Final Project: ECE 6260

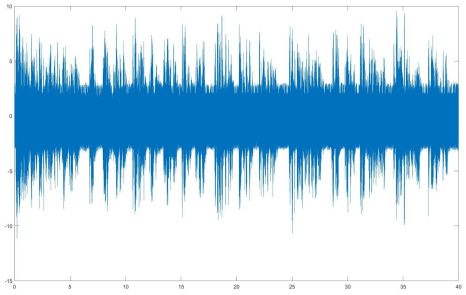
By Neha N Gowda and Thomas BENOIT.

Spring 2019

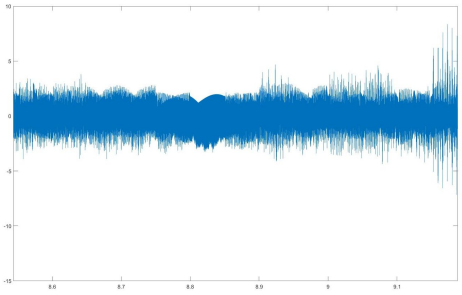
Table of contents

I) Observation Analysis

Before analysing deeply the signal, we generated an audio file of the given *FTPSig* file so as to listen to it. This signal was a real brouhaha! The following plot would be more explicit **(figure 1)**.



Global view of FTPsig [0 to 40 seconds]



Zoom on a part of FTPsig [8.5 to 9.2 seconds]

The initial signal (FTPSig) in the time domain ***- figure1***

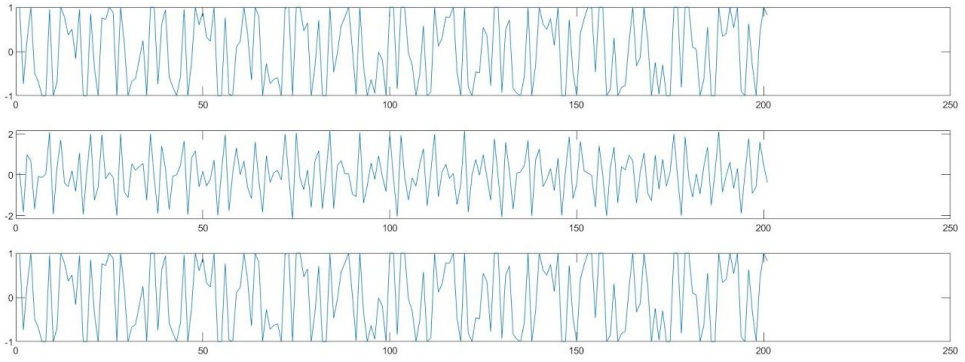
After listening to the signal we thought of *FTPSig* as very dense and surely an association of different sounds.

As in HW4, we did a short-time linear prediction analysis to obtain more features . In testing the further parameters in *LPCdemo.m (from the HW4 correction)* ***(table 1)***, we heard a low speech. Note that their associated plots are in the *IMGS/LPC-plots folder*.
Actually, the synthesized signal show some redundancies that remind speech patterns, as we have seen in the course [*Lecture 27*] ***(figure 2)***. Thus, the listening and the mathematical analysis support the assumption that FTPSig would be composed to a speech part, and so, a “no-speech part”.

Table 1: ***tested LPC’ parameters - their associated plots are located in MGS/LPC-plots.***

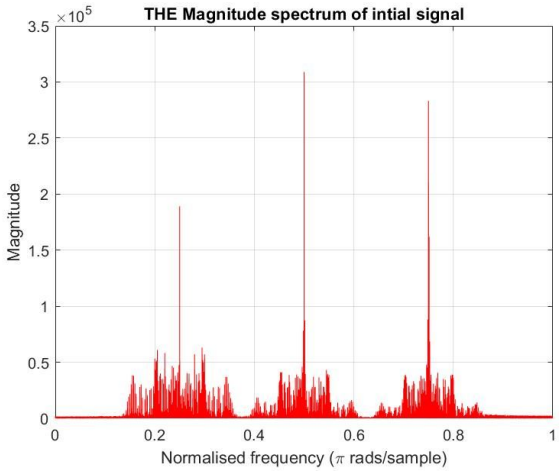
length of Frames analysed	Covering selected	Could hear a speech?
20ms	10%	badly
20 ms	25%	badly
20 ms	50%	slightly
20ms	60%	slightly

50ms	10%	slightly
50ms	20%	slightly
50ms	50%	slightly
50ms	60%	slightly
75ms	10%	slightly
75ms	25%	slightly
75ms	50%	slightly



An example of plot obtained from LPCdemo.m . 1st line: the initial FTPSig, 2sd: the error of prediction of FTPSig and 3d: the resynthesized FTPSig - Located in IMGS/LPC-plots folder ***- figure 2***

Another point to explore is the signal's representation in the frequency domain. To analyse this aspect, we know two tools: the spectrogram and the magnitude spectrum. Firstly, let's plot the magnitude spectrum of FTPSig **(figure 3)**. There are "peak clusters" respectively located in *4000Hz, 8000Hz and 12000Hz*.



Magnitude spectrum of FTPSig (plotted on a demi-period) ***- figure 3***

In studying each cluster composition, we have noticed that these “peak clusters” would respect a specific “shape pattern”. After testing different magnitude spectrums, we found that “peak clusters” are actually “chirp clusters”. Each cluster would at least have three chirps (with different widths). At this analysis stage, we have to confirm these assumptions with another representation. For instance, the central peaks in each cluster could be both a *cos* and a *sin*.

Spectrogram Analysis -> there are chirps and parasites , how to separate them from the speech? What is their structure?

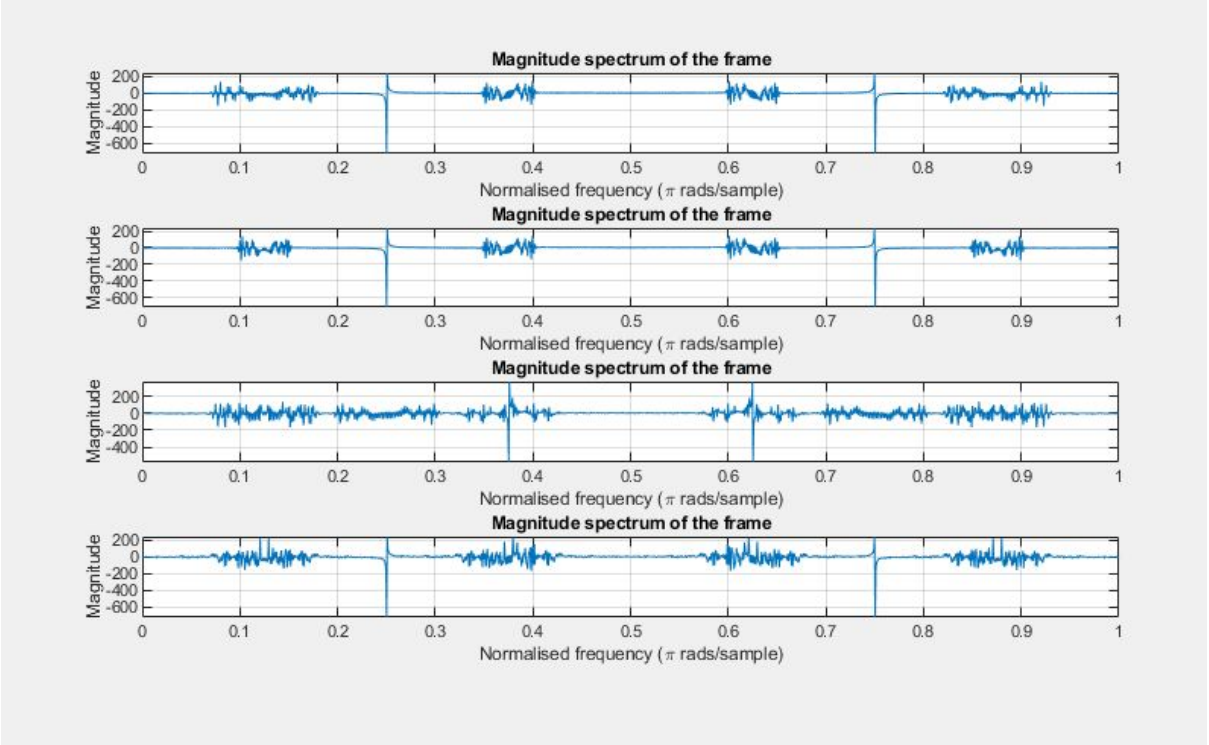
We started off by looking at the spectrogram of the signal at different frame lengths and recognized easily two different kinds of chirp which exists in three different band frequencies. So on seeing this we modelled them and tried to subtract it from the signal only to realise that the chirp signal was not continuously appearing as the same in all frames.
On observing consécutive frames of the signal with each frame of 1600 samples we found that the chirp signals were present only at periods of 50ms. We could not find out if these chirps appeared periodically, but we saw each 1600 sampled frame without overlap had different combination of the two chirps. There were places in the signal where neither chirp existed.

