Chapter 1

Speech coding – general introduction

2006–2016[1]

1.1 Introduction

Speech is the primary form of communication between humans. For this reason, even nowadays one of the most important applications of the telecommunication networks is the transmission of speech. During this laboratory measurement we will deal with the basics of speech transmission and speech coding. The goal of the laboratory measurement is to introduce speech coding as part of the telecommunication technologies.

The first part of this guide is a brief introduction about human speech and speech production. The importance of this is that modern speech coders make use of the properties of speech, and thus can achieve lower bitrates without significant quality loss. The second part presents the basics of speech coding: building blocks of encoders and decoders, and the evaluation criteria of speech coders. The third part is about linear prediction. Almost every speech coder today (e.g. GSM, 3G or 4G codecs) is based on linear prediction. If you understand the essence of this, than you can have an overview of the principles of most speech coders. The laboratory measurement can be implemented in Matlab / Octave / Python.

1.1.1 Human speech

Speech is an *acoustic wave*, being the vibration of air molecules. In case of acoustic waves, the source of the wave changes the pressure level of the air, and this pressure change spreads in the air as a form of a longitudinal wave. Fig. 1.2 shows the waveform of a section of speech: it is the change of pressure level as a function of time.

[1]Originally written in Hungarian by Tamás Bőhm, Sep 27, 2006.
Translated to English by Tamás Gábor Csapó <csapot@tmit.bme.hu>, Sep 27, 2016.
Last version, Oct 10, 2018.

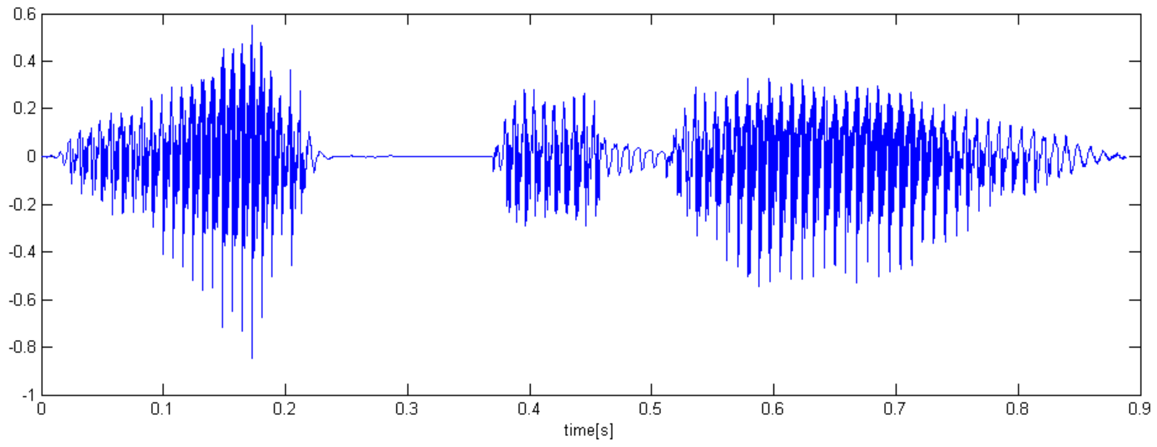

Figure 1.1: Speech waveform of the word *'lábpedál'* (in English: *'foot pedal'*) from a male speaker.

1.1.2 Speech production

Speech production can be split to two main parts: phonation and modulation. During phonation, with the help of the air coming from the lungs, we create a *voice source*. During modulation, we change this voice source and shape the *speech sound*s.

The source can be *voiced*, *noise*, or *combined*. Voice can be produced using the periodic vibration of the vocal folds. We can create the voiced speech sounds by modulating the voiced source. This means that the waveform corresponding to the voiced speech sounds is periodic (see Fig. 1.2). In reality, it is only roughly periodic, because the vibration properties of the vocal folds change continuously. The periods of the speech sounds are called fundamental period, and the reciprocal of the time period is called fundamental frequency $(F_0): F_0 = \frac{1}{T_0}$ where T_0 is the length of the fundamental period. The perceptual correlate of fundamental frequency is the pitch (or intonation contour). The fundamental frequency of males is usually between 80–150 Hz, the F_0 of females is mostly between 170–350 Hz, whereas for children it is between 300–500 Hz.

