

Speech Signal Analysis

Enno Hermann

Contents

| | | |
|----------|---------------------------------------------------------------------------------|-----------|
| 1 | Speech Signal Observation | 2 |
| 2 | Observation of the Short-Time Speech Signal and Manual Pitch Computation | 3 |
| 3 | Autocorrelation Analysis | 4 |
| 4 | Fourier Spectrum | 5 |
| 4.1 | Windowed Speech Analysis | 6 |
| 5 | Spectrogram | 8 |
| 5.1 | Wide-band Spectrogram | 8 |
| 5.2 | Narrow-band Spectrogram | 9 |
| 6 | Linear Prediction (LP) Analysis | 9 |
| 6.1 | LP Spectrum | 9 |
| 6.2 | LP Residual | 10 |
| 7 | LP Spectrum of Different Speech Sounds | 12 |
| 7.1 | /a/ | 12 |
| 7.2 | /e/ | 13 |
| 7.3 | /i/ | 14 |
| 7.4 | /o/ | 15 |
| 7.5 | /u/ | 16 |
| 8 | Intra- and Inter-Speaker Variability | 17 |
| 8.1 | Intra-Speaker Variability | 17 |
| 8.2 | Inter-Speaker Variability | 18 |
| 9 | SIFT Algorithm and Pitch Contour | 19 |

Introduction

Speech is produced by the excitation of the time-varying vocal tract system by a time-varying source (vibrations of vocal cords). The excitation is generated by air flow from the lungs carried by the trachea through the vocal cords. As the acoustic wave passes through the vocal tract, its frequency content (spectrum) is altered by the resonances of the vocal tract. Vocal tract resonances are called formants. Thus, the vocal tract shape can be estimated from the spectral shape (e.g. formant location and spectral tilt) of the speech signal. The speech produced is an acoustic wave that is recorded, sampled, quantized and stored on the computer as a sequence of numbers (signal). The speech signal can't be used directly, as the information is in the sequence of the numbers. So the speech signal has to be processed and then features relevant to the task have to be extracted. The extracted features may be related to the voice

source, i.e. vocal cords, like pitch frequency, pitch frequency contour etc. or the vocal tract system, like linear prediction parameters, cepstral etc. In this laboratory session, we are going to learn about speech signal processing and extraction of features related to voice source and vocal-tract system.

We will conduct the following experiments:

1. In a 2 second speech signal, you will observe that there is more energy in speech regions than in non-speech regions.
2. The speech signal is non-stationary in nature, so it is processed as a short-time signal where it is quasi-stationary. In the second experiment, we will select a short-time speech signal and estimate the pitch frequency manually.
3. We will observe the autocorrelation of the short-time signal and compute the pitch frequency from it.
4. We will estimate the Fourier spectrum of the short-time signal and also study the effect of windowing.
5. We will study the spectrogram of the speech signal observed in the first experiment.
6. We will learn about linear prediction analysis and study the vocal-tract response, like formants and voice source feature like pitch frequency.
7. We will perform linear prediction analysis on different speech sound signals and observe that they have distinct features.
8. Although the features are distinctive in nature they can vary, which makes tasks such as speech recognition or speaker recognition difficult. Thus we will study variability introduced by speakers.
9. Finally, we will study the pitch contour and the information embedded in it.

The sampling frequency sf of all speech signals is 16000 Hz. Every file name contains information about the gender, speaker, trial number and the sound, for example the speech signal file `f_s1_t1_a` means it is the vowel /a/ spoken by female (`f`), (for male its `m` and child `c`) speaker 1 (`s1`) and trial number 1 (`t1`).

```
utterance = "data/f_s1_t1_a"
```

If you have problems in using any of the functions, use `help` to know the usage, for example

```
help(speech_signal_observation)
```

will give the usage of the function `speech_signal_observation`.

Note: The speech files are stored in ascii format so kindly don't edit or tamper with them. In all the experiments, the usage of the function is explained and then an example usage is given. Follow the example usage for now.

1 Speech Signal Observation

Plot the 2 second speech utterance using the `speech_signal_observation` function and observe the envelope of the signal. The usage of the function is:

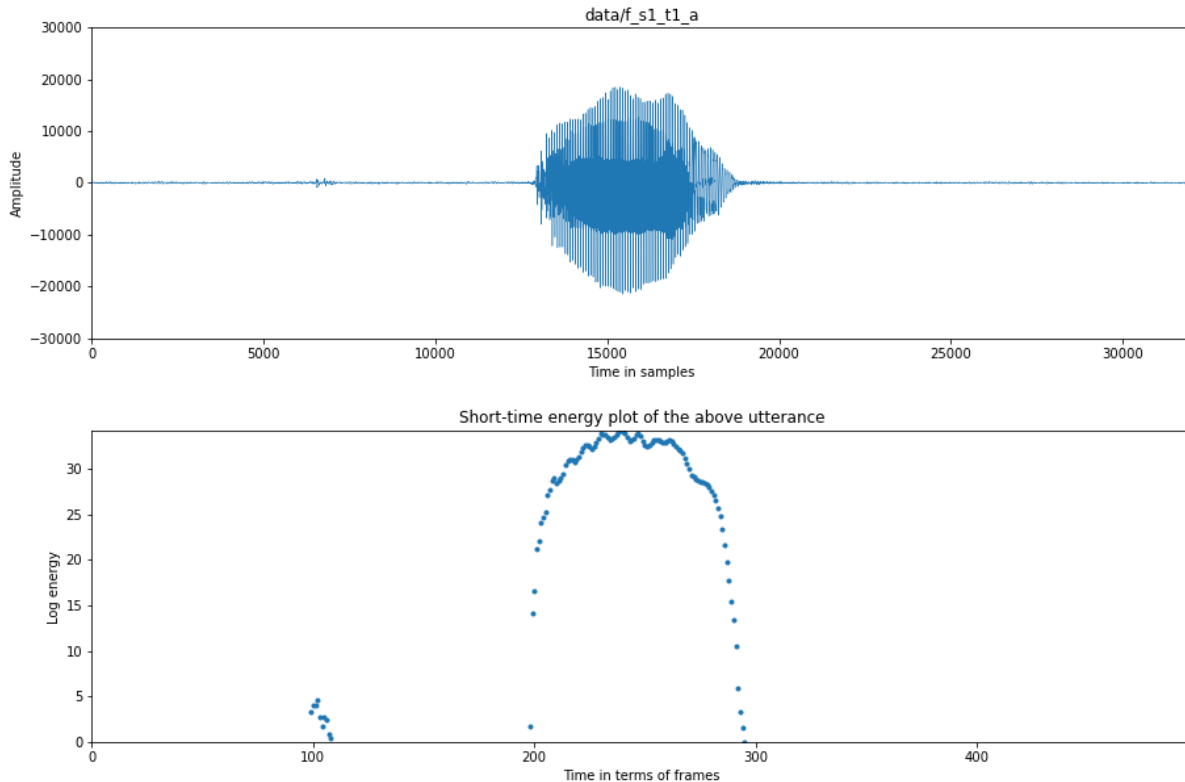
```
from speech_analysis import *  
help(speech_signal_observation)
```

Help on function `speech_signal_observation` in module `speech_analysis`:

```
speech_signal_observation(filename: str, title: str = 'Speech signal') -> numpy.ndarray
    Load a speech signal from the given file and plot amplitude and log energy.
```

This plots the speech signal specified with `filename` with an optional title and returns the speech signal array, for example:

```
data = speech_signal_observation(utterance, title=utterance)
```



The figure shows the speech utterance plotted in the upper part of the figure and the short-time energy plotted below, which is the envelope of the speech signal.

2 Observation of the Short-Time Speech Signal and Manual Pitch Computation

The speech signal is non-stationary in nature but it can be assumed to be quasi-stationary for one to three pitch periods (short-time signal). In this experiment, we are going to observe a short-time speech signal. Select a 30 ms window from the 2 second speech signal observed in experiment 1 by using the `select_speech()` function:

```
help(select_speech)
```

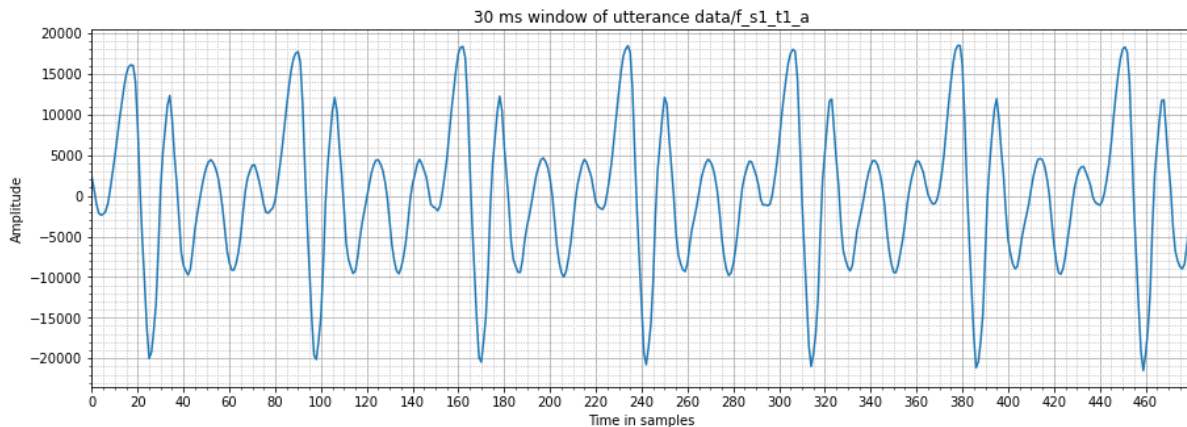
Help on function `select_speech` in module `speech_analysis`:

```
select_speech(data: numpy.ndarray, begin: int, end: int, title: str = 'Windowed signal') -> numpy.ndarray
    Return a window of the given speech signal and plot it.
```

The data between frame numbers `begin` (inclusive) and `end` (exclusive) is returned.