Firstly, we tried to directly remove the chirp signal without considering that they do not appear at all times. Obviously it did not work, the chirps were persistent. We then started to look at the spectrogram even more closely and decided to do frame by frame analysis and reconstruction of chirps based on each frame and frequency band. We observed frames of 1600 samples and saw their spectrogram and the fft analysis. The lead to the observation that there were three different chirp signals in each band of frequency. The three band of frequency was identified as:
1600 to 6400 - 4000Hz band
5600 to 10400 - 8000Hz band
9600 to 14400 - 12000Hz band

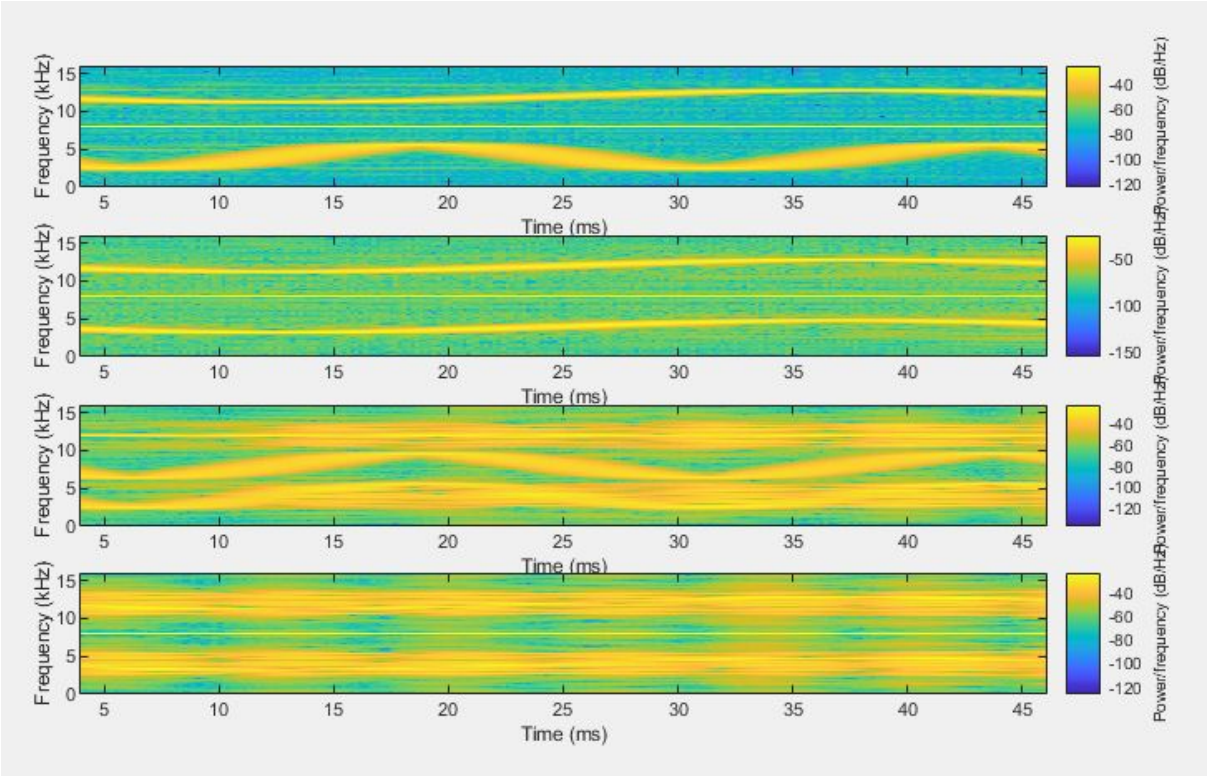
The three various chirps were identified and were modeled as given in the file **“” ”**. Further, the pattern of the appearance of these chirp signals was a new task to model after we constructed the chirps in each frame. This was achieved by using the instantaneous frequency of the signal being studied. The signal was divided into three different bands and based on the instantaneous frequency of the signal in each frame, the chirp signal present in that frame was noted. In each band a similar analysis with the help of the instantaneous frequency was done and the overall chipr was calculated.

We need to also note that, for the chirp analysis, it was not the spectrogram alone which was observed. We used the FFT of the signal in each frame seeing how the chirp signal varied with respect to time. Seeing the FFT frame analysis and the spectrogram together only helped us to observe the third chirp.

Some of the frame analysis is given below.



