonant /n/ is pronounced as a syllable. Examples: *written* /ˈrɪtᵊn/, *listen* /ˈlɪsᵊn/.

Synthesize the words using Praat software and record your own voice for the particular word and compare the results and separate voiced and unvoiced sounds.

NOTE: The report can be prepared while doing laboratory exercise and submitted after finishing the work.

INLAB REPORT:

Hand in the following:

- Time-domain plot and spectrogram plot of the recorded voice.
- Time-domain plot and spectrogram plot of the synthesized and recorded word and phonemes for at least three words.