

Frequency Domain Methods for Coding the Linear Predictive Residual of Speech Signals

Pablo Pérez Zarazaga

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 04.07.2017

Thesis supervisor:

Prof. Tom Bäckström

Thesis advisor:

M.Sc. Goran Markovic

Author: Pablo Pérez Zarazaga

Title: Frequency Domain Methods for Coding the Linear Predictive Residual of Speech Signals

Date: 04.07.2017

Language: English

Number of pages: 7+72

Department of Signal Processing and Acoustics

Professorship: Speech Communication Technology

Supervisor: Prof. Tom Bäckström

Advisor: M.Sc. Goran Markovic

The most frequently used speech coding paradigm is ACELP, famous because it encodes speech with high quality, while consuming a small bandwidth. ACELP performs linear prediction filtering in order to eliminate the effect of the spectral envelope from the signal. The noise-like excitation is then encoded using algebraic codebooks. The search of this codebook, however, can not be performed optimally with conventional encoders due to the correlation between their samples. Because of this, more complex algorithms are required in order to maintain the quality. Four different transformation algorithms have been implemented (DCT, DFT, Eigenvalue decomposition and Vandermonde decomposition) in order to decorrelate the samples of the innovative excitation in ACELP. These transformations have been integrated in the ACELP of the EVS codec. The transformed innovative excitation is coded using the envelope based arithmetic coder. Objective and subjective tests have been carried out to evaluate the quality of the encoding, the degree of decorrelation achieved by the transformations and the computational complexity of the algorithms.

Keywords: Speech Coding, EVS, ACELP, Transform Coding, Vandermonde decomposition, Eigenvalue decomposition, Discrete Cosine transform, Discrete Fourier Transform

Preface

I would like to thank Prof. Tom Bäckström for giving me the opportunity to develop my master's thesis under his supervision and for his support and ideas.

I would also like to express my gratitude to my advisor Goran Markovic for his help along the duration of this thesis, his valuable comments and his patience, which allowed me to successfully complete this thesis work.

Finally, but not less important, I would like to thank my parents, for supporting me all these years and encouraging me to work in what I most like.

Otaniemi, 04.07.2017

Pablo Pérez

List of Figures

1	Block diagram of the EVS encoder.	3
2	Block diagram of the EVS decoder.	4
3	Representation of the vocal tract showing its most significant parts. "Bäckström, T., Lecture material from ELEC-E5500 Speech processing, Aalto University, 2015", Copyright Tom Bäckström, reproduced with permission.	6
4	Tube model of the vocal tract	8
5	Spectrum of the phoneme /a/. In blue: The spectrum of the speech signal. In red: Spectrum of the Linear Prediction coefficients with order 24	9
6	Algorithm of the Levinson-Durbin recursion	11
7	Block diagram of the ACELP encoder	12
8	Block diagram of the ACELP decoder.	13
9	Block diagram of a TCX encoder	14
10	Block diagram of a TCX decoder	14
11	Structure of the MDCT-based TCX encoder in EVS	15
12	Structure of the MDCT-based TCX decoder in EVS	15
13	Block diagram of an N-point DFT divided into $2 \frac{N}{2}$ -point DFTs.	18
14	Block diagram of an 8-point FFT.	18
15	Frequency bands at which the DFT takes values compared to the values of the Vandermonde decomposition in the same subframe. Blue circles: DFT bands. Red crosses: Vandermonde decomposition.	23
16	ACELP block diagram in detail.	26
17	Block diagram of the target vector computation.	28
18	Magnitude response of a Pre-emphasis filter with $\alpha = 0.3$	30
19	Magnitude response of a De-emphasis filter with $\alpha = 0.3$	30
20	Spectrogram of signal encoded with Vandermonde transformation. Left: Using tilt filter. Right: No tilt filter.	31
21	Spectrum of a frame encoded using the tilt filter and without using it	31
22	Integration of the tilt filter in the ACELP structure	32
23	Block diagram of the frequency-domain CELP (FD-CELP) encoder using DFT	33
24	Block diagram of the FD-CELP decoder using DFT	34
25	Block diagram of the FD-CELP encoder using DCT	35
26	Block diagram of the FD-CELP decoder using DCT	36
27	Block diagram of the FD-CELP encoder using Vandermonde decom- position	38
28	Block diagram of the FD-CELP decoder using Vandermonde decom- position	38
29	Average Percentage of Variance (<i>PoV</i>) and error with respect to MatLab functions. Blue Line: Percentage of information loss based on the <i>PoV</i> . Red Line: Error of each eigenvalue with respect to the MatLab function.	40