FFT analysis of each frame(1:1600) (4 consecutive frames) : ***figure 4a***

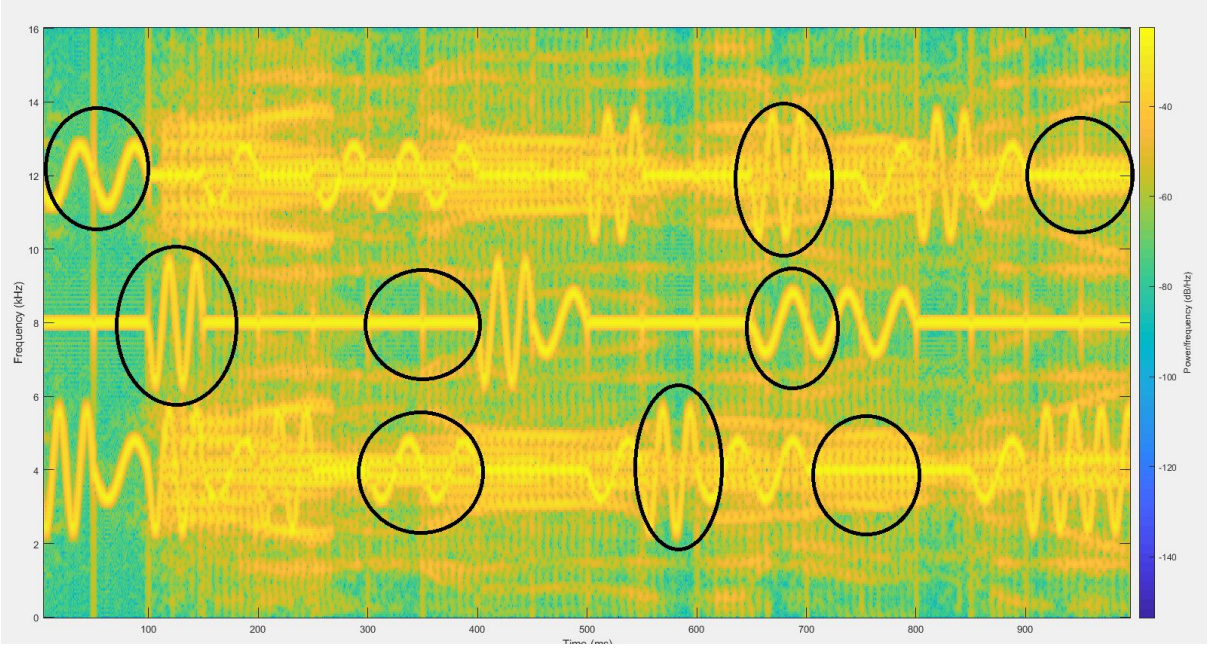


Spectrogram analysis of each frame(1:1600) (4 consecutive frames) : ***figure 4b***

Analysis conclusion

FTPSig consists of several elements. From the short-time linear prediction analysis, we inferred that the signal includes a speech. The frequency-domain studies revealed that three clusters of chirp are placed at *4000Hz, 8000Hz and 12000Hz*. Each cluster regroups three chirps. In telecommunication, the voice frequencies range from *300 Hz to 3400 Hz*. Hence the chirps cannot belong to the speech part. Only the chirps in 4000 Hz could be mixed with the speech frequencies.

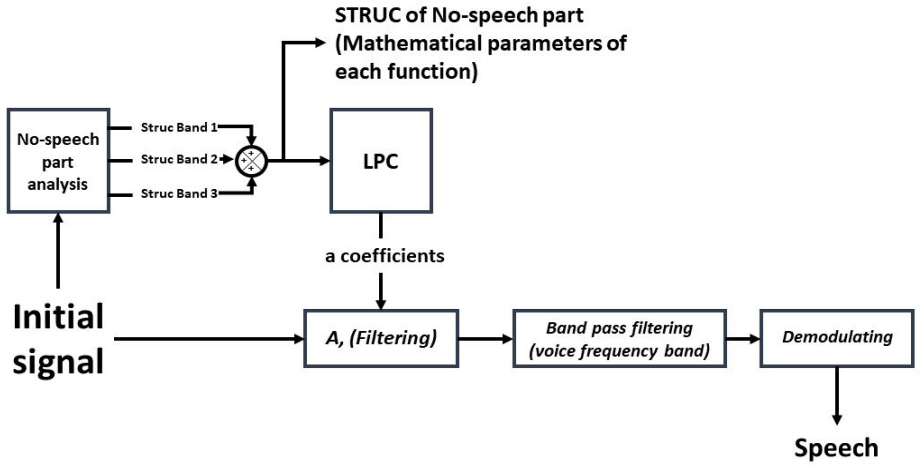
Finally, we find the *FTPSig* constitution. To compress effectively *FTPSig,* we should separate the speech from the chirps part, encode them separately and then gather together them. The next parts will display and detail how we proceeded these steps.



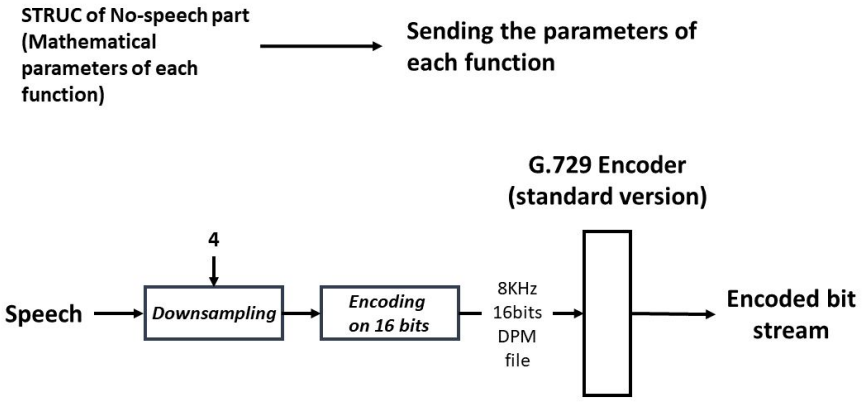
Spectrogram of the signal (1:32000) with the different chirps encircled: ***figure 5***

II) Operating diagrams

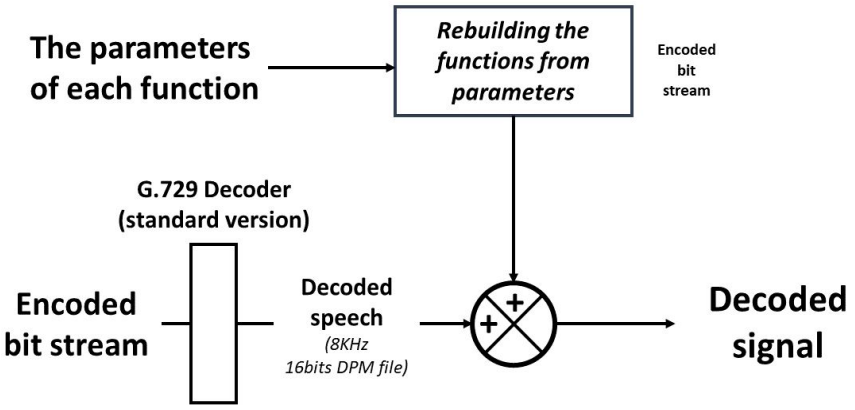
In this section, operating diagrams are displayed. The details of each block is exposed in the third section (III Implementation). The three following diagrams are respectively the separation, the encoding and the reconstruction schemes **(figure 6a, figure 6b , figure 6c)**.



The steps to separate the speech to the other elements in the initial signal ***-figure 6a***



The steps to compress the signal elements ***-figure 6b***



The steps to reconstruct the global signal ***-figure 6c***

III) Implementation

Structure of chirps

The chirps as seen observed from the FFT analysis and spectrogram analysis were modelled. Using the instantaneous frequency comparisons their occurrences were determined. We got the parameters of the chirp signals in three different bands as follows:

Table 2: ***Different Chirps and their parameters***

4KHz band

	Centre Frequency	Amplitude	Phase
Chirp 1	4000	1	$8000\pi t + 40\ cos(80\pi t)$
Chirp 2	4000	1	$8000\pi t + 40\ cos(40\pi t)$
Chirp 3	4000	1	$8000\pi t + cos(\pi t)$

