Project 1 - Speech coder

Péter Á. Bánkuti

2019. 05. 06.

ISEL - Instituto Superior de Engenharia de Lisboa

Author's Note **3**

Abstract **3**

Introduction **4**

Component structure 5

Stages **6**

Signal preprocessing 6

Linear Predictive Coding 8

Signal synthesization 11

Signal encoding 12

Signal decoding 13

Decoded signal synthesization 13

Results **14**

References **15**

Author's Note

This project was created in context of the Speech Processing subject of Instituto Superior de Engenharia de Lisboa, held by prof. Carlos Eduardo de Meneses Ribeiro, and the main source of information was his book "Processamento Digital de Fala", for further information on the formulas and theory, consult the book.

Abstract

Speech coding is the process of obtaining a compact representation of voice signals for efficient transmission over band-limited wired and wireless channels and/or storage. In other words, speech coding is also a way to create minimally redundant representation of the speech signal that can be efficiently transmitted or stored in digital media, and decoding the signal with the best possible perceptual quality.

Like many other signals, a sampled speech signal contains information that is either redundant or perceptually irrelevant. Most telecommunications coders are lossy, meaning that the synthesized speech is perceptually similar to the original but may be physically dissimilar. This project is a practical introductory application for such a LPC (Linear Predictive Coding) vocoder, which is considered to be of the higher complexities among vocoders.

Introduction

The aim of speech coding is to reduce the bitrate of the transmitted voice, however with encoding means signal loss. By using the method of LPC vocoder we are getting a very low bitrate, but during the compression the signal loses some of its human tones, on the other hand the content of the speech is clearly understandable. For this reason, this type of encoding is used for example for satellite telephony, by the military.

A vocoder usually breaks the speech-sound into small frames (segments), encodes it and transmits it on the fly, while the decoder receives it frame-by-frame and decodes one at a time. In the context of this project this happens a little differently, however the code base could be converted easily to work as described above. Here, the application instead of transmitting a processed frame of audio it processes a complete audio file and transmits all frame information in one batch to the decoder.

During the project the development was executed in stages. In case of this application each stage has its working component (python script), whereas the input of a component is the output of the previous.

Component structure

The code is organized so each stage of the development can be run independently as component, if the previous was run at least once and dumped the the relevant files. The input of the application is the audio file and the config file, where the latter contains parameters for data segmenting into frames, preprocessing, encoding and decoding.