30	Block diagram of the FD-CELP encoder using Eigenvalue decomposition	41
31	Block diagram of the FD-CELP decoder using Eigenvalue decomposition	41
32	Quantiser structure of the envelope based encoder proposed in [1]	43
33	New quantiser structure using the encoder instead of estimating the probabilities	44
34	Perceptual SNR of english male noisy speech. Top: Time representation of the signal. Bottom: Perceptual SNR over time.	46
35	Autocorrelation matrix of a frame containing a voiced sound	51
36	Transformed autocorrelation matrices of a voiced sound. Left: Eigenvalue decomposition. Right: Vandermonde decomposition	51
37	Transformed autocorrelation matrices of a voiced sound. Left: DFT. Right: DCT	52
38	Autocorrelation matrix of a frame containing an unvoiced sound	52
39	Transformed autocorrelation matrices of an unvoiced sound. Left: Eigenvalue decomposition. Right: Vandermonde decomposition	53
40	Transformed autocorrelation matrices of an unvoiced sound. Left: DFT. Right: DCT	53
41	Autocorrelation matrix of a frame containing silence	54
42	Transformed autocorrelation matrices of silence. Left: Eigenvalue decomposition. Right: Vandermonde decomposition	54
43	Transformed autocorrelation matrices of silence. Left: DFT. Right: DCT	55
44	Average POLQA results at 32kbps evaluated on 972 samples	56
45	Average POLQA results at 24.4kbps evaluated on 972 samples	57
46	Average POLQA results at 16.4kbps evaluated on 972 samples	58
47	Bitrate against MOS representation of the different algorithms	58
48	Average MUSHRA results for 32kbps encoding using 8 expert listeners.	59
49	MUSHRA results represented with respect to the EVS-TCX encoding. 32kbps encoding using 8 expert listeners	60
50	Average MUSHRA results for 16kbps encoding using 8 expert listeners	61

List of Tables

1	Classification of unvoiced sounds	7
2	Voiced phonemes and the frequencies of their first three formants	9
3	Average perceptual SNR of the different test files processed with the different algorithms	45
4	Initial complexity study	47
5	Complexity values with silence detection in the encoder	48
6	Complexity values assuming an optimized gain estimation	48
7	Execution time of the original implementation of the algorithms encoding an 8 seconds file	49
8	Weights for the different operations analysed in the complexity study	71

Contents

Abstract	ii
Preface	iii
Contents	vi
Symbols and abbreviations	vii
1 Introduction	1
2 State of the Art	3
2.1 Enhanced voice services (EVS) codec	3
3 Theoretical Background and Motivation	6
3.1 Voice Generation System	6
3.2 Linear Prediction	7
3.2.1 Autocorrelation method and Levinson-Durbin recursion	10
3.3 ACELP	11
3.4 TCX	13
3.5 Transformations	15
3.5.1 Discrete Fourier Transform	16
3.5.2 Discrete Cosine Transform	19
3.5.3 Singular Value Decomposition	19
3.5.4 Vandermonde Decomposition	20
4 Implementation	25
4.1 Integration in EVS	25
4.2 Target Vector	27
4.3 Target Update	29
4.4 Pre-emphasis, de-emphasis and tilt filter	29
4.5 Residual transformation	32
4.5.1 DFT	32
4.5.2 DCT	34
4.5.3 Vandermonde decomposition	36
4.5.4 Eigenvalue decomposition	39
4.6 Quantisation and encoding	41
4.6.1 Envelope based arithmetic encoder	42
5 Testing and Results	45
5.1 Complexity Study	46
5.2 Entropy study	49
5.3 POLQA test	55
5.4 MUSHRA Test	59
6 Discussion and Future Lines of Study	63