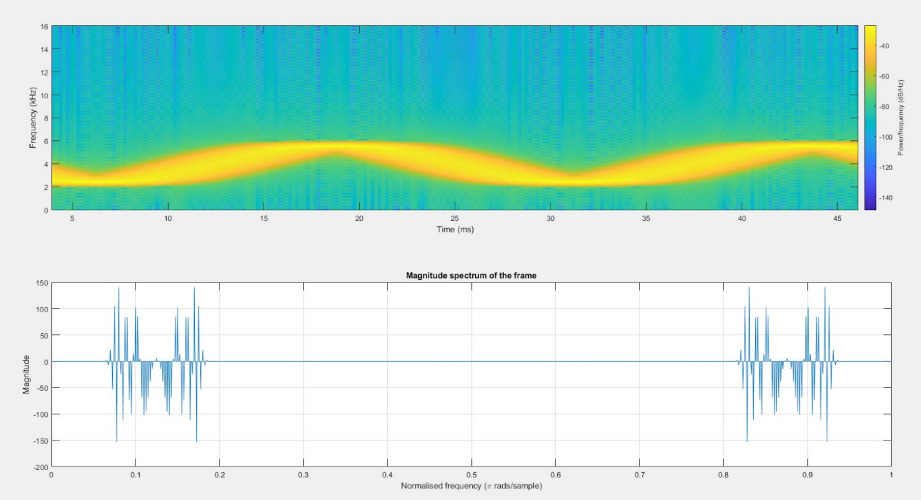
8KHz band

	Centre Frequency	Amplitude	Phase
Chirp 1	8000	1	$16000\pi t + 40\ cos(80\pi t)$
Chirp 2	8000	1	$16000\pi t + 40\ cos(40\pi t)$
Chirp 3	8000	1	$16000\pi t + \ cos(\pi t)$

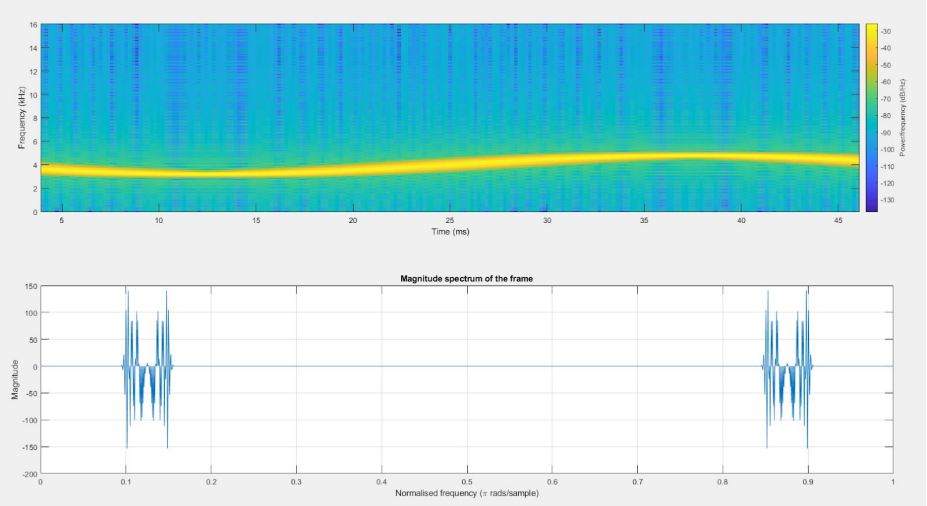
12KHz band

	Centre Frequency	Amplitude	Phase
Chirp 1	12000	1	$24000\pi t + 40\ cos(80\pi t)$
Chirp 2	12000	1	$24000\pi t + 40\ cos(40\pi t)$
Chirp 3	12000	1	$24000\pi t + cos(\pi t)$

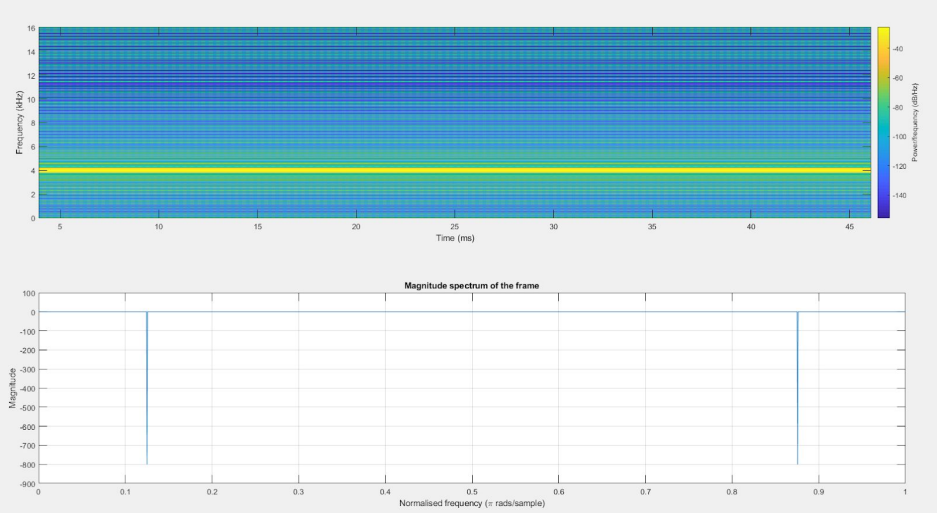
Based on the analysis the chirp signals that were modelled looked as follows:



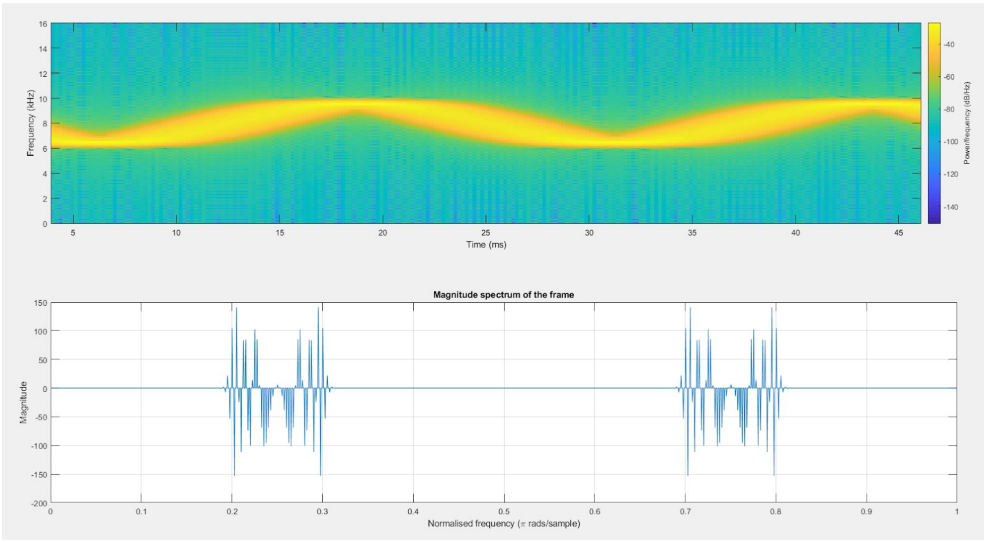
Chirp 1 at 4000Hz : ***figure 7a***



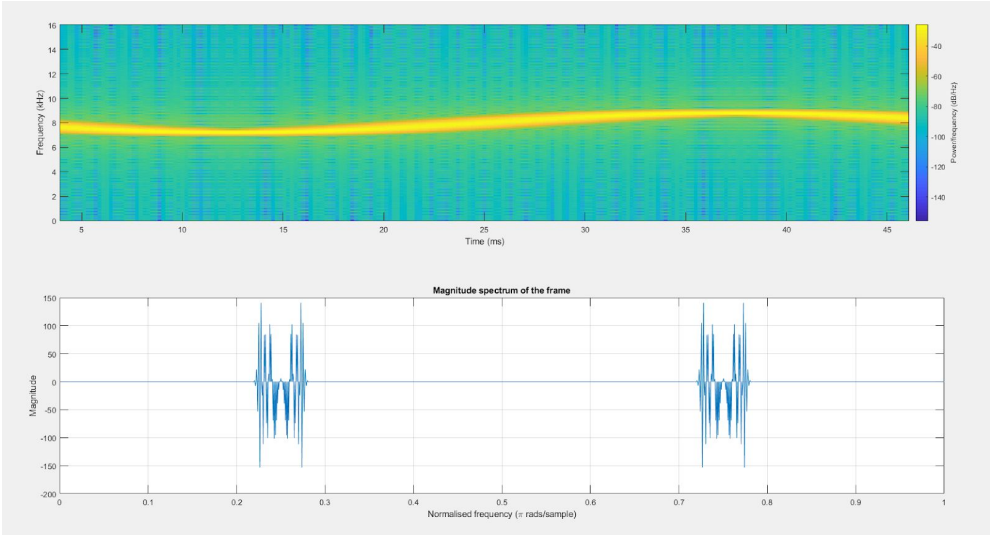
Chirp 2 at 4000Hz : ***figure 7b***



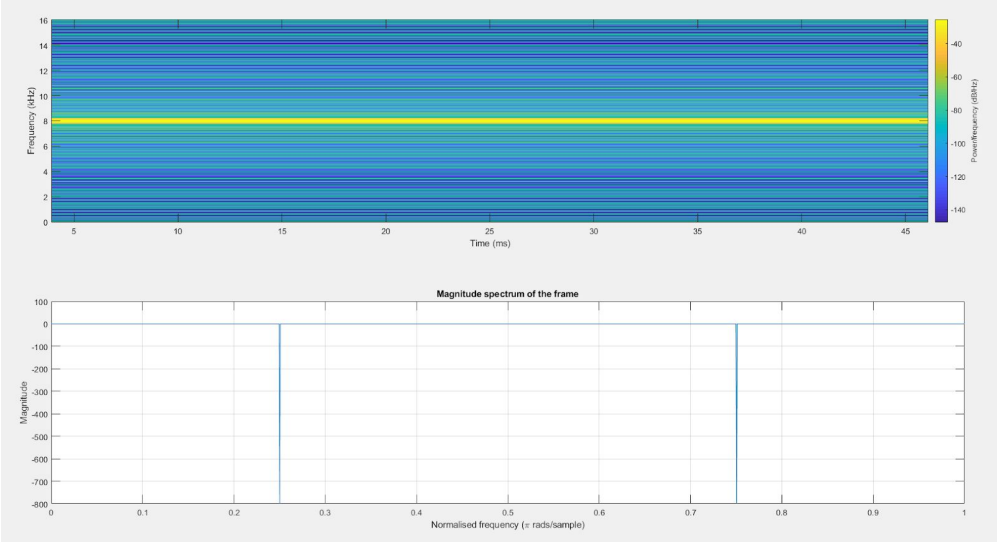
Chirp 3 at 4000Hz : ***figure 7c***



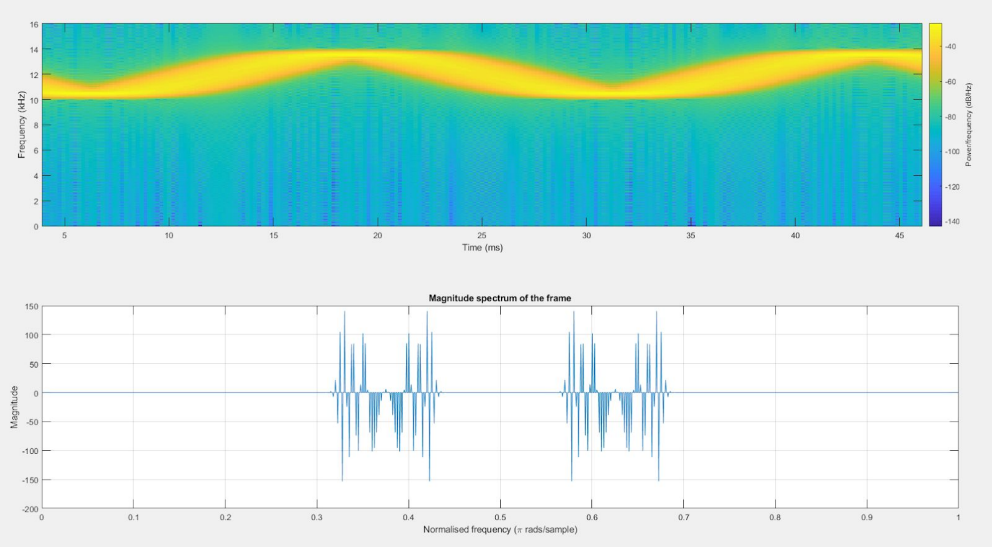
*Chirp 1 at 8000Hz : **figure 8a***



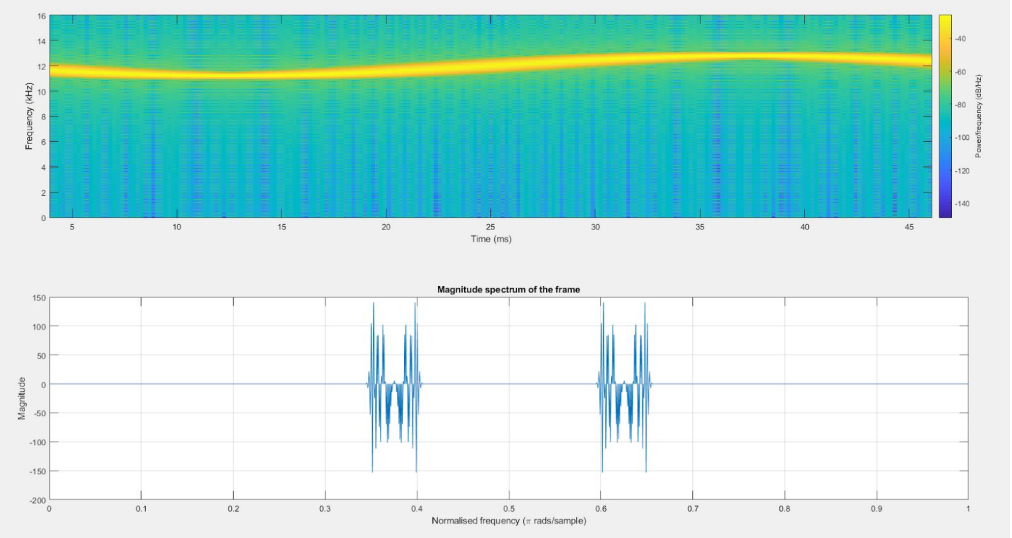
*Chirp 2 at 8000Hz : **figure 8b***



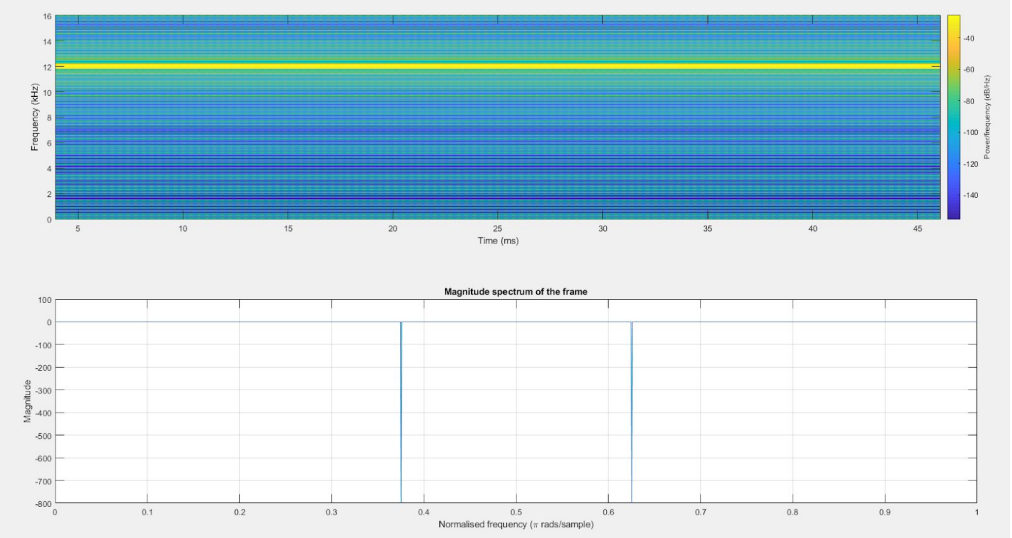
