Automatic Speech Recognition (2008-09): Lab-session sheet (1) — Signal processing for speech recognition —

Hiroshi Shimodaira January 2009 (*Revision* : 1.1)

This practise is intended to give you some idea about basic techniques for spectral analysis of speech and representation of phonemes in frequency domain.

In this tutorial you will

- synthesise sinusoidal signals.
- analyse synthesised or real speech signals with DFT, Cepstral Analysis, and LPC.
- experience "Spectrogram Reading".

Software tool

"Octave", which is a Matlab like free software package, is mainly used in this exercise rather than using software dedicated to speech analysis such as "wavesurfer" and "xwaves". You can write programs yourself in either online(interactive) or offline(script) manner on Octave.

Octave home:	http://www.octave.org/
Official manual:	http://www.gnu.org/software/octave/docs.html
Tutorials:	http://www.aims.ac.za/wiki/index.php/Octave
Wiki:	http://wiki.octave.org/
online help:	"help" or "doc" (in Octave)

Initialising your computing environment

```
% cd                 ··· This is to make sure that you current directory is your home directory, i.e. "~".
% mkdir asr                                  ··· This will create a directory whose name is "asr".
% cd asr
% source /home/hshimoda/pub-asr/bin/init-t1
```

- If you have the directory `~/asr`, rename either the existing one or the above one to another name. Hereinafter we assume `~/asr` as our work directory, although you can use arbitrary directory for this tutorial.

Running Octave

- To run Octave:

```
% cd Octave
% ./octave-asr          ··· This is a shell script. The actual Octave command name is "octave".
```

- To exit Octave, just enter "quit" or "exit":

```
octave:>> quit
```

1 Signal Synthesis

```
octave:1> source("comm-speech.m")          ··· Load Octave programs developed for this exercise.
octave:2> t = GenTimeSeries;
octave:3> x0 = sin(2 * pi * 1 * t);                                  ··· here '0' is a digit zero
octave:4> PlotSig
octave:5> x0 = sin(2 * pi * 2 * t);
octave:6> PlotSig
octave:7> x0 = sin(2 * pi * 500 * t);
octave:8> PlotSig
octave:9> PlaySig             ··· If there is no sound from your computer, read descriptions bellow.
```

Caution: Do not forget putting a semi-colon, ';', at the end of each line, otherwise all the return values from functions will be displayed on your screen.

Descriptions:

- The first line of the above example tells Octave to read the script file "comm-speech.m", in which several functions such as "GenTimeSeries" and "PlotSig" are defined.
- A sine curve of frequency f [Hz] is mathematically expressed as

  $$x(t) = \sin(2\pi f t)$$

  whose representation in Octave is

  ```
  x = sin(2 * pi * f * t);
  ```

  where "`pi`" is an Octave variable for π, and "`f`" is a user defined variable for frequency.
- The second line generates a sequence of time instances "`t`" for the duration of 1 second at a sampling frequency of 16KHz. Hence, "`t`" is a vector of 16000 elements, ie. length(t) is 16000. You can find how to generate such a sequence by taking a look at the Octave script file, "comm-speech.m".
- A function naming rule is used in this exercise to distinguish user-defined functions from Octave original ones. Every user-defined function name begins with a capital letter here. A list of user-defined functions is available in section 4.2.1.
- Some of the user-defined functions accept variable number of arguments. In which case, default values are used if no arguments are specified.
- PlaySig() plays back a signal. If there is no audio from your computer, there would be two reasons for this. One is that your computer might have no audio speaker inside, but probably an audio head-phone jack available instead. Another possibility is that the audio volume is turned down. If this is the case, open a software panel for audio-volume control on your desktop.

2 Spectral analysis based on DFT

2.1 Analysing the synthesised signals

```
octave:1> source("dft-demo.m")
octave:2> SetFrame
octave:3> RunDFT
```

Descriptions

- RunDFT carries out DFT(FFT)-based spectrum analysis on the frame which was set by SetFrame, and shows two different power spectra; one is obtained with a rectangular time window, and the other with a *hanning window*.

2.2 Analysing real speech signals

```
octave:> ReadSig
octave:> PlotSig
octave:> PlaySig
octave:> SetFrame
octave:> RunDFT
octave:> PlaySig( DupSig(x, 5) )
octave:> RunDFTs
```

Descriptions

- ReadSig reads the speech data file "../data/sa1.wav" as default. The data is from the TIMIT speech corpus, and its phone transcription is available as "../data/sa1.phn".

2.3 Things to try

- Try different window lengths. Window length can be specified as the second argument of SetFrame (defulat width is 400).
- Try different window positions in the speech. Window position can be set as the first argument of SetFrame. (See section 4.2.1.)