If there is a constriction in the way of the air during speech production, than unvoiced source will be produced. An example for this is the 's' sound (constriction between the dentils). We can also produce unvoiced speech sounds if we close the way of the air and there will be a burst caused by the increased pressure behind the closure. An example is the 'p' sound (closure between the lips). The speech sounds that are produced without periodic vibration, just by noise, are called unvoiced sounds. The noise is an aperiodic wave, therefore it can be clearly distinguished from the voiced sounds on the speech waveforms (see Fig. 1.2).

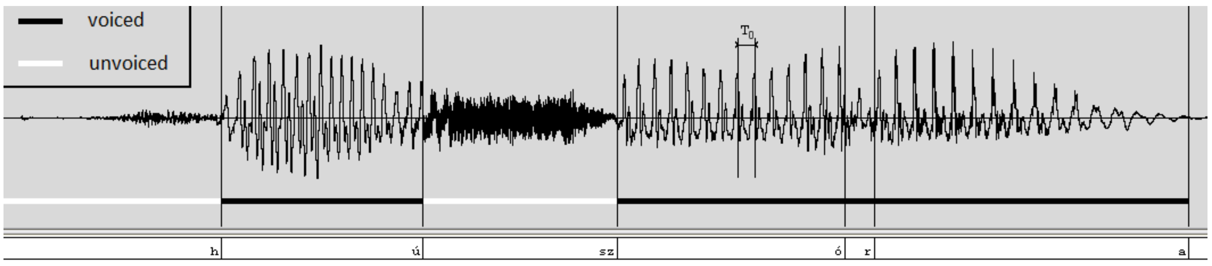

Figure 1.2: The sentence *'Húsz óra'* (in English: *'Twenty hours'*) from a male speaker. Vertical lines show the boundaries of speech sounds. The black and white sections in the bottom denote the voiced and unvoiced speech sounds. One pitch cycle is indicated in the vowel 'a' at the end of the sentence.

1.2 Speech coding

The goal of *speech coding* is to substitute the transmission medium that is necessary for the spreading of the acoustic waveform, and to bridge the temporal and geographical distances between speaker and listener of the human speech communication chain. This way people who are not near each other physically at the same time will also be able to communicate using speech. The temporal distances can be bridged by making recordings and playing them back later. The geographical distances can be bridged using telecommunication technologies. Although the two applications are highly different, from the point of view of speech technology we have to solve the same problem: assign a bit sequence to the speech signal (*encoding*) and reconstruct it (*decoding*). The encoding-decoding method is called *codec*.

1.2.1 Speech and audio coding

In case of *speech coding* it is known that the input is a speech signal, therefore we need to deal with only a part of the possible acoustic waves. If the input can be any acoustic signal (most typically music), it is the field of *audio coding*.

The question arises: if we have a good audio coder, why do we need a speech coder as well? The answer is that the speech coders make use of the specific properties of speech, and thus can achieve significantly lower bitrates (and higher compression) than audio coders, without perceivable quality loss.

The process of speech coding is shown in Fig. 1.3, and we will go through the blocks in the following.

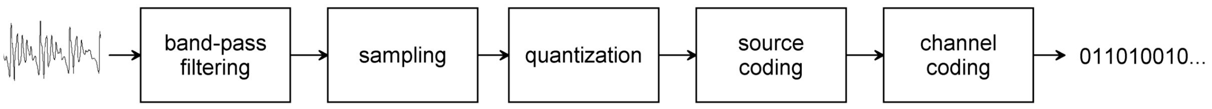

Figure 1.3: Block diagram of speech coding.

1.2.2 Band-pass filtering and sampling

The goal of *sampling* is to convert the continuous time speech signal to a discrete time signal. This can be achieved by storing the current value of the continuous signal from time to time – so we take samples from the signal. The reciprocal of the time between each sample is the *sampling frequency* (f_s). During decoding, we reconstruct the continuous time signal by interpolating between these samples.