For example:

```
st_data = select_speech(data, 15000, 15480,
                        "30 ms window of utterance " + utterance)
```



Observe the damped sinusoids repeated periodically. Find the period of each sinusoid (neglect the sinusoids which are not complete in the plot) in the following way:

1. Note down the sample number of the largest peak of each sinusoid.
2. Find the number of samples between each of the consecutive peaks. It gives the period of each sinusoid.

Average these periods to estimate the pitch period, p_t . Calculate the fundamental frequency or pitch frequency F_0 using the following equation (sf is the sampling frequency)

$$F_0 = \frac{sf}{p_t} \quad (1)$$

3 Autocorrelation Analysis

In this experiment, we compute the autocorrelation of the short-time speech signal obtained from the Experiment 2 using the `autocorrelation()` function:

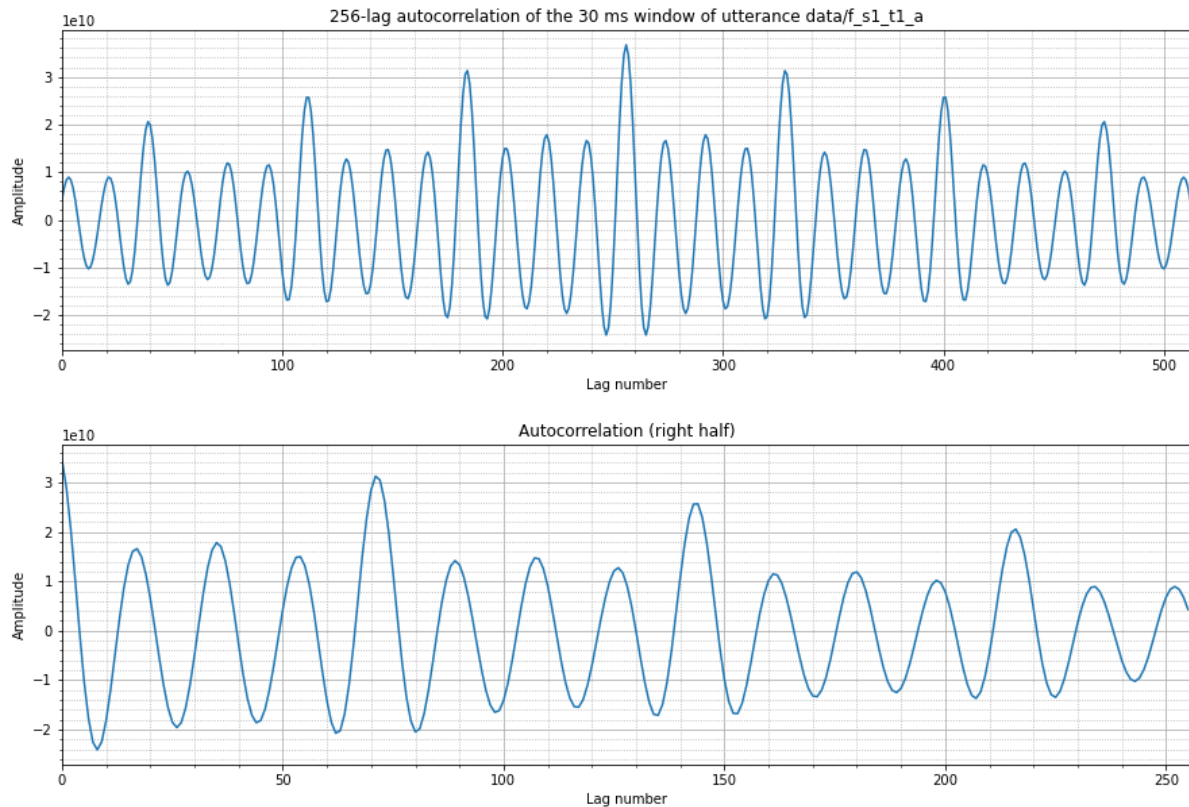
```
help(autocorrelation)
```

Help on function autocorrelation in module speech_analysis:

```
autocorrelation(data: numpy.ndarray, max_lags: int, title: str = 'Autocorrelation', plot: bool = True)
    Return and plot the autocorrelation of the given signal.
```

For example:

```
corr_data = autocorrelation(  
    st_data, 256,  
    "256-lag autocorrelation of the 30 ms window of utterance " + utterance  
)
```



The length of the autocorrelation array is $lag + lag + 1$ which is symmetric to the point $lag + 1$ (for the above example it is 257). The value at this point is the energy of the short-time signal for which the autocorrelation was computed. The upper plot shows the actual autocorrelation (observe the symmetricity) and the plot below shows the right-half symmetry (i.e. from $lag + 1$ to $lag + lag + 1$). Find the second peak in this plot and note down the lag number, it is the pitch period p_t . Use equation 1 to find the fundamental frequency F_0 . Compare it with the F_0 obtained in the previous experiment.

4 Fourier Spectrum

In this experiment, we compute the Fourier spectrum of the short-time signal `st_data` obtained in Experiment 2 using the `fourier_spectrum()` function:

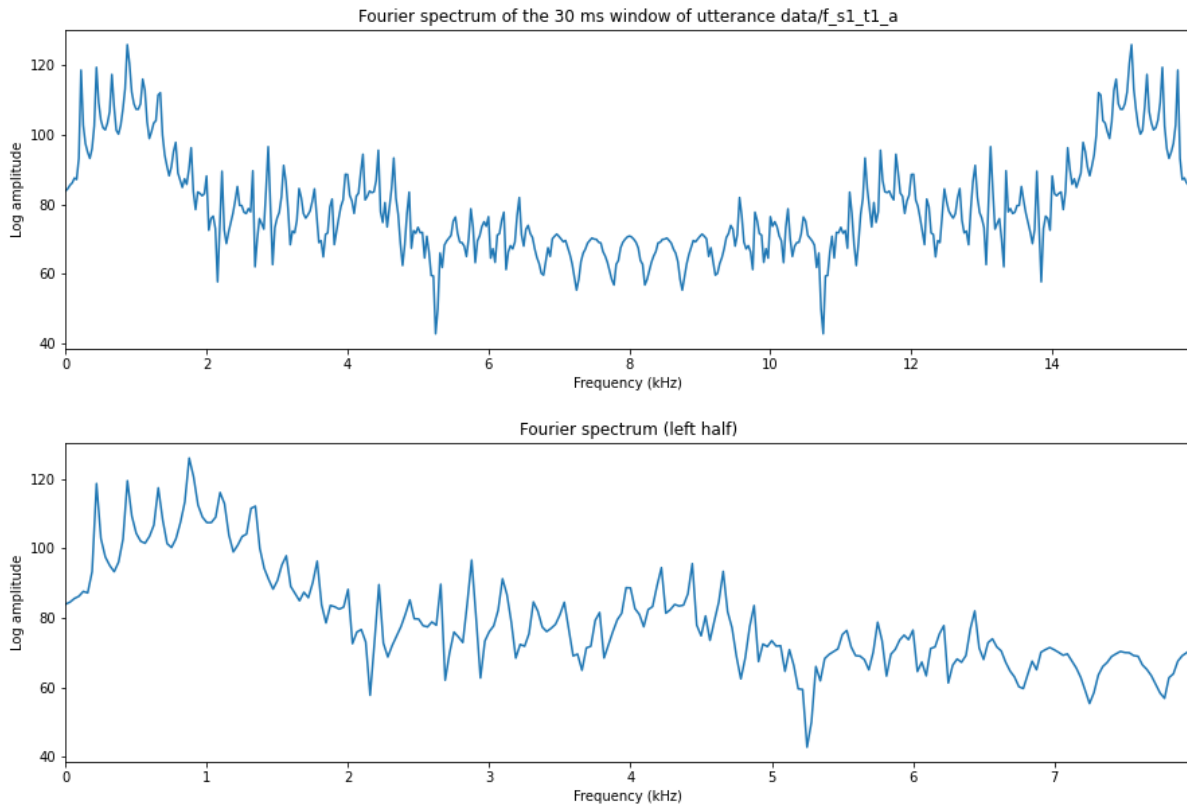
```
help(fourier_spectrum)
```

Help on function `fourier_spectrum` in module `speech_analysis`:

```
fourier_spectrum(data: numpy.ndarray, order: int = 512, sf: int = 16000, title: str = 'Fourier spec  
    Computes and plots the fourier spectrum of the given signal.
```

It computes the DFT of order `order` of the short-time signal `data`. The order of the DFT is generally chosen such that it is a 2^n value to take advantage of the FFT routine. Depending upon the number of samples, select the order of FFT which is near to it, for example the 30 ms window we are using has 480 samples, so we select an order of 512:

```
fourier_spectrum(
    st_data, 512,
    title="Fourier spectrum of the 30 ms window of utterance " + utterance
)
```