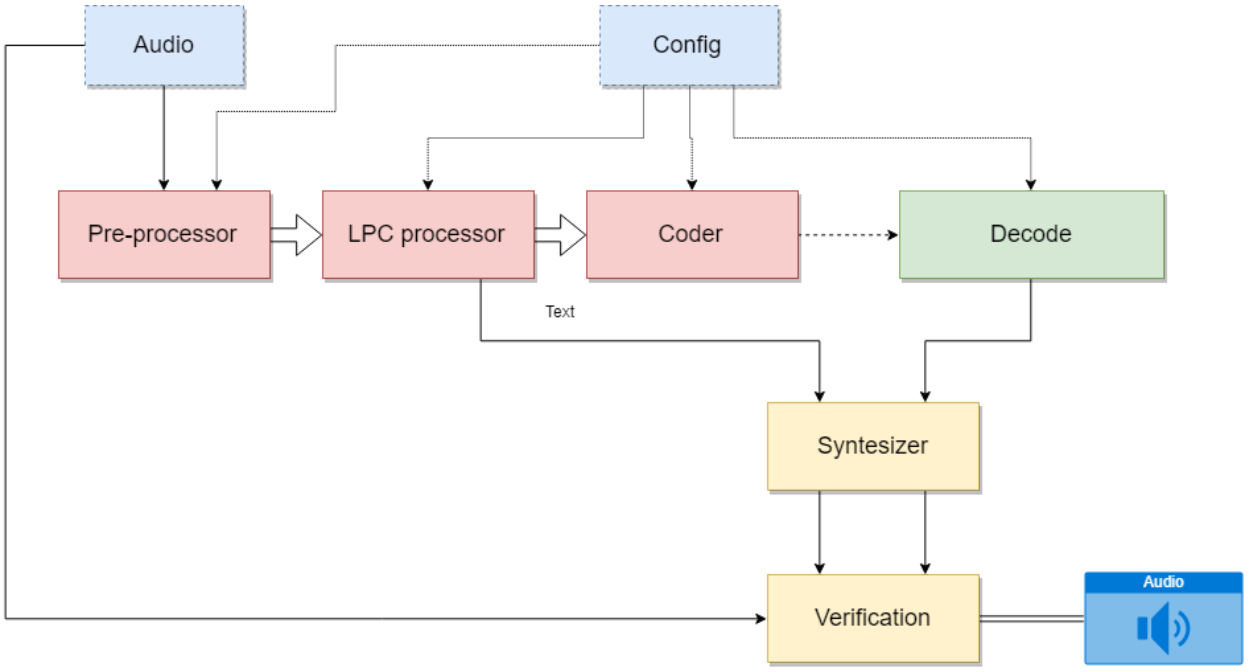

Fig 1: *The structure of the application components and their dependencies.*

Stages

1. Signal preprocessing

During the pre-processing stage the audio signal is segmented into frames (fig.1), and each has its gain, pitch, and autocorrelation extracted.