Symbols and abbreviations

Operators

M^T	Transpose of the matrix M without complex conjugation
M^H	Complex conjugate transpose of matrix M
$ M $	Determinant of the matrix M
a^*	Complex conjugate of scalar a
$ a $	Absolute value of scalar a

Abbreviations

ACELP	Algebraic Codebook Excited Linear Prediction
AMR-WB	Adaptive Multi Rate Wideband (codec)
CELP	Codebook Excited Linear Prediction
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DSP	Digital Signal Processing
DTX	Discontinuous Transmission
EVS	Enhanced Voice Services
FB	Full Band
FDNS	Frequency Domain Noise Shaping
FFT	Fast Fourier Transform
HP	High Pass
IFFT	Inverse Fast Fourier Transform
IGF	Intelligent Gap Filling
LP	Low Pass
LPC	Linear Prediction Coding
LTP	Long Time Predictor
MDCT	Modified Discrete Cosine Transform
MDST	Modified Discrete Sine Transform
NB	Narrow-Band
PCA	Principal Component Analysis
PoV	Proportion of Variance
SNR	Signal to Noise Ratio
SVD	Singular Value Decomposition
SWB	Super Wide-Band
TCX	Transform Coded Excitation
TNS	Temporal Noise Shaping
VoIP	Voice over IP
VoLTE	Voice over LTE
WB	Wide-Band
WMOPS	Weighted Millions of Operations Per Second

1 Introduction

In a world where everybody lives connected via the Internet, people consume a larger amount of data every day in order to communicate between them. However, bandwidth is a limited resource which will restrict the amount of transmitted data and, as a result, the speed of this transmission. Compression techniques have become increasingly important over the years, as they offer the tools to make the most of digital communications, while consuming fewer resources. More specifically, in real time communications such as telephony, compression techniques provide a higher quality for the available resources.

Speech coding techniques are becoming high in demand due to the digitalization of voice transmission. The last mobile telephony standards have also got rid of the dedicated channel for voice transmission and included it in the conventional data network. These progressive updates have proven that it is necessary to reduce the bandwidth consumed by the voice transmission maintaining a high quality of speech.

One of the most common techniques used in speech coding today is Algebraic Codebook Excited Linear Prediction (ACELP)[2]. This technique is based on a parametrization the speech signal using a linear prediction filter (LPC) to model its spectral envelope and a long time predictor (LTP) to extract the fundamental frequency of the voice. The remaining signal after this processing is commonly known as the residual or excitation, which is represented in a codebook that contains a large number of possible excitations. This way, the excitation can be encoded as the position of the most similar excitation in the codebook, in terms of an error minimization criterion.

The problem that ACELP faces is that an exhaustive search of the optimum codebook is a tedious task, and by using a brute force search, this can turn into a computationally heavy task when the bitrate grows, making the size of the codebook too large. It is possible to devise multiple algorithms to perform this task more efficiently [3, 4], however, as [5] states, these algorithms are not optimal due to the correlation between the values of the excitation. This translates in a compromise between the quality of the quantization and the complexity of the process.

In [6], several methods are proposed that transform the linear prediction residual into a different domain to exploit the properties of the transformation in the encoding process. If we apply a transformation that decorrelates the samples of the residual, it would be much easier to encode the excitation, simplifying the codebook search algorithm.

The objective of this thesis is to implement some algorithms to transform the residual of the LP and integrate them in a functional codec (EVS). The performance of the algorithms will be compared to analyse how they behave using ACELP as a reference.

Section 2 focuses on the current state of speech codecs and their applications. An overview of the latest speech coding techniques is presented, as well as how they are integrated in the state-of-the-art speech codecs. The different components of the new EVS codec will be briefly explained in order to establish its relationship with the work conducted for this thesis.

In section 3, the theory behind these speech coding algorithms will be examined, thus providing a deeper insight of the problem faced during this thesis work. The different transformations used to solve this problem will be also be described, providing an intuitive point-of-view to aid the readers interpretation of the solutions.