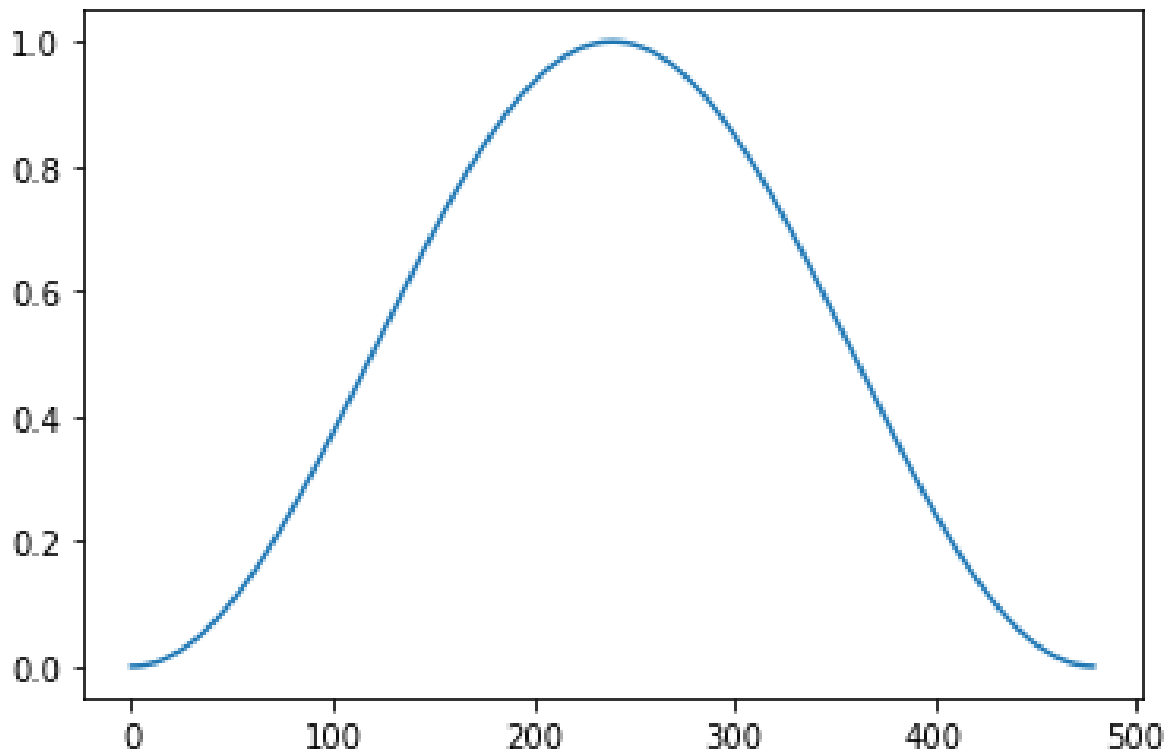


The upper plot shows the 512-point DFT spectrum (observe the symmetry) and the plot below shows the left symmetry of the plot (from point 1 to 256). Observe the spectral peaks, which are the formants (resonances in the vocal tract). The 512-point range covers the entire sampling frequency range, i.e. 16000 Hz, which has redundant information, whereas the plot below covers half of the sampling frequency, i.e. 8000Hz, which is the region of interest (recall the sampling theorem).

4.1 Windowed Speech Analysis

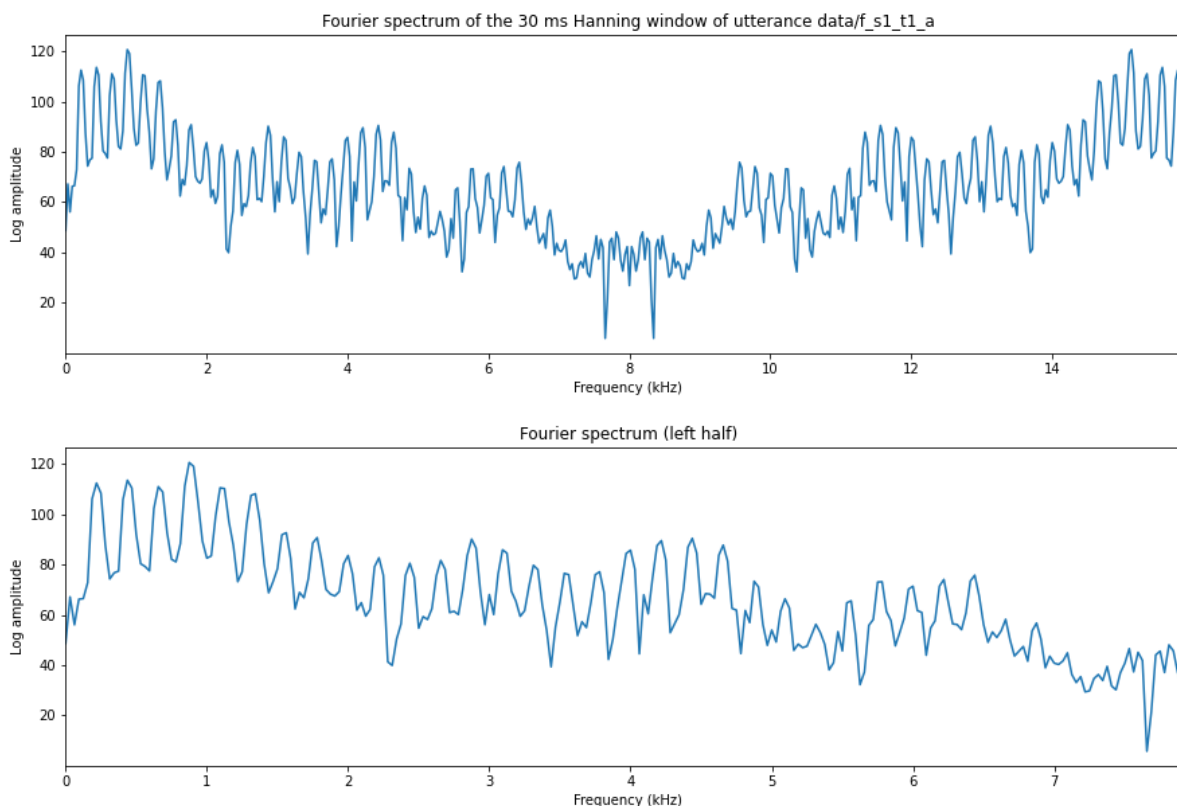
Window the short-time speech signal `st_data` with the Hanning window:

```
import matplotlib.pyplot as plt
hanning_window = np.hanning(len(st_data))
plt.plot(hanning_window)
st_data_hanning = st_data * hanning_window
```



Compute the Fourier spectrum for the windowed short-time signal `st_data_hanning`. Observe the difference in the Fourier spectrum of the signal `st_data` using a rectangular window (which was implicit when we created it in Experiment 2) and the signal `st_data_hanning` using the Hanning window.

```
fourier_spectrum(  
    st_data_hanning, 512,  
    title="Fourier spectrum of the 30 ms Hanning window of utterance " + utterance  
)
```



5 Spectrogram

In this experiment we are going to compute the narrow-band and wide-band spectrogram of the entire utterance i.e. the signal `data` obtained in Experiment 1. Recall that in the wide-band spectrogram we get good time resolution and in the narrow-band spectrogram we get good frequency resolution. The spectrogram is computed using the `spectrogram()` function:

```
help(spectrogram)
```

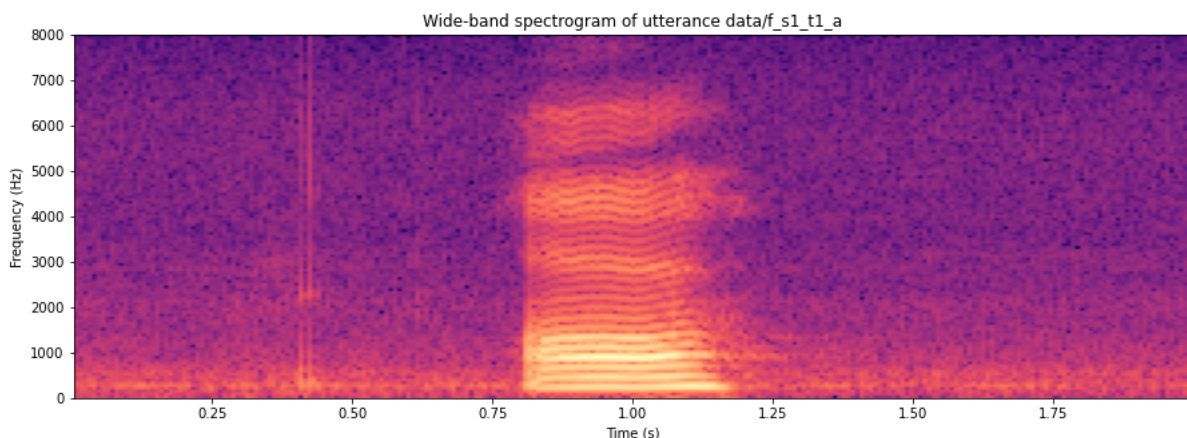
Help on function spectrogram in module speech_analysis:

```
spectrogram(data: numpy.ndarray, order: int, window: Union[Callable[[int], numpy.ndarray], NoneType], NoneType)
    Compute and plot the spectrogram of the given signal.
```

The type of spectrogram depends upon the order `order`. For a wide-band spectrogram we need a small window and choose order 256 or 128, which is a short duration. For a narrow-band spectrogram we choose order 1024 or 2048, which is a long duration, so we loose the time resolution. The function uses the Hanning window internally.

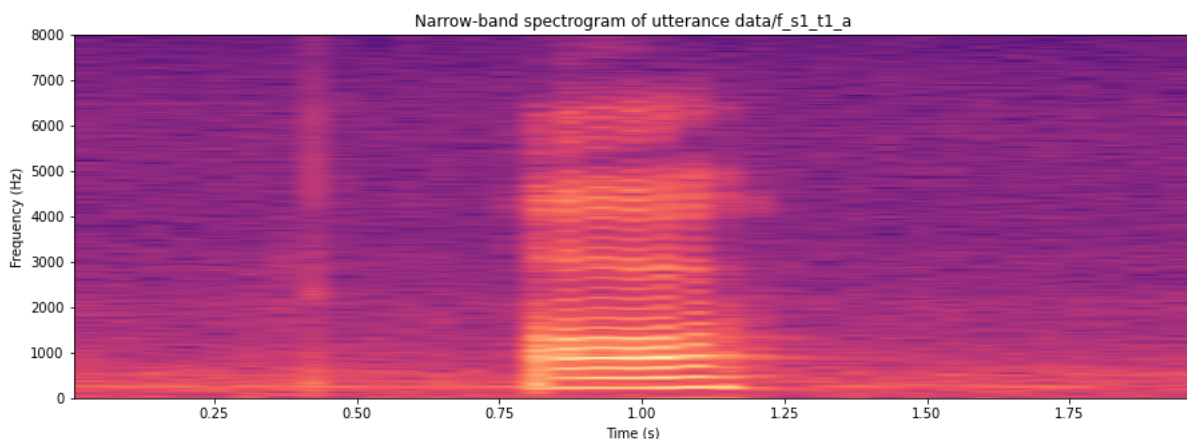
5.1 Wide-band Spectrogram

```
spectrogram(data, 256, title="Wide-band spectrogram of utterance " + utterance)
```

5.2 Narrow-band Spectrogram

```
spectrogram(data, 1024, title="Narrow-band spectrogram of utterance " + utterance)
```



6 Linear Prediction (LP) Analysis

Linear prediction is the most common technique to estimate the shape of the vocal tract. A p -th order linear prediction expresses every sample as the linear weighted sum of the past p samples. The resulting difference equation expressed in the z -domain is

$$H(z) = \frac{1}{1 - \sum_{j=1}^p a_j z^{-j}}$$

The idea behind linear prediction analysis is to estimate the p a_k -s that minimize the mean-square error of the prediction. The linear prediction error is also called LP residual. The a_k -s determine the solution of the equation. The solution of the equation in the denominator is called pole. A real pole determines the spectral roll-off and a complex pole (which always exists with a conjugate) determines the location of the formant in the LP spectrum. The LP spectrum is the Fourier transform of the a_k -s.