3 Spectral analysis based on Cepstral Analysis

3.1 Analysing speech signals

```
octave:> source("cepstrum-demo.m");
octave:> ReadSig
octave:> PlotSig
octave:> SetFrame
octave:> RunCep
```

Descriptions

- Change the size of the window "Figure 3" into "portrait" i.e. make the window height longer than the width to have a better view of each graph.
- The graph shown at the top of the window of "Figure 3" shows log-magnitude power spectrum for the given analysis frame. The second top graph displays *cepstrum* (cepstral coefficients) and a *lifter* used for reconstructing the spectrum envelope shown in the third graph. The horizontal axis of the cepstrum is called "*Quefrency*".

3.2 Things to try

- Try "RunCep(n)" where n is a positive integer, which defines the width of the *lifter* for cutting off the higher-order cepstral coefficients. When no argument is given, the function uses a default value of $n = 30$.
- Try "RunCeps", which calls "RunCep" with varying window positions.

4 Spectral analysis based on LPC

4.1 Analyse speech signals

```
octave:> source("lpc-demo.m")
octave:> ReadSig
octave:> SetFrame
octave:> RunLPC
octave:> PlaySig( DupSig(x, 5) )
octave:> PlaySig( DupSig(er, 5) )
```

4.2 Things to try

- Change prediction orders to see how the estimated spectra and residual signals (er) change. The prediction order is specified as the first argument of the function `RunLPC()`. Compare with DFT based spectrum.
- Change frame positions and frame widths. (You can also use `RunLPCs()` for this.)

4.2.1 Functions

GenTimeSeries([*duration*, *Fs*])	: generate a time series. e.g. GenTimeSeries(1.5, 16000)
ReadSig([*filename*])	: read a speech data file. e.g. ReadSig("../data/sx2.wav")
PlotSig([*var*])	: plot a speech signal given in the variable. e.g. PlotSig(x0)
PlaySig([*var*])	: playback a speech signal given as the variable. e.g. PlaySig(x0)
SetFrame([*pos, width*])	: set an analysis-frame position and its width. e.g. SetFrame(1500)
RunDFT([*var*])	: apply DFT to the frame data. e.g. RunDFT(x)
RunDFTs([*pos, width*])	: repeat RunDFT with shifting the position to the right.
RunCep([*lifterwidth*, *var*])	: apply cepstral analysis to the frame data with a lifter of the width. e.g. RunCep(25, x)
RunCeps([*pos, width*])	: repeat RunCep with shifting the position to the right.
RunLPC([*order*])	: apply LPC to the frame data with the prediction order. e.g. RunLPC(12)
RunLPCs([*pos, width*])	: repeat RunLPC with shifting the position to the right.
DupSig(*var*, *n*)	: duplicate a variable n times. e.g. DupSig(x, 5)

4.2.2 Variables

x0	: (vector)	speech signal read from a file. x0[1]∼x0[nx0]
x	: (vector)	speech signal of the frame. x[1] ∼ x[frame_width]
er	: (vector)	residual signal by LPC (er[n]=x[n]-x̂[n])
a	: (vector)	LPC coefficients. a[1] ∼ a[Na]
Na	: (scalar)	LPC prediction order

5 Spectrogram reading

Experienced speech scientists can read spectrograms. Before developing a continuous speech recognition system, it's a good idea to have basic knowledge how each phoneme looks like in time-frequency domain.

```
% cd ~/asr/data
% wavesurfer sa1.wav &
```

- If a window "Interpret Raw File As" appears, set the value of the box, "Read Offset (bytes)", to 1024, and click "OK" button.
- On a new window "Choose Configuration", Choose "Demonstration" or "HTK transcription" and click "OK" , then you will have a main WaveSurfer window, in which spectrogram, phoneme transcription, and wave form are displayed. You can change the height of each sub-window by dragging a boundary between sub-windows.
- Investigate how time-spectral features differ even among the same phonemes in various contexts.

5.1 Links

- `http://speech.bme.ogi.edu/tutordemos/SpectrogramReading/spectrogram_reading.html`
- `http://home.cc.umanitoba.ca/~robh/`
- `http://www.speechandhearing.net/lecture/`

6 Further exercises

- Try spectral analysis on more than one frame: two different frames at least, one corresponds to vowel sound, the other corresponds to consonant sound, for example.
- Investigate the effects in respect to window lengths, spectral analysis methods (DFT, cepstral analysis, LPC), and parameters used in the analysis, i.e. lifter width for cepstral analysis, and prediction order of LPC.

Supplemental information

- You can find speech date (wave and label) files in "`~/asr/data`". Files with ".wav" are wave data, and ".phn" are phoneme labels in which you can find start and end points of each phoneme in the corresponding wave file. For example, in "sa1.phn"

  ```
  0  2260  h#
  2260  4070  sh
  ...
  ```

  which tells you that "sh" stars at point 2260 and ends at point 4070 in the wave file "sa1.wav". The sampling frequency used in the wave data is 16kHz.