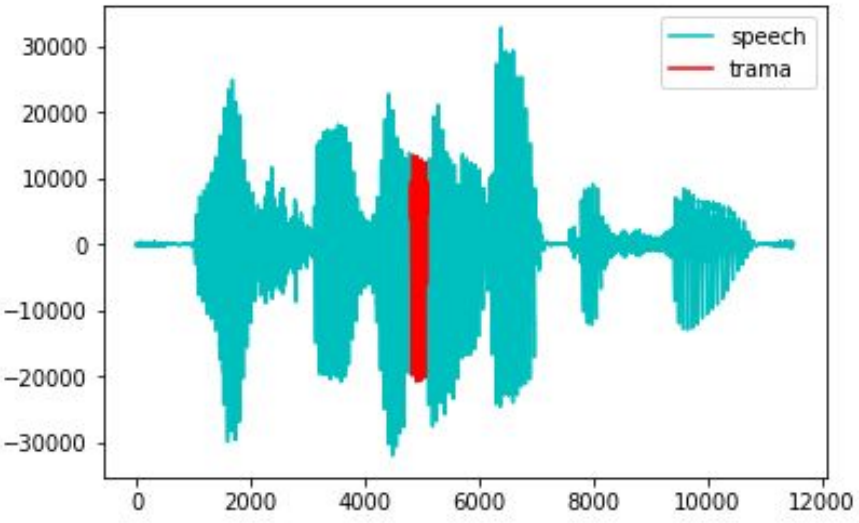

Fig 1.1: *The audio file and a 30th frame of it.*

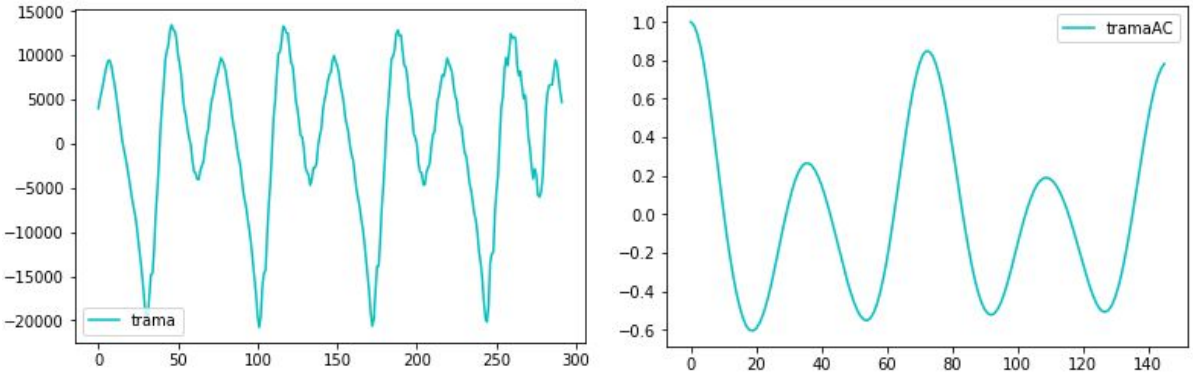

Fig 1.2-1.3.: *The 30th frame up close, and its autocorrelation.*

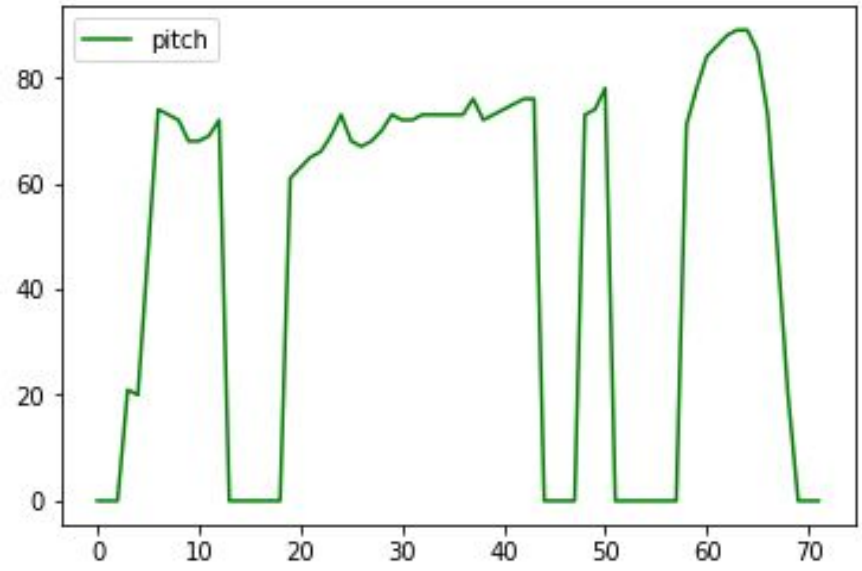

Fig 1.4.: *Pitch of each frame.*

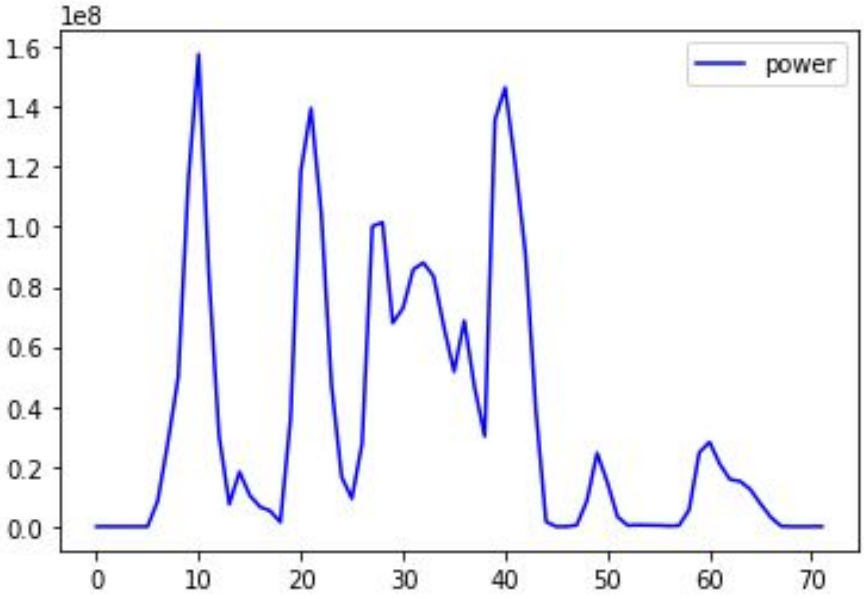

Fig 1.5.: *Pitch of each frame.*

2. Linear Predictive Coding

Linear predictive coding (LPC) is a tool used mostly in audio signal processing and speech processing for representing the spectral envelope of a digital signal of speech in compressed form, using the information of a linear predictive model. It is one of the most powerful speech analysis techniques, and one of the most useful methods for encoding good quality speech at a low bit rate and provides extremely accurate estimates of speech parameters.erat.

From the outputs of the previous component only the energy (~power) and pitch was used as input, the autocorrelation of the frame served no purpose, because the frame had to be processed differently.

Frame processing steps:

- Extract the frame from the signal.
    - If is a voiced (pitch of the frame > 0, then use a high pass filter on it.
- Place a hamming window on it.
- Take the autocorrelation.
- Extract the first p (value given in config = 10) parameters of the AC.
- Calculate the LPC coefficients, using the first p params.
- Calculate the gain.