6.1 LP Spectrum

In this experiment, we will observe the LP spectrum of the short-time speech signal obtained from Experiment 2 using the function `lp_spectrum()`:

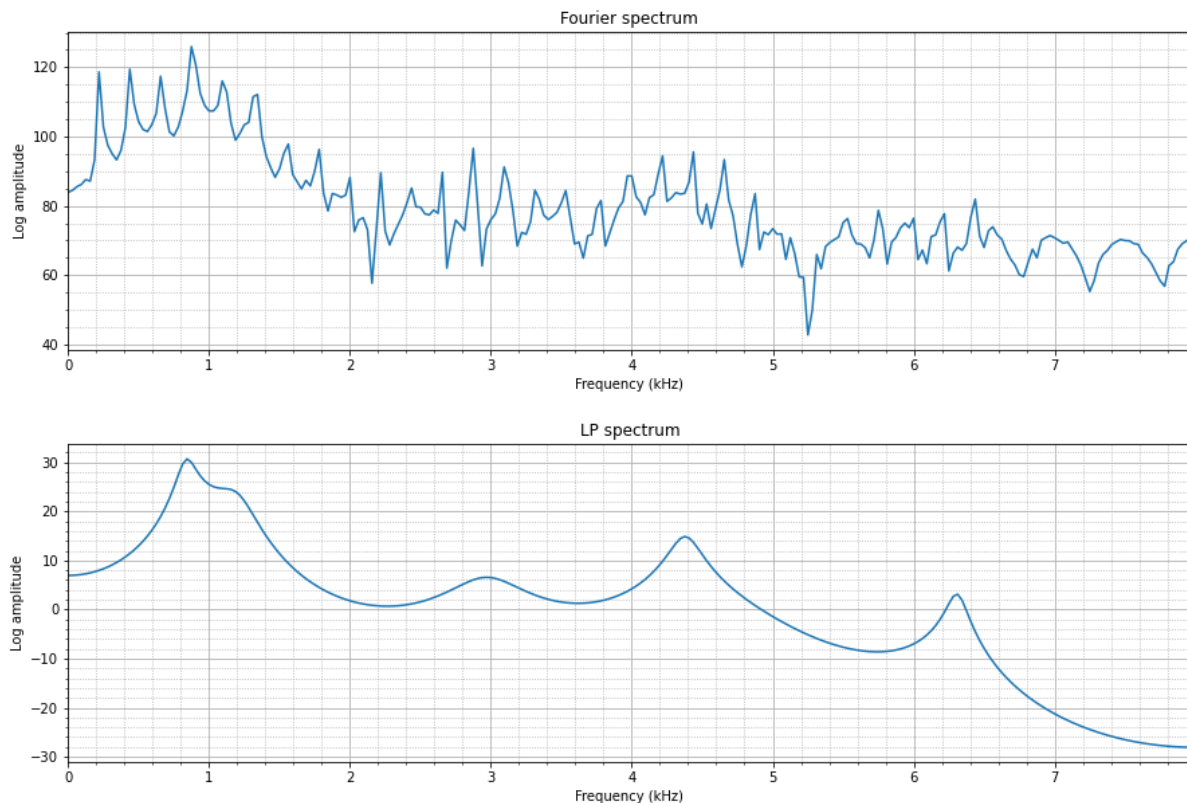
```
help(lp_spectrum)
```

Help on function lp_spectrum in module speech_analysis:

```
lp_spectrum(data: numpy.ndarray, lp_order: int, order: int, window: Union[Callable[[int], numpy.ndarray], numpy.ndarray])  
    Compute and plot the LP spectrum.
```

`lp_order` is the linear prediction order p . The default `window` function is the Hanning window. `order` is the FFT order needed to compute the linear prediction spectrum from the a_k -s. For example:

```
lp_order = 14  
_ = lp_spectrum(st_data, lp_order, 512)
```



In the figure, you will observe two plots. The upper plot is the Fourier spectrum and the lower plot is the linear prediction spectrum. Observe the more prominent spectral peaks (formants) in the linear prediction spectrum compared to the Fourier spectrum. Note down the frequency of each peak, then go back to the wide-band spectrogram from Experiment 5 and observe that the energy is indeed high near that spectral frequency. Now change the linear prediction order `lp_order` to, say 1, 3, 16, 20, 30, 50 and observe the changes in the LP spectrum. Try to reason about it.

6.2 LP Residual

In this experiment we will perform linear prediction analysis and compute the LP residual of the short-time speech signal obtained from Experiment 2 with the `lp_residual()` function:

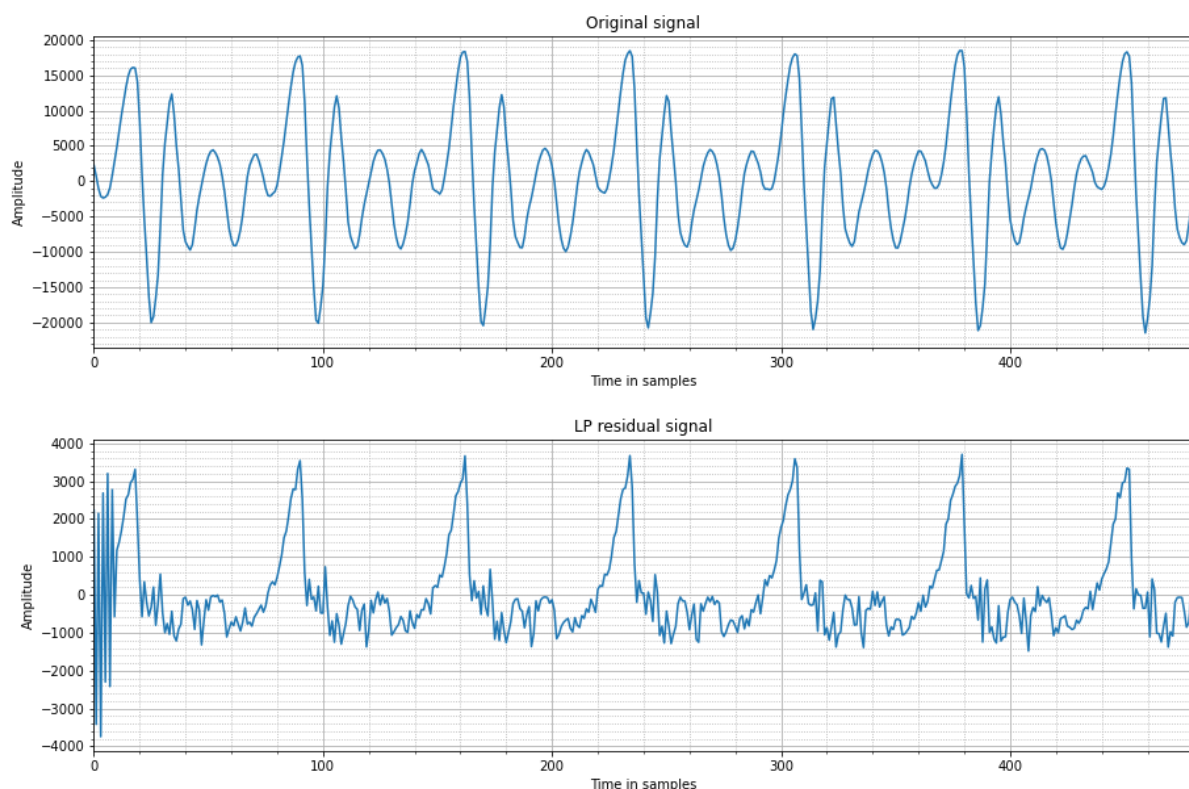
```
help(lp_residual)
```

Help on function lp_residual in module speech_analysis:

```
lp_residual(data: numpy.ndarray, lp_order: int, window: Union[Callable[[int], numpy.ndarray], NoneType])  
    Compute and plot the LP residual.
```

For example:

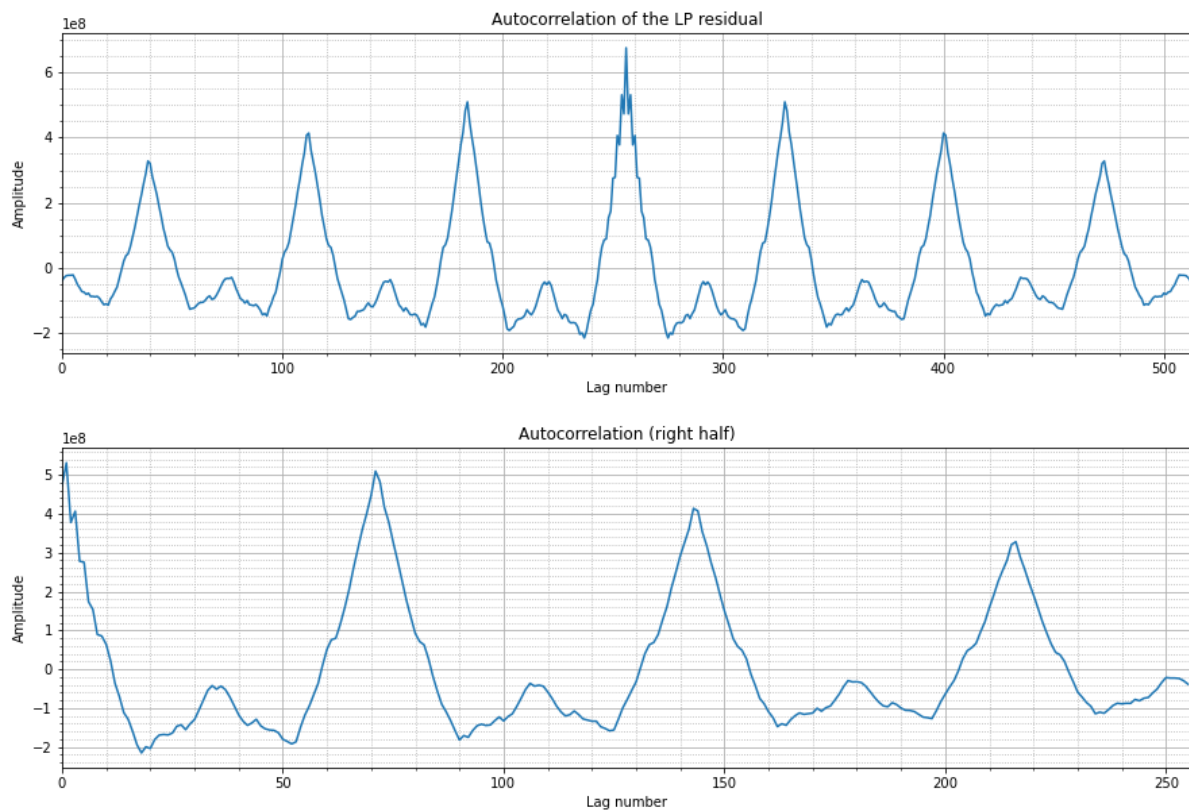
```
lp_order = 10  
residual = lp_residual(st_data, lp_order)
```



1. Note down the sample number of the largest peak of each sinusoid in the upper plot.
2. Note down the sample number of the corresponding peaks in the LP residual.
3. Compare these two observations. Are they the same?

Perform an autocorrelation analysis on the residual signal using the `autocorrelation()` function and find the pitch period as it was done in Experiment 3:

```
_ = autocorrelation(residual, 256, title="Autocorrelation of the LP residual")
```

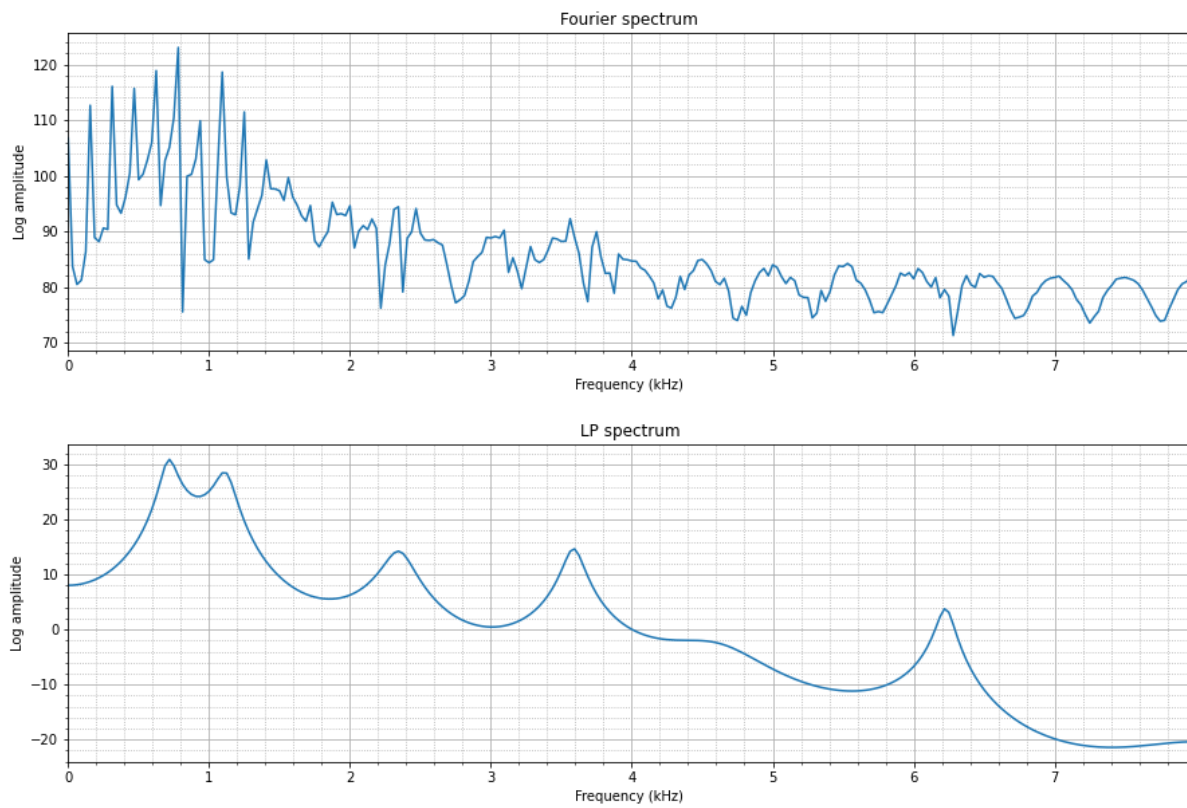


7 LP Spectrum of Different Speech Sounds

In Experiment 6.1, we studied the LP spectrum of a short-time signal. In this experiment, we are going to study the LP spectra of different vowels. Note down your observations.

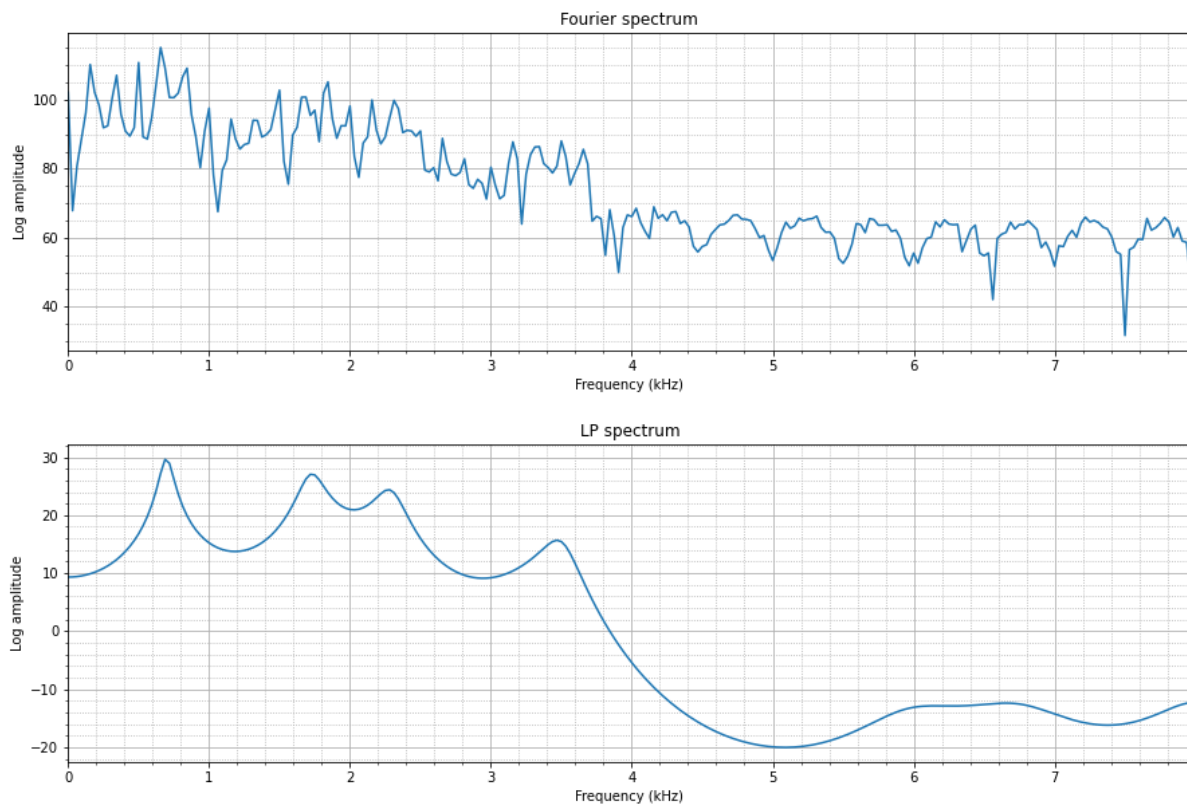
7.1 /a/

```
st_data_a = load_signal("data/m_s2_t1_a")[9000:9480]
_ = lp_spectrum(st_data_a, 16, 512)
```