Sampling theorem: *The input signal can be reconstructed from the samples, if the highest frequency component (bandwidth) of the input signal is smaller than half of the sampling frequency.*

In practice this can be imagined in the following way. In order to unambiguously reconstruct a sinusoidal, we need to take at least two samples from each period. The input signal can be seen as the sum of several different sinusoids (see Fourier methods). In order to reconstruct the sinusoid with the highest frequency, the sampling frequency should be at least twice this frequency.

If the sampling theorem does not hold, than *aliasing* will occur during reconstruction. This can be very annoying, as the speech will be sonorous / unpleasant-sounding, and there is no way to filter this out later.

In order to surely avoid aliasing, the input signal is band-pass filtered before sampling. This can 'erase' the frequency range above the sampling frequency, and thus the condition of the sampling theorem will hold.

1.2.3 Quantization

As a result of sampling, the input signal will be discrete in time, but the values of the samples are still continuous. In order to handle the signal later in digital systems, it should be converted to have discrete values. This is the *quantization* step.

The number of bits on which a sample is stored is called *bit depth.* The number of *quantization levels* depends on the bit depth. For example, if 16 bit quantization is applied, than $2^{16} = 65\,536$ quantization levels are available – we can differentiate 65k discrete values.

The continuous values are always mapped to the nearest quantization level. The difference between the continuous and quantized value is the quantization error or *quantization noise.*

The quantization technique specifies where the quantization levels are mapped in the continuous range. The simplest quantization technique is linear quantization: the quantization levels are linearly divided in the full range. In case of non-linear (e.g. logarithmic) quantization we take advantage of the fact that some amplitude values are more frequent than others. Therefore, it is worth to define more quantization levels around the frequent values; and to have less levels elsewhere. This way the quantization noise can be decreased. The noise can be further decreased if we quantize several consecutive samples – this is called vector quantization.

1.2.4 Source and channel coding

The goal of source coding is to convert the sampled and quantized speech signal efficiently to a bit sequence (with as few bits as possible). After this, channel coding is used to construct frames or packages that can be sent through the telecommunication network.

1.2.5 Evaluation of speech coders

A speech coder can be evaluated or compared to other speech coders according to several criteria, e.g.:

Speech quality: Different speech coders distort the speech in several ways. The goal is that the encoded-decoded speech remains as intelligible and as natural as possible. The speech quality is a subjective measure: can be estimated by listening tests.

Bit rate: It shows the necessary bit rate of a connection in order to transmit the speech stream (e.g. during a phone call). In case we would like to bridge a temporal distance (and not a geographical distance), than we can calculate the necessary storage capacity to store the encoded file. So the bit rate is a measure of compression as well.

Frequency range: The sampling frequency specifies the frequency range that the codec can transmit (see the sampling theorem). In general, the higher the value of f_s, the more natural the speech is.

Computational requirement (complexity): The computational requirement can be expressed in MIPS (million instructions per second) and gives the complexity of the encoding-decoding algorithms.

Delay: A lot of the speech coders do not encode the speech signal sample by sample, but instead they process longer speech segments (frames). On the receiver side, a frame can only be decoded if all of its bits have arrived. This way, such codecs have larger delay.

Robustness: Robustness means the sensitivity to bit errors and background noise.

Of course, these criteria and aspects are not independent from each other. For example, lowering the bitrate lowers the speech quality, or increases the computational requirement. It is always the target application that determines which of the above aspects are important. In case of speech coders on mobile phones the key questions are the speech quality, the delay and the complexity. The higher the complexity, the higher the power consumption of the cell phone (and the faster the battery runs down). On the other hand, in case of military applications the robustness is a primary criteria instead of speech quality. Here, it is not a problem if the speaker cannot be recognized (they identify themselves at the beginning of every message), but it would be a problem if a helicopter or the battlefield noise distorts the speech.