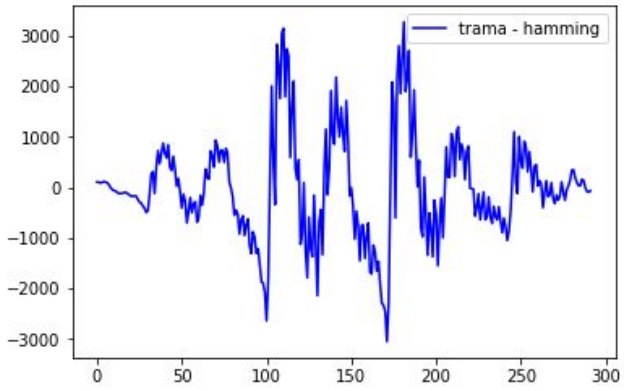

Fig 2.1.: *30th frame after applying high pass filter and hamming window (Hp-H-frame).*

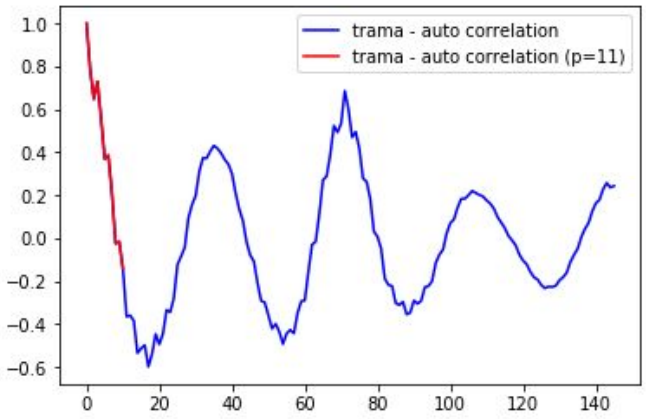

Fig 2.2.: *The first p values of the autocorrelation.*

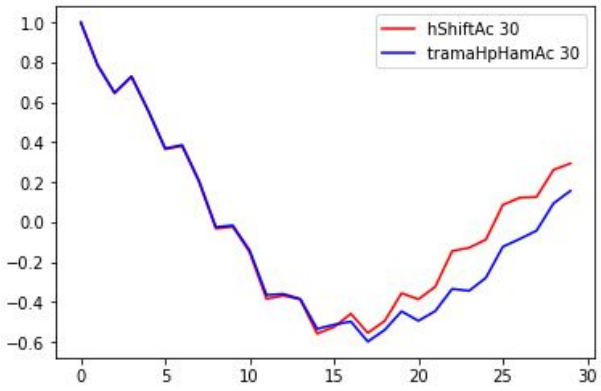

Fig 2.3.: *AC of the impulse response of LPC of order p+1 and of the Hp-H-frame.*

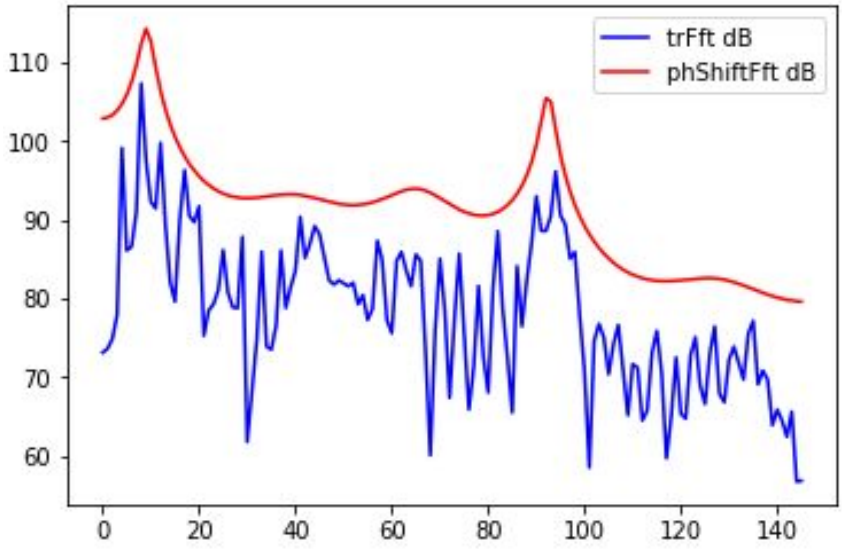

Fig 2.4.: *Periodogram of the 30th fame*

The figure above (fig.2.4) shows the relationship between the fourier-transform of the frame and of the impulse response of the LPC coefs of order p+1. Given that the first p values of the autocorrelation are define the spectral envelope of signal. The p is chosen to be 10 because the the improvement of the prediction is minimal above this order.