*Chirp 3 at 8000Hz : **figure 8c***



*Chirp 1 at 12000Hz : **figure 9a***

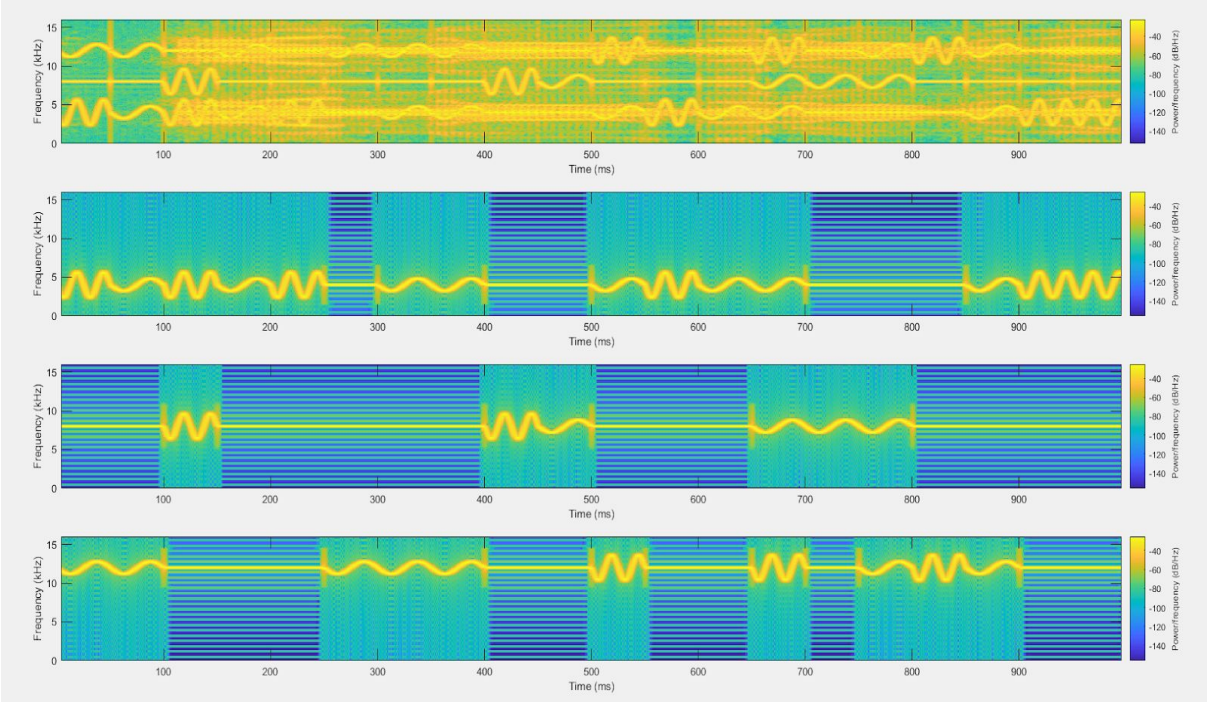


*Chirp 2 at 12000Hz : **figure 9b***

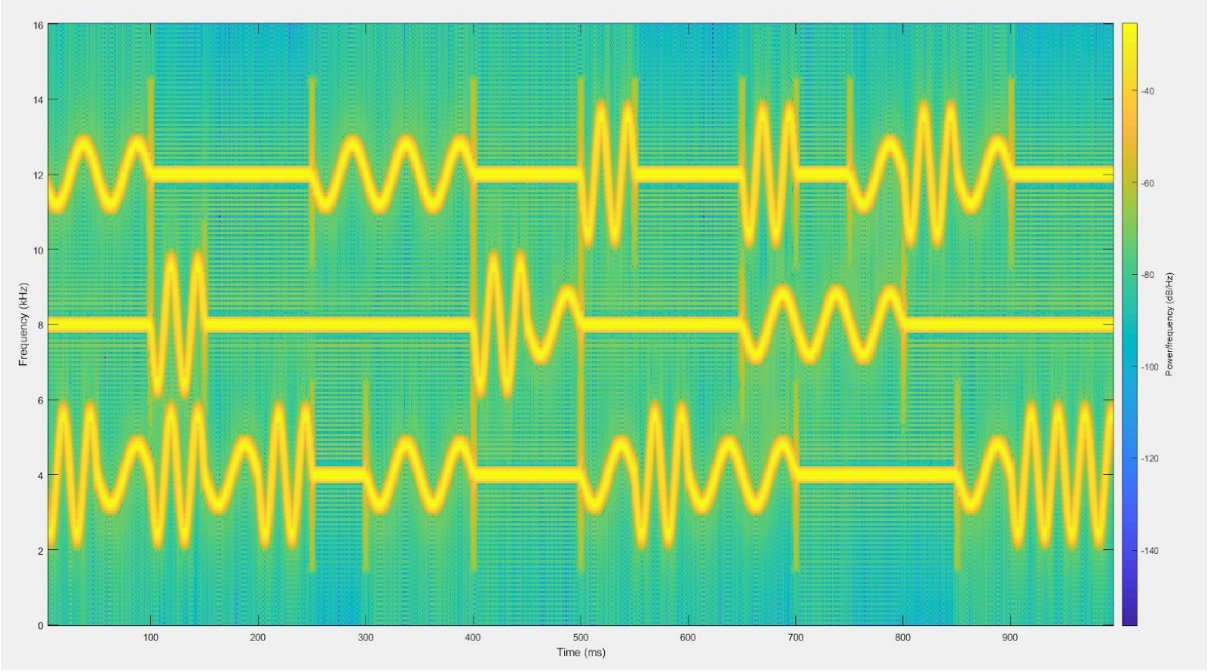


*Chirp 3 at 12000Hz : **figure 9c***

The above were the various types of chirp signals, now having them to occur in the way that it occurs in the original signal gave the following overall chirp signals:



Spectrogram of Chirps of various frequency bands (1:32000) : ***figure 10a***

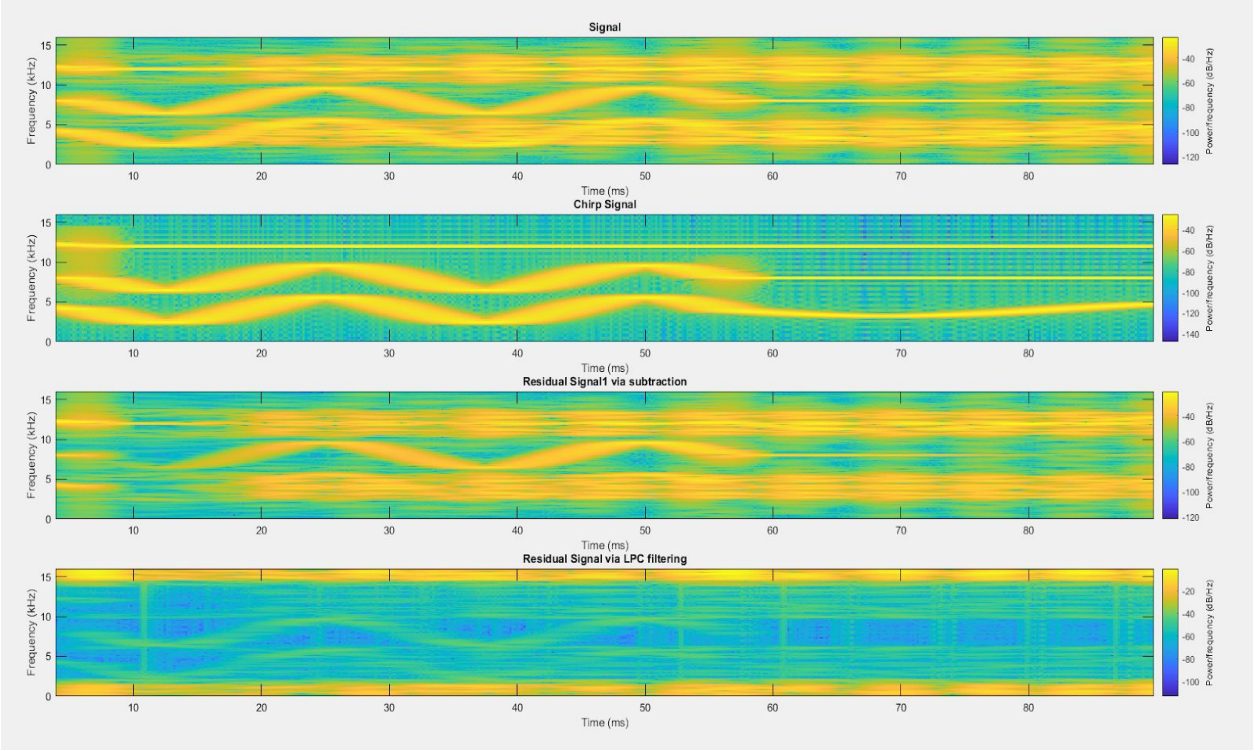


Spectrogram of Overall Chirps(YCH(1:32000)) : ***figure 10b***

Filtering the structure of chirps

Now once we got the structure of the chirps, we tried to remove it from the given signal. The method that we first tried was to just subtract the signals. But, we were not successful and so we had to resort to other methods to filter out the chirp. Our next stop was the LPC filtering approach. We modelled the overall chirp signal through LPC analysis. Then used the generated filter coefficients to filter the signal. This way we removed the components of the chirp from the signal. Below are the spectrograms of both kinds of approach.

Note: This was not the effective method, but we found this to be better than just subtracting the signals.



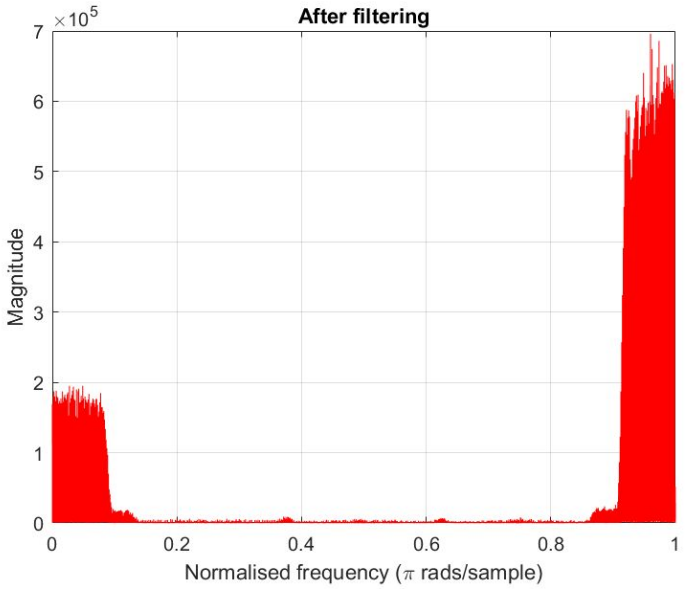
Spectrogram of the signal, chirp, residual signal1 via subtraction and residual signal via LPC filtering (3000:6000) : ***figure 11***

The residual part of the signal after filtering also had chirps still persisting. We tried to analyse these and found they are also associated with the higher frequency. We could also on more observation of the spectrogram of the residual signal tell that the chirps that were present were redundant with respect to frequency. The chirps existed in higher and lower frequencies. But due to shortage of time we could not remove these chirps. Having more different other tools like bandpass filtering and demodulation we went about extracting the speech as explained in the next sections.

The reconstructed chirp signal is in the audio file: RecontructedChirps.wav.

Processing the part filtered out chirps

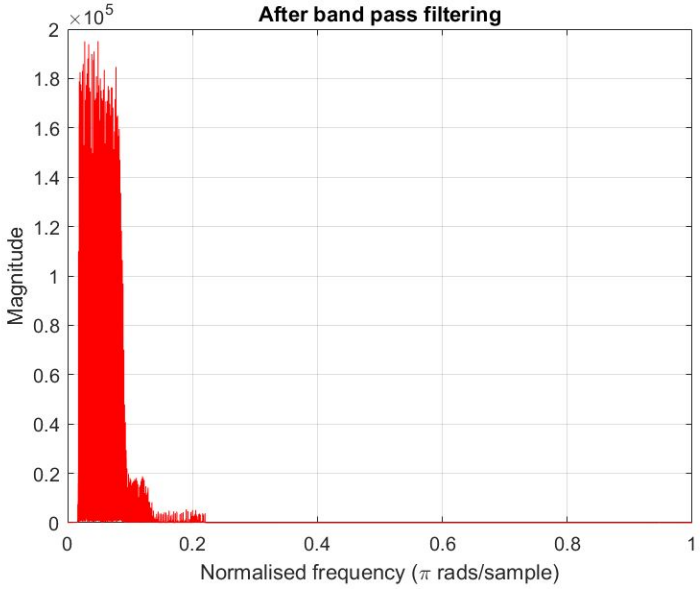
After filtering the initial signal, we isolate the speech from chirps. Nevertheless, residual chirp parts and unwanted signals remain **(figure 12).**



The magnitude spectrum of the signal filtered out chirps ***-figure 12.***

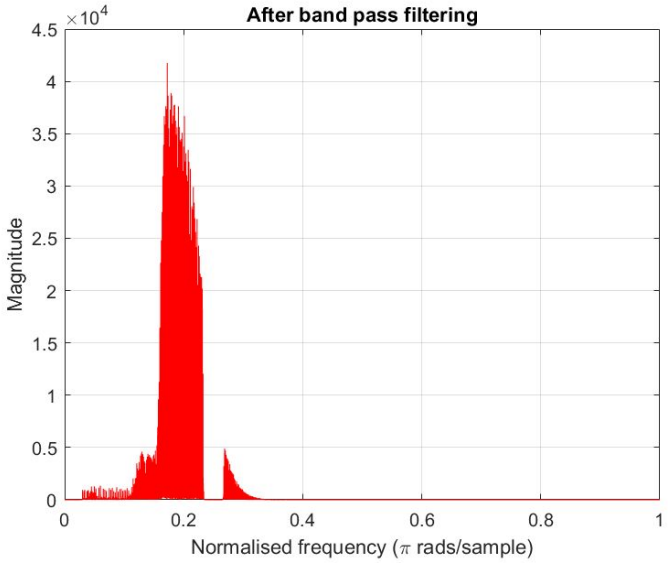
From the course documentation, we infer that the *"right and the left hills"* are the frequency representations of two damped sinusoids. The right damped sinusoid is not useful to recover the inner speech part (*remind that voice band ranges from 300Hz to 3400Hz*). Furthermore, some persistence frequencies of chirps could stay after filtering.