1.2.6 PCM (Pulse Code Modulation) codec

PCM is the current reference codec in telecommunications. It uses an 8 kHz sampling frequency and 8 bit logarithmic quantization. The source coding is extremely simple, as there is no compression at all. The quality of encoded-decoded speech is excellent, the delay and computational requirement are low (the latter is 0.01 MIPS).

1.3 LPC (Linear Predictive Coding) codec

1.3.1 Basics

It can be observed that the consecutive samples of the speech signal are not independent: from the several previous samples we can estimate the next sample. Linear prediction is a method which aims to predict the next sample based on the previous samples. It is called linear, because the predicted value is the linear combination (weighted sum) of the prevous samples.

If we denote the nth sample with $s(n)$ and p previous samples are used for the prediction, than the predicted sample is:

$$\tilde{s}(n) = \textstyle\sum_{i=1}^{p} \alpha_i s(n-i)$$

The α_i weights are called *linear prediction coefficients*, whereas p is the *order of prediction.* The error of the prediction is the difference between the original and predicted signal:

$$e(n) = s(n) - \tilde{s}(n)$$

This is called the *residual signal.* Fig. 1.4 shows the predicted signal and the residual signal corresponding to a short speech segment. By rearranging the above equation we see that the original signal can be losslessly reconstructed if we know the coefficients and the error signal. Therefore, speech can be efficiently coded using linear prediction. If we choose the linear prediction coefficients well, than the residual signal will be small.

1.3.2 Linear prediction in practice

The residual signal will be small if the original signal is stationary (by this we mean that the statistical properties of the signal do not change a lot). This is not true for the speech in general, but it is more or less true for smaller speech segments. Within 30 ms the speech signal does not change too much, therefore we can consider speech segments of such lengths as stationary signals.

As a first step of linear prediction, we cut the speech signal to short (around 30 ms long) frames. For each frame, we estimate the optimal coefficients, and after that we estimate the residual signal of that speech segment. Therefore, the frame can be represented by the coefficients and the residual signal.

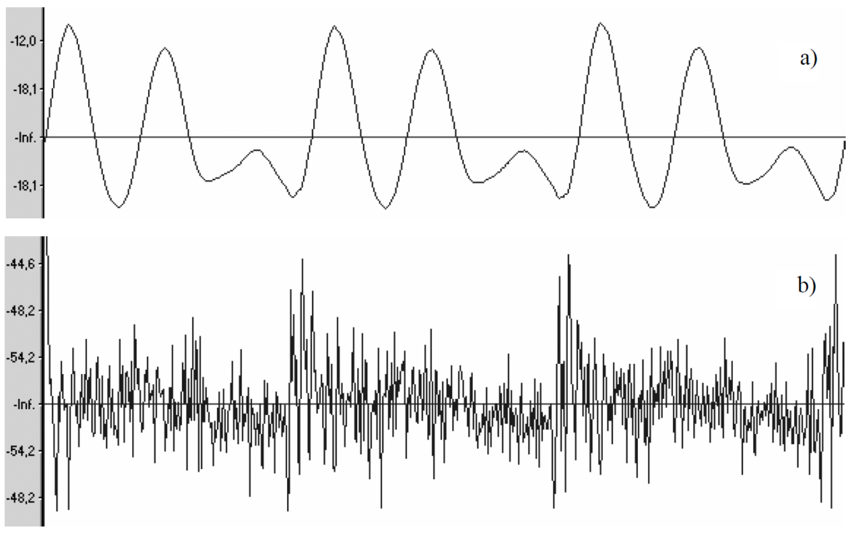

Figure 1.4: a) predicted signal and b) residual signal corresponding to a speech signal. (the Y axis has different amplitudes on the two subfigures).

1.3.3 Estimation of coefficients

The coefficients of linear prediction are optimal, if the predicted signal is as close to the original signal as possible. This can formulated as a requirement that the residual signal should be as small as possible. The coefficients are optimal if the energy of the residual signal is minimal. This minimization task means that it is necessary to solve p equations with p variables. For this, several solutions exist (e.g. covariance method, PARCOR, Burg). We do not deal with the estimation of these parameters, instead we will use built-in functions for them.