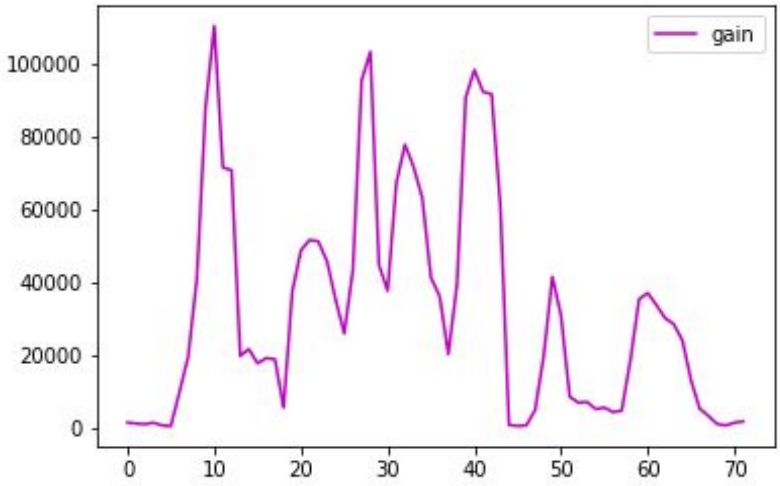

Fig 2.5.: *Gain of each frame.*

3. Signal synthesization

A component was made for synthesizing the audio from the LPC coefficients, gain and pitch of each frame. Consequently, at this stage it can be verified the result of the previous stages by comparing the synthesized audio with the original.

Synthesization:

- First, the pitch is checked if is available at a given frame:
  - If not than, the frame will have the waveform of white noise multiplied by the gain..
  - If yes the audio wave is composed from the pitch, gain and LPC coeficients.
- During the composition each frame is segmented in a way that the pitch is the length of the segment, and the last segment runs over the border of the frame into the next. Consequently, the next frame starts with an offset.
- For each segment an impulse waveform is given, representing the gluteal pulse. Where the gain will set the amplitude, and the pitch the length of this wave. (Fig 3.1)

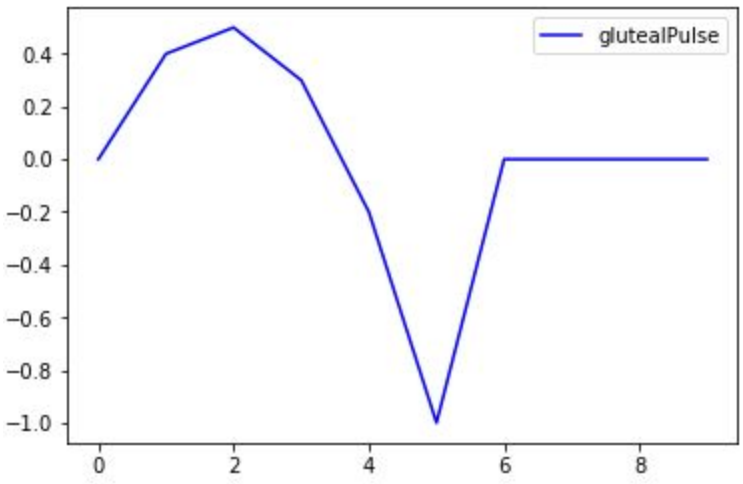

Fig 3.1.: *Gluteal pulse for gain=1 and pitch=10.*

- After having concatenated all gluteal pulses within the frame, the waveform is created by providing into the linear predictor:
    - the last p+1 values of the last frame (in case of the first frame only zeros);
    - the gluteal waveforms;
    - and the LPC coefficients;
- Lastly, the predicted wave of each frame is concatenated, thus the soundwave is regenerated.

The resultant waveform of this synthesization visually highly differs from the original, however by listening to it the content can be clearly understood.

4. Signal encoding

After, having decomposed the signal into LPC coefficients, pitch and gain for each frame the data, the audio can be transmitted efficiently. For this, these values have to be binarized.

Pitch binarization: the pitch will be represented by 7 bits where the unvoiced value 0000000 is 0-19, 20+127 is the range of the rest.

Gain binarization: the gain is represented by 5 bits and the binarization happens with the use of a predefined bit quantization scale after the normalization of the gain values. In order to be able to restore the values during the decoding stage the max value of gain is saved, too.)

LPC binarization: first, the LPC is converted into intermediary representation of Line Spectral Pairs (LSP) The LSPs have some interesting characteristics: the frequencies are related to the formant frequencies; the dynamic range of is limited and the two alternate around the unit circle; the pairs correlate and they change slowly from one frame to another, hence, interframe prediction is also possible. Following the conversion the p amount of LSP coefficients are

biarized with the use of a predefined bit quantization scale for each of them, summing a total of 34 bits (3+4+4+4+4+3+3+3+3+3).