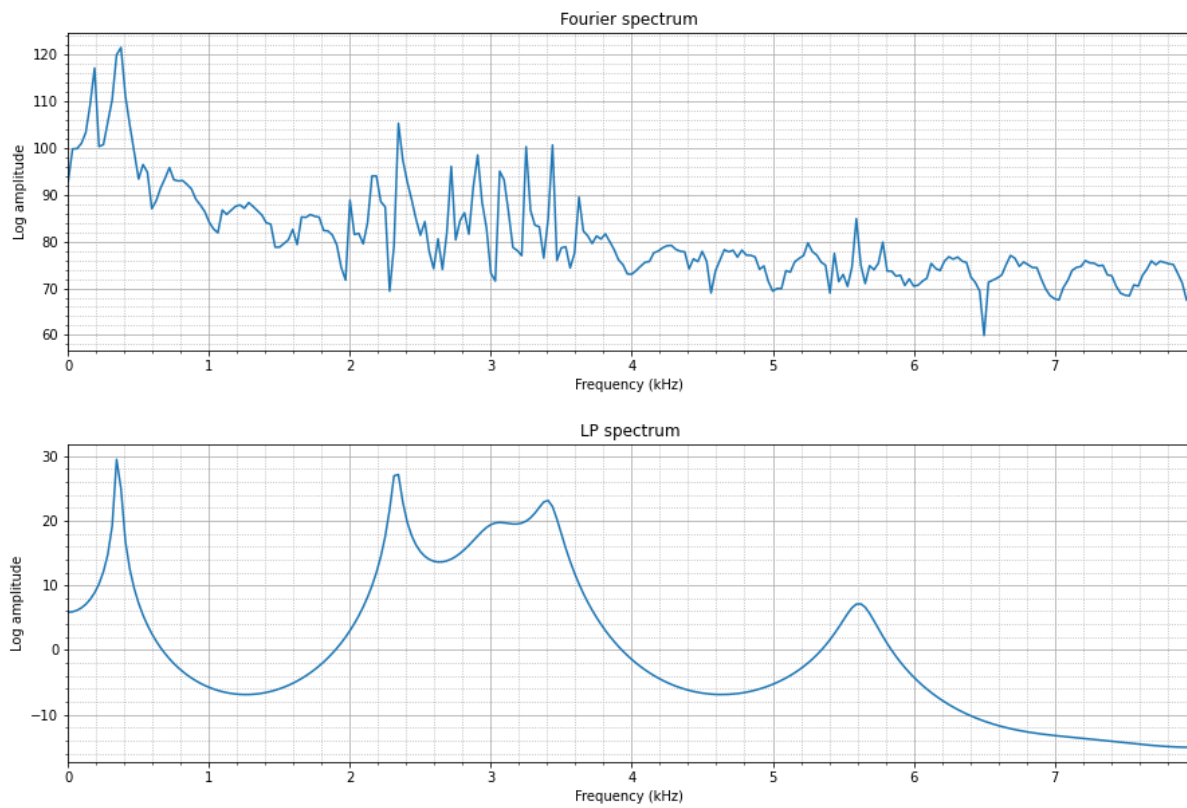
7.2 /e/

```
st_data_e = load_signal("data/m_s2_t1_e")[8000:8480]
_ = lp_spectrum(st_data_e, 16, 512)
```



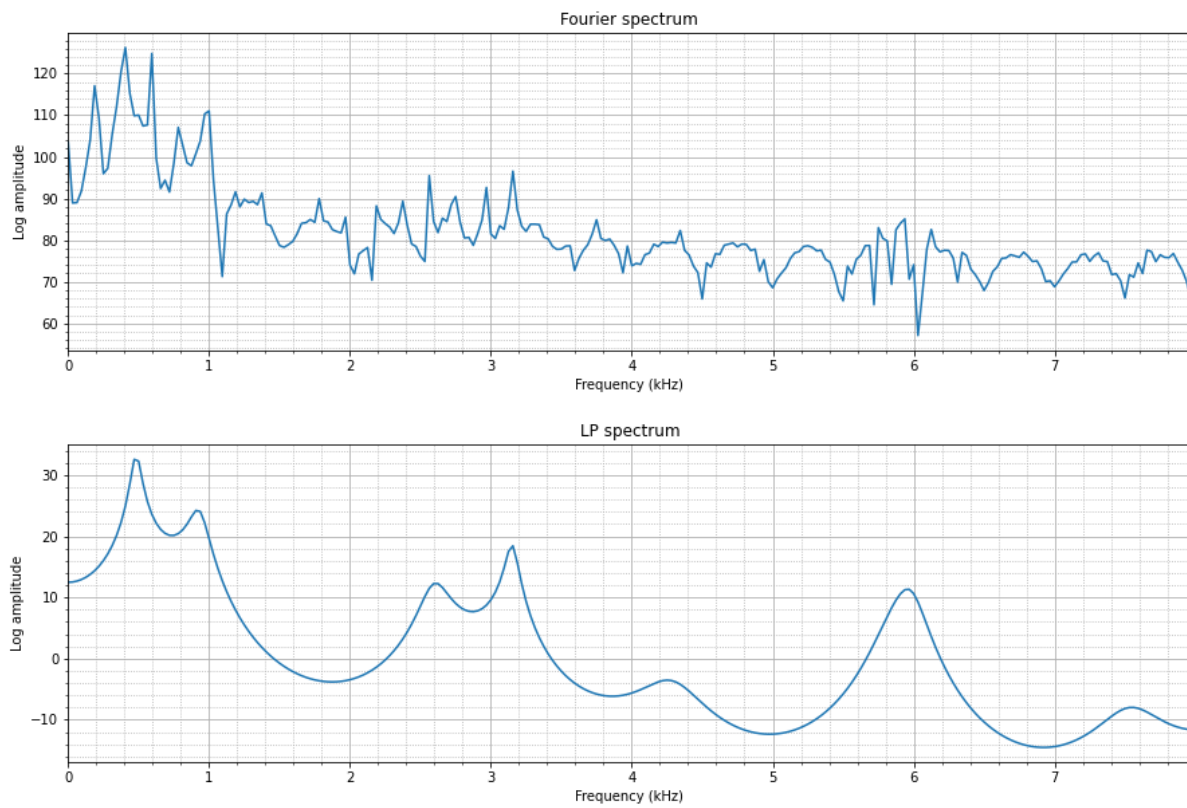
7.3 /i/

```
st_data_i = load_signal("data/m_s2_t1_i")[14000:14480]
_ = lp_spectrum(st_data_i, 14, 512)
```



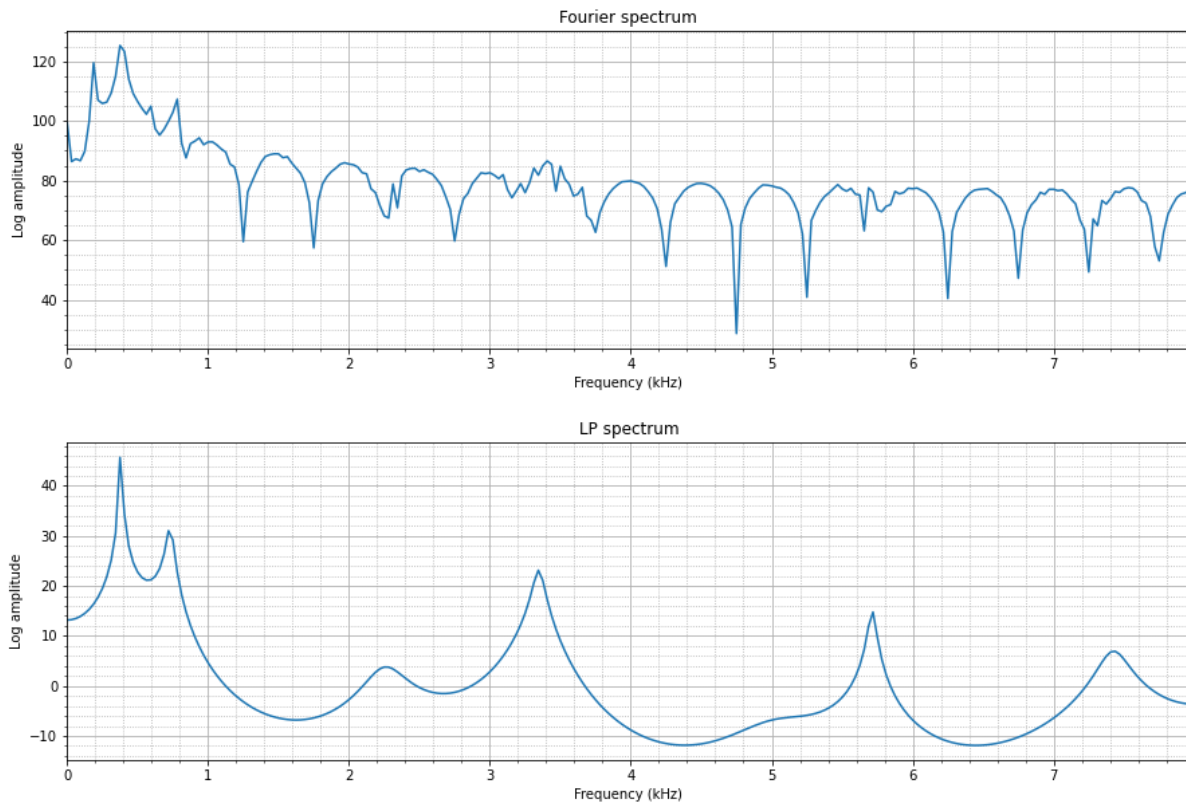
7.4 /o/

```
st_data_o = load_signal("data/m_s2_t1_o")[12000:12480]
_ = lp_spectrum(st_data_o, 18, 512)
```

7.5 /u/

```
st_data_u = load_signal("data/m_s2_t1_u")[17000:17480]
_ = lp_spectrum(st_data_u, 18, 512)
```

8 Intra- and Inter-Speaker Variability

In Experiments 6 and 7, we studied the effect of order on linear prediction and also observed that for different sounds the formants are different. In this experiment, we are going to analyse the variability caused by speakers. There are two kinds of speaker variability that are of interest:

1. **Intra-speaker variability** is the variability introduced by the same speaker while producing the same sound repeatedly.
2. **Inter-speaker variability** is the variability introduced by different speakers producing the same sound.

This can be useful depending upon the type of application, such as in speech recognition it is good if there is no speaker variability, whereas, for speaker recognition inter-speaker variability is very important. Intra-speaker variability is neither useful for speech recognition nor for speaker recognition applications.

8.1 Intra-Speaker Variability

For this experiment we use 3 utterances of the same sound /a/ spoken by the same speaker 3 different times. We will use the `speaker_variation()` function, which takes the utterance file names and their corresponding starting points defining the short-time signal. A length of 480 samples for the short-time signal is assumed by default.