1.3.4 Types of Linear Prediction codecs

Every speech codec that we use today in telecommunications – e.g. GSM Half Rate, Full Rate, Enhanced Full Rate, 3G, etc. – are applying linear prediction. All of these methods split the speech signal to short frames. After that the coefficients and the residual signal of all frames are sent through the network. The codecs mainly differ in the way how they compress the residual signal.

Residual Excited Linear Prediction (RELP)

In case of RELP, the residual signal is quantized and sent together with the coefficients of the frame. The quality of the reconstructed signal is similar to that of PCM, but the computational requirement and the delay are much higher (due to the estimation of the coefficients and due to splitting the signal to frames). Why is RELP better than PCM? Because the residual has a smaller energy than the original signal, therefore can be quantized more efficiently, fewer bits are enough. This way, the speech can be compressed, i.e. the bit rate is smaller than in case of PCM.

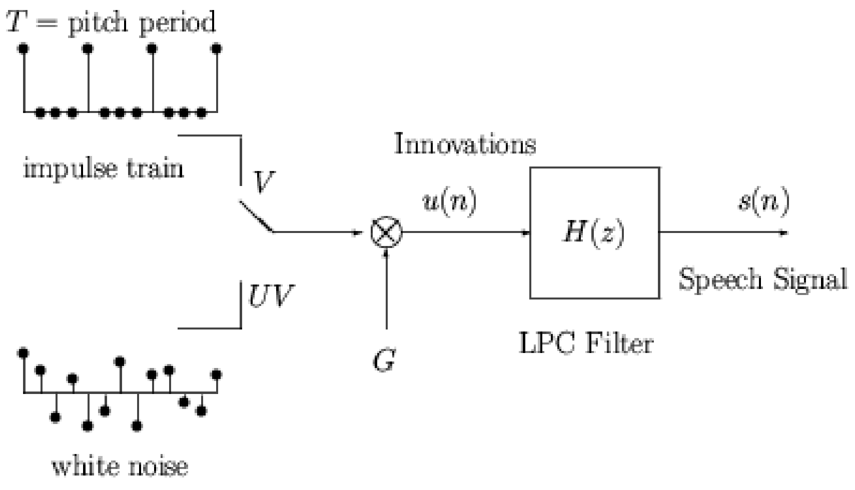

Figure 1.5: Block diagram of the LPC-10 decoder.

LPC-10

LPC-10 is sometimes just called 'the linear prediction codec'. The number 10 means the order of prediction. The codec was developed in the 1970s in the United States Department of Defense for military aims. During this laboratory measurement, you will write the source code for an LPC-10 codec (but with slightly higher order of prediction).

This codec will represent the residual signal in an extremely compressed way, which results in significant quality loss (see Sec. 1.2.5 about the connection between several properties of codecs). For this reason, LPC-10 is not used in its original form in commercial telecommunications.

The central element of the codec is the *pitch detector*. This is an algorithm that can decide for each frame whether it is voiced, and if voiced, it estimates the fundamental frequency.

The *LPC-10 encoder* first cuts the input speech signal to frames. After that all frames are encoded. For each frame, the LP coefficients and the residual signal are calculated. After that, the pitch detector decides about voicing and estimates the F_0 based on the residual signal. For each frame, the following parameters are calculated and transmitted: one fundamental frequency value (F_0), one gain value (energy of the residual signal) and p values of LP coefficients.

The *LPC-10 decoder* generates first an artificial residual signal (see Fig. 1.5). If the frame is voiced, this will be an impulse sequence. If the frame is unvoiced, this will be white noise. The distances of the impulses are based on the reciprocal of the pitch. The energy of the artificial residual signal is set according to the gain of the original residual signal. After that, the speech is reconstructed with the LP synthesis filter.

Chapter 2

Speech coding – laboratory measurement

2.1 Speech analysis

2.1.0 You can download the files necessary for the laboratory measurement from here: `https://github.com/csapot/SpeechCoding/`. Please write a documentation about the steps you have made, including source codes and figures. If you are using Matlab / Octave, the documentation can be a well commented *.m script*. If you are using Python, the documentation should be a well commented *jupyter notebook*.