The result is a 5+7+34 bit representation of each frame, which can be saved or transmitted efficiently.

5. Signal decoding

The receiver has to implement the inverse procedures of the encoder. First it has to unpack each 12 of the values. Following, the pitch can be retrieved by simply adding 19 to every non-zero pitch values. The gain is restored by dequantizating with the use of the predefined table and multiplying the value by the previously saved max gain. The LPC is retrieved by dequantizating the LSP values and transforming those back to LPC.

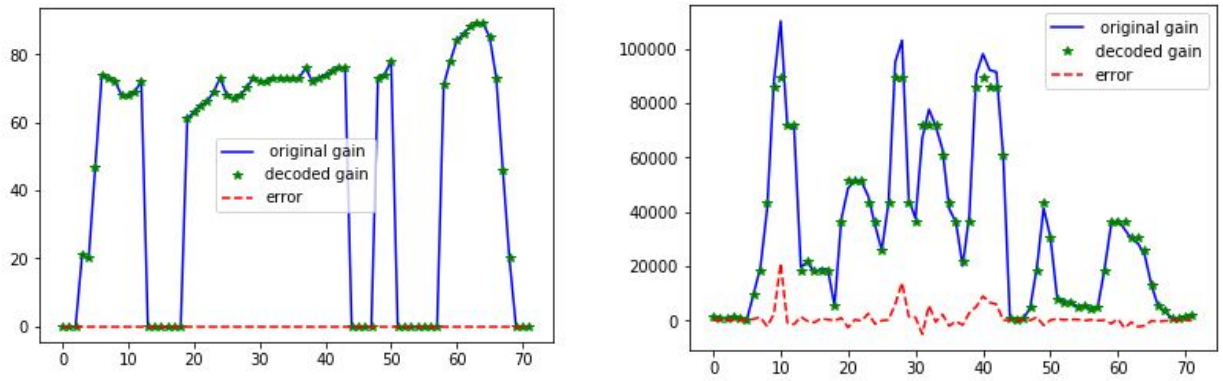

Fig 5.1-5.2.: *Comparing the decoded and original gains and pitches.*

6. Decoded signal synthesization

After having decoded the pitch, gain and LPC coefficients of each frame, the audio synthetiser can be called again to create an audio waveform.

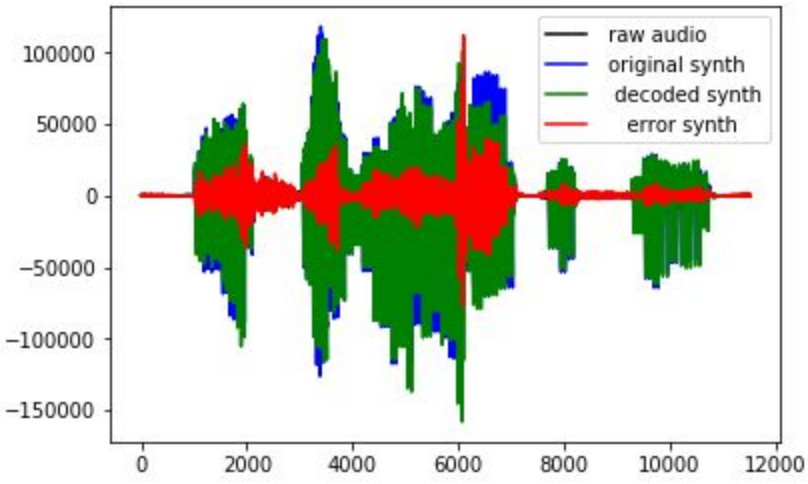

Fig 6.1.: *The audio waveform.*

Results

As expected, visually, from a birds-eye-view the resultant waveforms are fairly similar with similar amplitude patterns, however up-close the individual values are showing a big disparity.

By listening to the audio the the loss of quality is obvious in the case of comparing the original to the synthesized, however the content is still clearly audible. While the quality of the synthesized encoded and decoded signal is pretty much the same.

The compression:

Original file size:	184368 bits
Encoded file size:	3312 bits

References

1. Carlos Eduardo de Meneses Ribeiro.: Processamento Digital de Fala. Instituto Superior de Engenharia de Lisboa
2. Alwan, A.: Speech coding: Fundamental and applicatons. *University of California at Los Angeles*