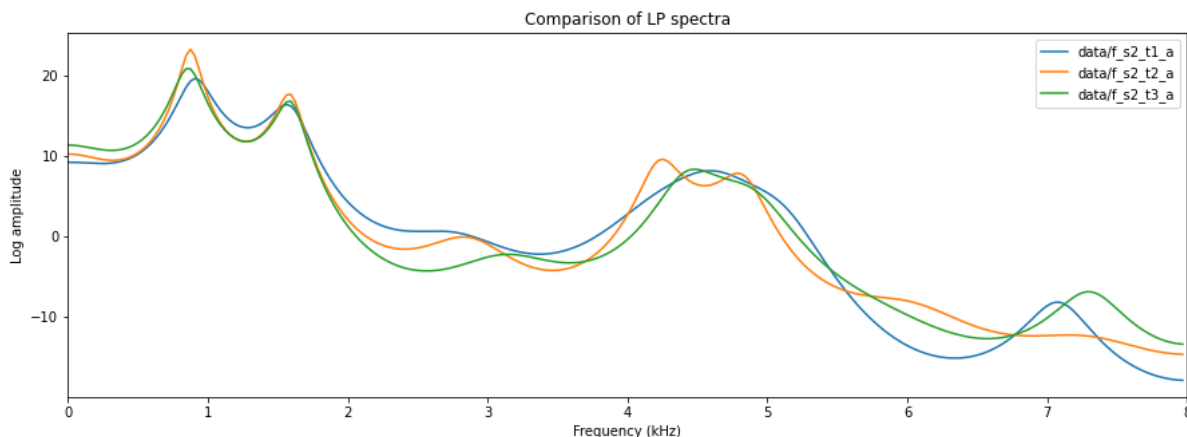
```
help(speaker_variation)
```

Help on function `speaker_variation` in module `speech_analysis`:

`speaker_variation(utterances: List[Tuple[str, int]], lp_order: int = 16, sample_length: int = 480,`
Plots the LP spectra for a list of utterances.

For example:

```
speaker_variation([("data/f_s2_t1_a", 14000),
                  ("data/f_s2_t2_a", 10000),
                  ("data/f_s2_t3_a", 12480)])
```

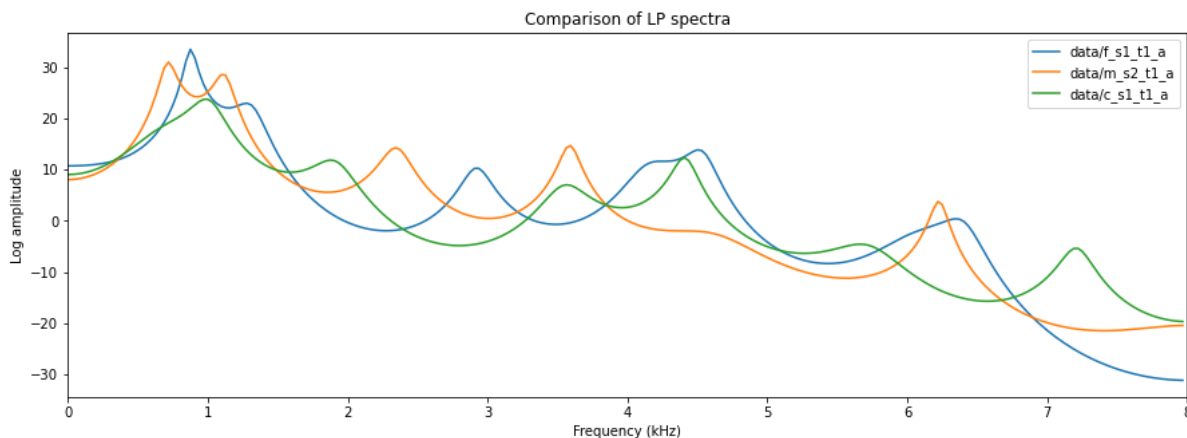


This computes the LP spectrum of the short-time signal of all 3 utterances and plots them in the same figure. Observe that the first two formant regions for all 3 utterances are almost the same, while this is not the case for higher formants.

8.2 Inter-Speaker Variability

Next, we take 3 utterances of the sound /a/ spoken by a female, male and a child.

```
speaker_variation([("data/f_s1_t1_a", 15000),
                  ("data/m_s2_t1_a", 9000),
                  ("data/c_s1_t1_a", 12480)])
```



Again observe that the first two formant regions for the male and female speaker are almost the same. In case of child speech the second formant shifted much more than the first formant. Like in the previous experiment we observe that the higher formant regions are different for different speakers even though the same sound /a/ is being spoken.

9 SIFT Algorithm and Pitch Contour

In this experiment, we extend the idea of pitch estimation using the LP residual (see Experiment 6.2) into a pitch estimation algorithm. The pitch frequency can be estimated through the Simple Inverse Filter Tracking (SIFT) algorithm. It computes the pitch frequency for a given short-time speech signal in the following way:

1. Low-pass-filter the short-time signal.
2. Perform LP analysis and obtain the LP residual.
3. Perform autocorrelation on the LP residual.
4. Find the location of the second peak, make a decision on voicing. If voiced, compute the pitch frequency, else set the pitch frequency to zero.

The pitch frequency contour for a spoken sentence can be computed by taking a short-time window of, for example, size 30 ms:

1. Place this window at the beginning of the speech signal and compute the pitch frequency using the SIFT algorithm.
2. Shift the window by 10 ms and compute the pitch frequency using the SIFT algorithm.
3. Repeat step 2 until the end of the speech signal is reached.

The 10 ms shift is called a **frame**. So we obtain a pitch frequency for every 10 ms or every frame. The pitch contour is nothing but the array of pitch frequencies obtained for the sequence of frames. For applications like speech or speaker recognition, for every frame a feature parameter vector (e.g. LP coefficients) is obtained. In other words, the feature extraction stage yields a sequence of feature parameter vectors $x_1, x_2 \dots x_{N-1}, x_N$, where N is the number of frames.

In this experiment, first, we are going to observe the pitch contour for two different types of sentences, interrogative and declarative, using the `sift()` function:

```
help(sift)
```

Help on function sift in module speech_analysis:

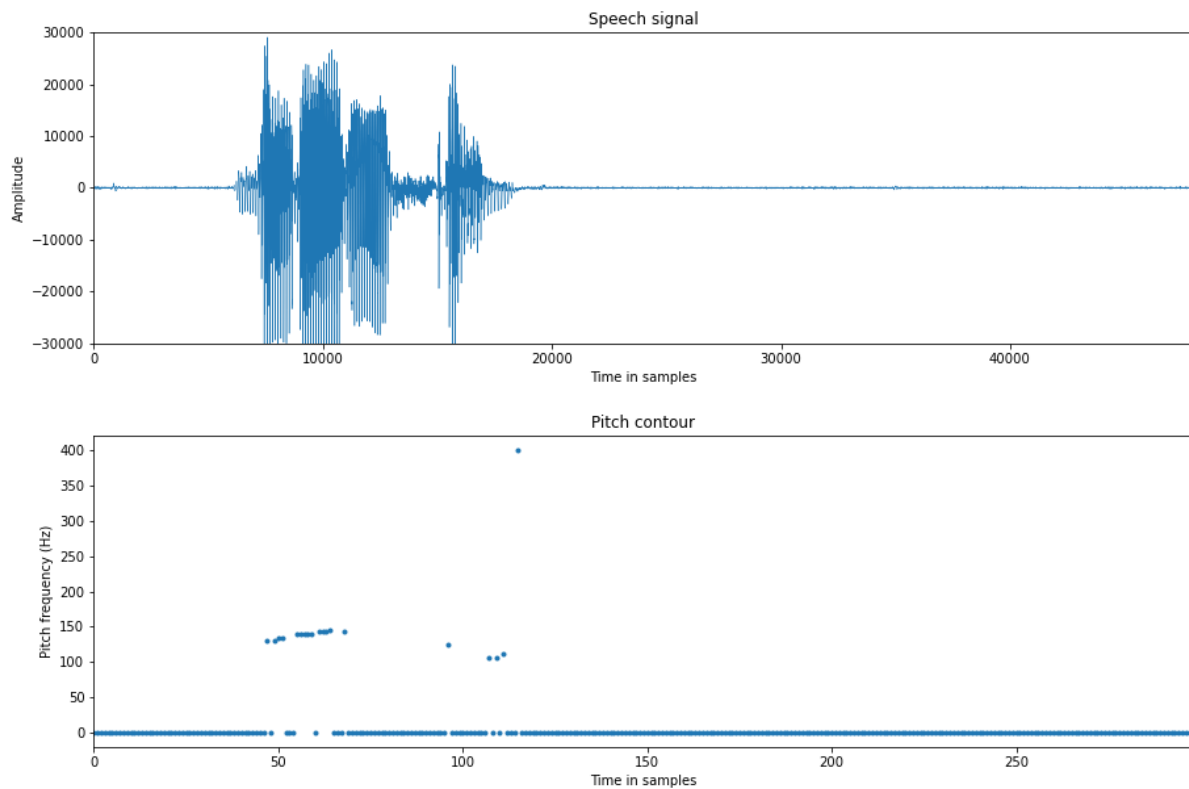
```
sift(filename: str, lp_order: int = 10, frame_size: int = 480, frame_shift: int = 160, sf: int = 16000)
    Compute the pitch contour using the SIFT algorithm.
```

For this study, we will use a 30 ms frame size (480 samples), a shift of 10 ms (160 samples) and a linear prediction order of 10.

```
lp_order = 10
frame_size = 480
frame_shift = 160
```

The sentence spoken is an interrogative sentence, *"Where are you from?"*:

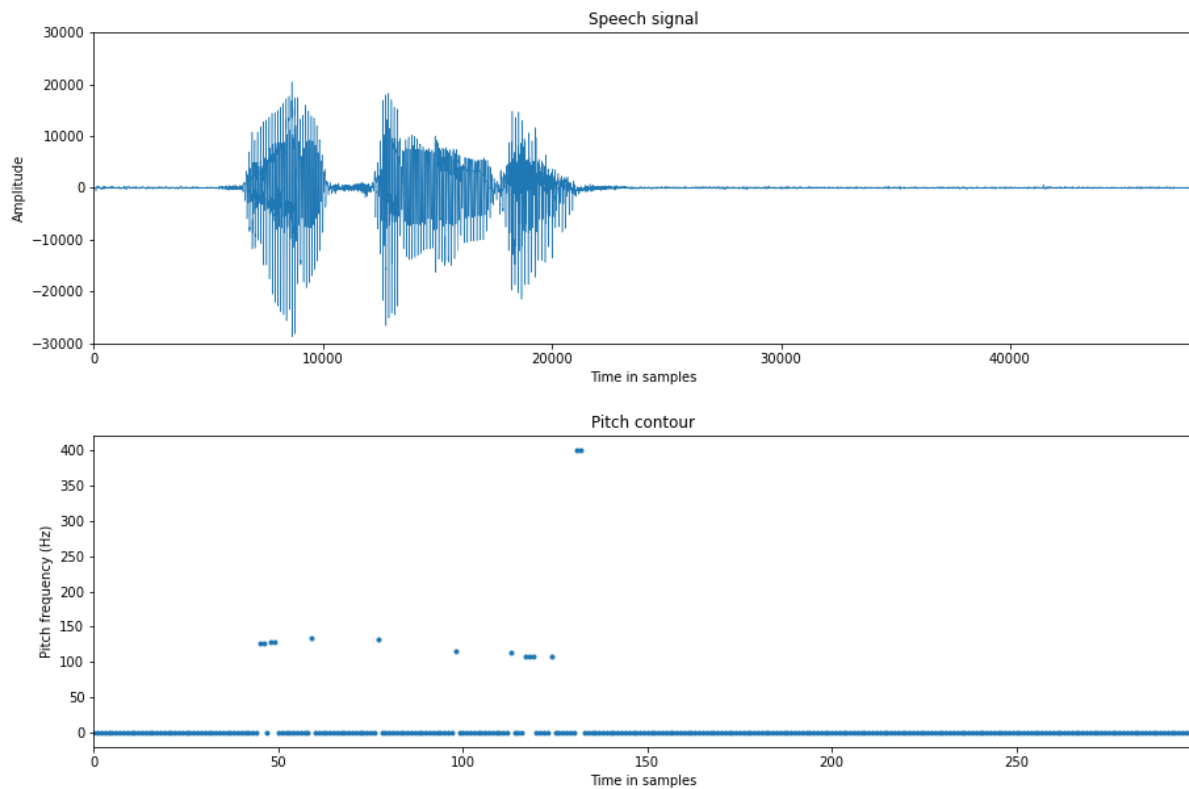
```
_ = sift("data/m_s1_i_sen1", lp_order, frame_size, frame_shift)
```



In this plot you see the speech signal and its pitch contour below. Observe the rise and fall of the pitch contour across the sentence. This rise and fall of pitch contour carries information like speaking style, type of sentence, emotional status of the speaker etc. Observe the rise of the pitch contour for the word *where* at the beginning of the sentence (in the context of interrogation). If a line is drawn interpolating the peaks and valleys in the pitch contour, it will have a positive slope. Observe at the end again a fall and then a rise of the pitch contour.

The next sentence is a declarative sentence, "*I am from India*":

```
_ = sift("data/m_s1_d_sen1", lp_order, frame_size, frame_shift)
```

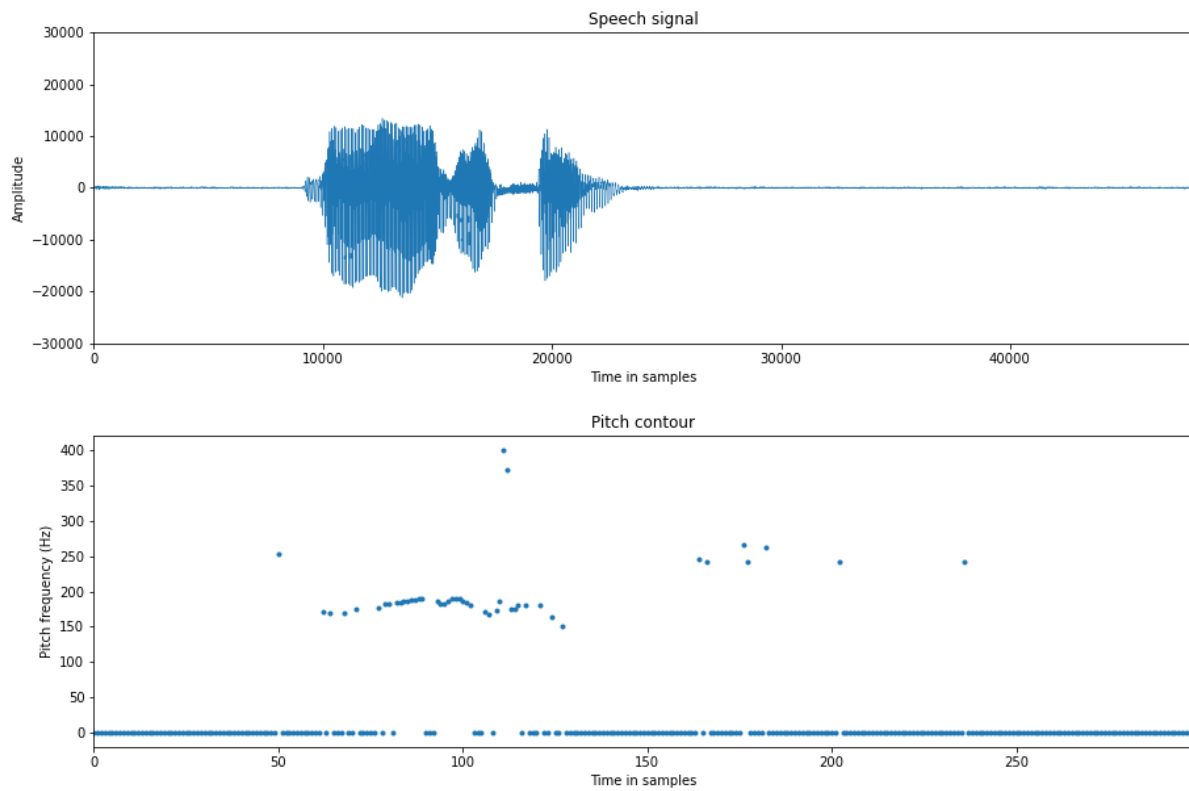


Observe the rise and fall of pitch contour across the sentence. If a line is drawn interpolating the peaks and valleys in this pitch contour, it will have a negative slope.

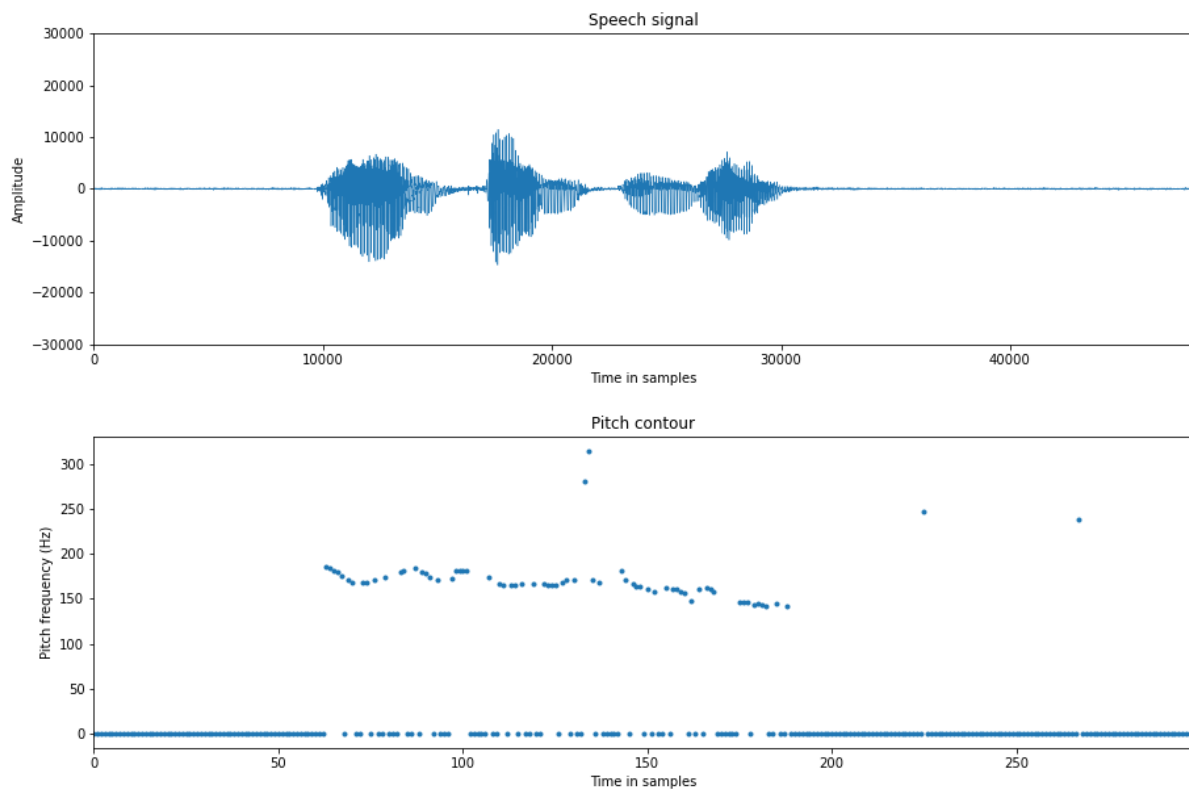
Note that the pitch frequency for a single frame is just an information about the speaker. It doesn't convey any information regarding the sentence being spoken or its message or the emotional status of speaker. But when a longer duration (say 100-300 ms), i.e. a sequence of frames, is considered then we can observe the rise and fall of the pitch contour and derive such information. Still, the pitch contour does not convey any information regarding the message being spoken.

Now we will perform the pitch contour analysis on the same sentences spoken by a different speaker:

```
_ = sift("data/m_s2_i_sen1", lp_order, frame_size, frame_shift)
```



```
_ = sift("data/m_s2_d_sen1", lp_order, frame_size, frame_shift)
```



Compare these pitch contours to those of the previous speaker who spoke the same sentences. Are they different? Human efficiently use the speaking style information which is embedded in the pitch contour to recognize another person.

Acknowledgements

This lab was originally developed in Matlab for the *Speech Processing and Speech Recognition* course at École polytechnique fédérale de Lausanne (EPFL).