In order to extract the inner speech, we firstly band pass the remaining signal to keep the voice frequencies. In testing different bands, we conserve frequencies between *300Hz and 3500Hz*. The matlab function *"bandpass()"* was used.The results are plotted in the **figure 13**.



The magnitude spectrum of the signal after bandpass filtering ***-figure 13.***

Now, we should remove the left damped sinusoid. To fix this problem, we have to demodulate the speech out the damped sin. We used the matlab function “demod()” and perform an amplitude demodulation. As a matter of fact, “demod()” in am mode multiplies the signal by a sinusoid and apply a fifth-order Butterworth lowpass filter. In sum, we demodulated the signal twice: firstly with a *8000Hz* frequency sinusoid and then with a *4000Hz* frequency sinusoid. We obtained the following signal **(figure 14)**.



After demodulating the signal ***- figure 14***

Therefore, we reduced the remaining damped sinusoid and extracted fully the speech part. This gives us the audio file: Residual1.wav.

G.729: ENCODING - DECODING

A the next implementation step, we have to encode the speech part. One popular coder is the standard version of G.729 provided on *Canvas*. In addition to be royalty-free, *G.729* is simple to use (the C files should just be compiled to obtain functioning *coder/decoder*).
Note that there are several annexes of G.729 offered by *the github of opentelecoms-org [2]*. We are seeking to keep a good voice quality, hence we are not interested to employ G.729A Although they are more sophisticated, *G.729B and G.729J/G.179.1* could be attractive, due to respectively the silence compression and the hierarchical enhancement layers. However, we will see that the standard G.729 gives good results. It is why we keeped the standard version to compressed the speech part.

Note that employed program only considers *8KHz 16bit pcm inputs*. This format defines a binary audio file modulated on *16 bits* and sampled on *8KHz*. The standard *“.wav”* format is already modulated on 16 bits, and we had to “*downsampled”* the speech part by 4 (a sampled frequency of *32000Hz* downsampled by 4 gives a sampled frequency of *8000Hz*). The matlab function “downsample()” was used to do this step. The input file was well

formatted. The coder executable computed a compressed bit stream. The decoder reconstructs the compressed speech in *8KHz 16bits pcm binary file.* To handle this files on Matlab, we used the fread() function to read the file content.

Therefore, we could generate a compressed speech file and compute a decoded/reconstruct speech.This gives us the audio file: F32000_CMPRSS_SPEECH.wav.

Reconstruction

For the reconstruction of the signal, we have two parts of the signal. Part one the compressed speech signal from G.729 and the second part is the chirp signals. We decode the signal from G.729 and just add it to the reconstructed chirp signal to achieve the original signal.

F32000_CMPRSS_SPEECH.wav + RecontructedChirps.wav
= RECONSTRUCTED_OVERALL_SIGNAL.wav

The reconstructed audio file is RECONSTRUCTED_OVERALL_SIGNAL.wav.
This file almost sounded like the original signal.

IV) Results

The results are computed in the "Computing Measurements" section in the main matlab program, located in the folder pgm/ .

The **table 3 and 4** displays the compression outcomes of our solution. Comments are attached to detail the computations and the selection of measurements.

Table 3 - compression results for speech:

Measurements		Values	Comments
bits/s (fixe rate)		8 kbit/s	*The standard version of G.729 uses a fixed 8 kbit/s data rate.*
Signal to coding noise ratio	*8000 Hz sampled **	-2.1482	They were computed such that: *SQNR = 10*log_10 (Psig/P(sig - decoded))* with sig: the initial signal and decoded: the decode one
	*32000 Hz sampled **	-0.7489	
Spectral distortion *(log spectral distance)***	*8000 Hz sampled **	0.0899	*The log spectral distance is computed from the truncated cepstral distance*
	*32000 Hz sampled **	0.9972	

*: *The results were recorded in using both frequency sampling. For 8000Hz, we kept the initial version of decoded speech and we downsampled the speech to encode. For 32000Hz, we upsampled the decoded compressed speech and we kept the initial version of speech to encode.*

**: *From the course (Lecture 14), note that the likelihood distortion and cepstral distance are similar. Hence the log spectral distance (derived from the truncated cepstral distance) is sufficient to measure the spectral distortion.*

Table 4 - compression results for chirps:

Features	Values	Comments
Number of chirps	9	*3 chirps per band cluster (4000Hz, 800Hz, 12000Hz)*
Number of coefficients per	4	*The amplitude, the*

chirps		*frequency, the phase amplitude and the frequency amplitude*
Nb bits to encode a parameters	16 bits	The values of parameters are between *[−32,768 to 32,767]* .
Total number of bit to encode the chirps	72 Bytes	Nb_bytes = (9*4*16)/8 (576 bits)

Result analysis:

From outcomes, we constat that both found spectral distortions are low, and the SQNR is more interesting for a *32000Hz* sampled frequency.
Furthermore, Notice that the trend of results is not similar according to the sampled frequency. With a *32000Hz* sampled frequency, the compressed signal is fewly distorted but it has a more important quantized noise. Otherwise, with a 800Hz, the compressed signal is more distorted, however, it shows less quantized noise.
Moreover, the coefficient approach to encode chirps is in effect more effective than compressed an entire signal. As a matter of fact, it is not surprising that encoding only 9 values offers la lighter compressed file than *1,280,000!* We will able to reconstruct the exact chirps signal, and there are not losses.

Therefore, we could reproduce a close decoded signals to the initial one.

V) Future Improvements

There are a couple of aspects in the project we felt could be improved. Firstly, when we modelled the chirp signals and reconstructed them, there were a couple of ways we could do that. Some potential options were autocorrelation and spectral distance measures. We chose to do the instantaneous frequency comparison, which otherwise might have given us better results.

Secondly, we were not able to subtract the constructed chirp signals from the given signal directly. This leads to the question of doubt, if the constructed chirp signal is the right one or not? Surprisingly, when the FFT of the each separate frame of the three different bands was studied, we found the chirps did not match the FFT. We assumed in this project that whatever we modelled the chirp as right, but conversely it is not completely right. We assumed the difference as noise which can be improved. We used LPC filtering of the signal which can be completely avoided if the chirp signals were modelled right. We studied various kinds of chirp, but were not successful in getting the good fit. During the study of these various chirps we found that the chirp signal has a time varying phase and not a constant one. The chirp signal seemed to be of something of this sort given by the equation:

$$\pm cos(4000\pi t + 40\, cos(80\pi t - \pi/2))$$

This is yet an unsolved mystery for us. Sadly, though we have worked hard on this and were unable to crack it, we call it "*the CHIRP nightmare*"!

Thirdly, we could compress the file better if we could get rid of all the chirp signals. We did not get a clear speech in the end, which can be achieved by better approaches. Compressing the clear speech would be an easier task and give a highly compressed file size.

VI)References and used opensource programs

[1] *LPCdemo.m* provided *from the correction of HW4*

[2] the G.729 given in class: the github of *opentelecoms-org:* https://github.com/opentelecoms-org/codecs/tree/master/g729/ITU-samples-200701/Soft/g729

[3] MathWork file exchange Matlab