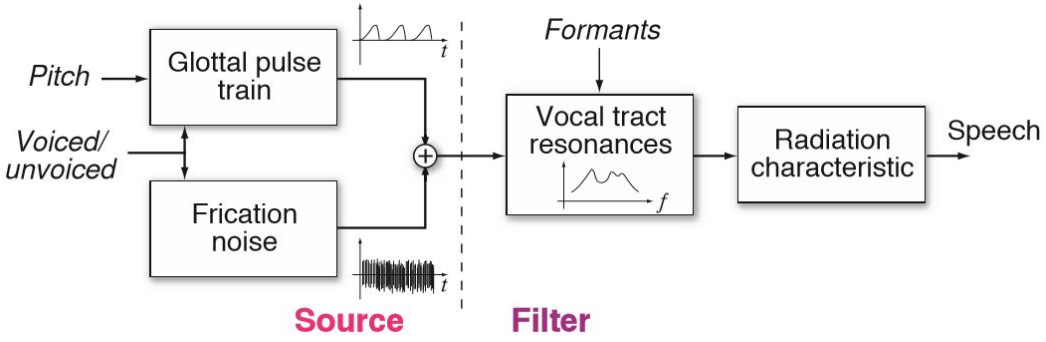
COMP.SGN.120 Introduction to Audio Processing

Exercise 4
Week 47

In this exercise, you will familiarize yourself with the linear predictive analysis for modelling source filter model of speech production and formant estimation, where the *source* is a fine structure in time/frequency and the *filter* is the subsequent shaping by physical resonances. There is a bonus problem worth 0.5 points, which is optional. **The submission should consist of a report of your observations, the figures and the python code with comments**. Alternatively, both can be combined in a Jupyter notebook as well. Refer to the general notes (given in the end) before starting the exercise.



Problem 1: Linear prediction (LP) analysis. (1 point)

Use the loop/function for the frame by frame analysis that you created in previous exercises for LP analysis. The loops divides a signal into overlapping frames, multiplies them with a window function (e.g., *scipy.signal.hann*) and computes the spectrogram. In addition to this now you have to compute LP coefficients for each signal frame and use it to compute the LPC (Linear Predictive Coding) spectrum. You can find audio samples corresponding to some of the vowels sounds to do the analysis. Choosing an appropriate analysis window length is important here, use 20 ms with 50 % overlap (i.e. hop length = 10 ms) between the frames. Choose nfft to be 1024. Now implement the following,

a) For each frame of the signal compute lpc coefficients. You will need *audiolazy* library for this. Install using *pip install audiolazy*. You can use the function *lazy_lpc.lpc.autocor.* In addition, you will compute gain of the residual signal using the given function *est_predictor_gain*.

b) Now once you have the filter coefficients and gain, you can now compute filter transfer function corresponding to the vocal tract using *scipy.signal.freqz* function. Use sampling frequency of the signal under consideration.

Now we will plot some figures to better understand what LP coefficients represent. First choose a representative signal frame, e.g., a frame from the middle of the vowel.

c) Plot the power spectrum of the signal frame in dB scale. In the same figure, plot the filter transfer function corresponding to the vocal tract computed above in dB scale which also represents your LPC spectral envelope.

d) Now experiment with the filter order, at least try three different values, where the differences are observable. Comment on the connection between lpc model order and peaks in the dft spectrum. Is the best order selection dependent on the vowel or not? Include one plot per vowel and the observations on model order in the report. Show the figures, nicely organized and each with their own title. At the end answer the question.

Problem 2: Formant analysis. (1 point)

Now in the same analysis loop add the following,

a) For each signal frame compute roots of the LP filter (hint *np.roots*). What represent those values?.

b) For each signal compute the first three formant frequencies and save them. Formants are essentially the angle of the roots computed above.

For each separate vowel calculate average formant frequencies (F1-F3). Report the values obtained. You can compare these with indicative values given in here. You can see that model order also plays an important part here. If you plot the formant values within a vowel you'll observe that the first few frames and last few frames are generally outliers. You can refine your formant estimates by using the given function.

Bonus: (0.5 point)

What happens if you apply LPC to the oboe sound? What are the corresponding "formants" ?

General notes:

1) Only submissions in python will be graded. It is advisable to use python framework for audio signal processing not only due to its open-source nature and ready availability of useful libraries but also because it has become a popular choice for prototyping audio focussed machine learning methods in recent years.

2) If you are using TUNI systems to do the exercise, it is advisable to have your own python installation so that you can install any additional libraries e.g., *librosa, sounddevice*. Download

Anaconda (*https://www.anaconda.com/distribution/*) to your own directory and install. Any additional libraries can then be installed using conda install, e.g. for *librosa* and *sounddevice*,

conda install -c conda-forge librosa
conda install -c conda-forge python-sounddevice
(-c flag here refers to the channel the library is being downloaded from)

Please ask the teaching assistant for help if you need assistance with this.

3) There may be implementation differences between libraries when it comes to reading a .wav file. For example, *librosa* supports floating-point values and rescales the input audio to [-1, 1] while *scipy* does not do that. To avoid confusion it is advisable to use *librosa* as the main library for audio manipulation.

4) If you are using *librosa* to read audio, ensure that you put sampling rate to *None* to avoid resampling the audio to 22050 Hz which is the default behaviour in *librosa,* i.e.,

librosa .load('audio.wav', sr=None)

5) A periodic hann window can be generated, e.g., using *scipy.signal.hann(winlength, sym=False)*. While choosing a window for analysis and synthesis processing, the chosen window should satisfy constant overlap-add (COLA) condition.

6) Audiolazy library functions for computing LPC coefficients. It can be installed by using pip
pip install audiolazy

More info:

- Formant frequencies (mini-) lesson video 5.30 min long.
- Formants explained and demonstrated video 8.22 min long.