2.1.1 Open one of the speech recordings using the `wavread` command. If you do not know how to use it, you can get information using the `help` function (in Matlab / Octave: `help wavread`; in Python: `help(wavread)`). Display the waveform and listen to the speech sample using the `wavplot` function. Use this later for displaying and playing waveforms. Try to display and play only the first second of the recording.

2.1.2 LPC encoding is more efficient in lower sampling rates, therefore we resample the loaded recording at 8 kHz. Use the `resample` function, and listen to the result. Pay attention to using the new sampling frequency (8000 Hz) as the second parameter of `wavplot`. Is there any noticeable difference between the original and the resampled recording? What do you think, why?

2.2 Analysis of one vowel with linear prediction

Before we would process the full recording, we will analyze a short vowel segment using linear prediction. An important element of LPC codecs is the function for automatic fundamental frequency detection. Therefore, we will test this first.

2.2.1 Cut a 100 ms long segment from one stressed vowel of the 8 kHz resampled recording (help: first, measure the start and end of the section on the figure in seconds; second, calculate the number of samples; 8 kHz sampling means that in every second we have 8000 samples). Save this to a list/vector, as you will need it later. Display and listen to the cut-out segment. Notice the periods on the waveform, which are the result of the vocal fold vibration. Measure by hand the fundamental period and calculate the fundamental frequency. Based on your knowledge about human speech, is this value realistic?

2.2.2 Now we will test the `pitch_detector` function, which can estimate the fundamental frequency of a voiced speech segment automatically. In case of unvoiced segments, the result will be 0. Perform the following steps.

We know that the fundamental frequency is usually in the range of 80–350 Hz, therefore higher frequencies just disturb the algorithm. Apply a lowpass filter with 500 Hz cut-off frequency on the cut-out vowel. You can calculate the `B` values for the filter with the `lowpass` function. After this, do the filtering with the `filter` function (the `A` value should be 1). This way, the frequency range above 500 Hz is 'erased'. Listen to the filtered speech segment.

After this, call the `pitch_detector` function on the lowpass filtered speech signal. Compare the automatically estimated and the manually measured fundamental frequency values. Note: F_0 estimation is an unsolved research topic, therefore the `pitch_detector` function might output erroneous values. Mostly these are the half or the double of the real fundamental frequency.

2.2.3 Calculate the linear prediction coefficients for the cut-out vowel (before the lowpass filtering). For this, use the `lpc` function. The prediction order should be 12 for now.

2.2.4 Now we will generate an artificial vowel based on the measured pitch value and using the LP coefficients. The `impulse_train` function will create an impulse sequence (see Fig. 2.1) as a source signal. The output of this function is a vector / list which contains the impulse sequence signal (zeros and ones). As an input parameter, you can give the length of the signal (`len`), the time between successive impulses (`period`) and the delay of the first impulse (`init_delay`). This latter determines how many samples should the first impulse be shifted. The `next_delay` output shows that if we would continue

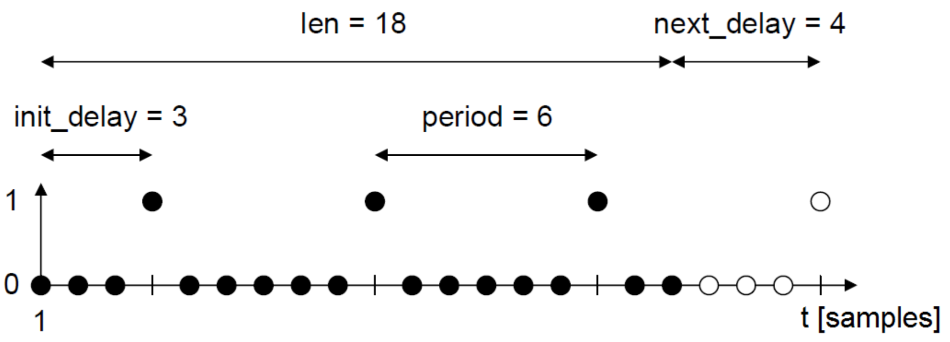

Figure 2.1: An example for the `impulse_train` function.

the sequence with the same time period, than how much shift would be necessary for the next impulse.

Create an impulse sequence with the `impulse_train` function. The distance between the impulses should be the time period of the vowel that you measured earlier (you have to calculate the value in samples), the length should be 100 ms (in samples), and the initial delay is 0. This way, you create an artificial voiced signal that you can listen to. Apply the LPC filter on this source signal with the `filter` function. As the `A` parameter of the `filter` function use the LPC coefficients, whereas the `B` should be 1. Listen to the generated speech and compare it with the original (8 kHz sampled) speech signal.

2.2.5 Now we will calculate the residual signal. For this, apply the inverse of the LPC filter on the original vowel. As the `B` parameter of the `filter`, use the LPC coefficients; and the `A` should be 1. Display and listen the residual signal. Has the amplitude changed compared to the original signal? Was it successful to decrease the energy of the residual signal?

2.3 Implementation of the LPC codec

In this part we will write the source code for a full LPC codec. The encoder cuts the input speech signal to smaller frames, and after that calculates the LPC coefficients, pitch and gain for each frame. The decoder can reconstruct the speech from these parameters.

2.3.1 Write the source code of the LPC encoder. For this, use the drafts in Matlab / Octave / Python. You need to fill the missing parts in the source code, based on the tasks that you did in Sec. 2.2. You can find the steps of the processing and detailed comments in the source code. The function will have three outputs:

- LPC coefficients (p x N size)
- gain values (N size)
- pitch values (N size)

If you successfully wrote the source code of the encoder function, try this with the earlier cut-out vowel. Do the output values (coeff, gain, pitch) seem to be realistic?

2.3.2 Write the source code of the LPC decoder. For this, use the drafts in Matlab / Octave / Python. The input of the function will be the parameters created by the encoder. You need to fill the missing parts in the source code, based on the tasks that you did in Sec. 2.2. You can find the steps of the processing and detailed comments in the source code.

2.4 Evaluation of the LPC codec

2.4.1 Encode and decode the chosen speech recording (not only the cut-out vowel but the full sentence) with your codec, using a prediction order of 12. Display and listen to the result! You can save the resynthesized speech as a .WAV file using the `wavwrite` function. If you want, you can try your codec on other recordings as well. For this, do not forget to resample the new recording at 8 kHz.

2.4.2 If we assume a 16 bit quantization, than what was the bit rate of the original recording (in bit/s)? Let's calculate the bit rate of the LPC codec. For this, we assume that every pitch value is quantized at 8 bit, every gain value is quantized at 12 bit, and every LP coefficient is quantized at 12 bit. What is the compression ratio between the original waveform and the LPC codec?

2.4.3 What is the smallest order of prediction, using which the encoded-decoded speech is still intelligible? For this, encode and decode an unknown recording with a low `p` value, and increase the order of prediction gradually. Take a note about the order when you can already understand what the speaker said!

2.4.4 Now we will modify the speech signal in three different ways.

- Try to generate monotonous speech (having constant pitch). For this, encode the speech, rewrite all non-zero elements of the pitch vector/list to 100 Hz, and decode the speech. Finally, listen to the result!
- Try to generate whispered speech (having no voiced parts). For this, encode the speech, replace all elements of the pitch vector/list to 0 Hz (this way the source signal will be always white noise), and decode the speech. Finally, listen to the result!
- Try to transform the male speech to female (having twice the pitch as males). For this, encode the speech, double all elements of the pitch vector/list, and decode the speech. Finally, listen to the result!

2.4.5 Load some music (e.g. 'hallelujah.wav') and listen to that. Encode and decode it with your LPC codec. Listen to the reconstructed music. What is your experience about this? Think about the differences between audio coders and speech coders. What do you think why is the reconstructed singing whispered?

2.5 Feedback

2.5.1 In your opinion, what was the most difficult in this laboratory measurement?

2.5.2 What was the most important thing that you learned here (1 sentence)?

2.5.3 What was the best and the worst thing in this measurement (1+1 sentence)?