The work of the thesis is to replicate several transformation algorithms using the programming language C integrating them in the EVS codec structure. Therefore, section 4 will go deeper into the actual implementation of the codec and which parts will need to be modified in order to integrate the new algorithms.

Finally, section 5 explains the different tests that were performed in order to evaluate the performance of the various algorithms that were implemented. To evaluate the algorithms' quality, four different tests were performed. Firstly, some objective measurements were conducted in order to evaluate the quality of the resulting signals, complexity of the algorithms and the quality of the decorrelation seen as an entropy estimation.

However, a subjective measurement is also necessary in order to see how humans perceive the quality of the compressed signal for each of the algorithms. For that purpose, a MUSHRA test was carried out in order to get a final quality measurement that is then compared with the other algorithms.

As stated before, the goal of this thesis is to evaluate different transformation algorithms that decorrelate the signal in order to perform a more efficient codebook search for the linear prediction residual. The entropy measurements will help to establish a first impression about the final quality of the algorithms, considering that a lower correlation between the samples of the residual should provide a better encoding. The simplification in the codebook search would be useless if the computation required by the transformation algorithms exceeded the complexity of the ACELP implementation. The complexity study will provide results in order to compare the complexity of the different algorithms. Finally, the quality measurements will be used to evaluate the effect of the transformations on the encoding and will help to establish a relationship between the decorrelation of the residual and the final quality provided by the encoder.

2 State of the Art

This section contains an introduction to the main systems that are used today in speech coding. Linear Prediction based algorithms have been shown to yield high performance when analysing the speech properties and, therefore, ACELP has become the main paradigm present in speech codecs.

2.1 Enhanced voice services (EVS) codec

The codec for EVS [7, 8] was recently standardized by the 3GPP at the end of 2014. This standard brought about new functionalities that are focused on the improvement of real time communications systems. This new standard represents the new generation after the Adaptive Multirate Wide Band (AMR-WB+) codec, providing a better quality in narrow band (NB) and wide band (WB) that were already treated in AMR-WB+, but also introducing a new option for super wide band (SWB) and full band (FB) speech. This standard was selected to work for Voice over LTE (VoLTE). The structure of this codec is shown in figures 1 and 2.

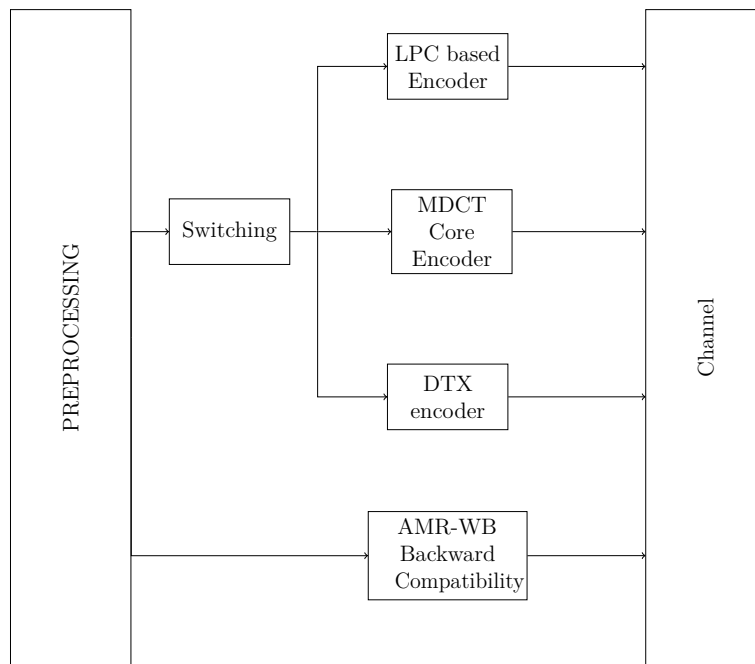


Figure 1: Block diagram of the EVS encoder.

One of the significant improvements provided by the EVS is the low delay switching between speech and audio coding, which provides a better quality when working with noisy speech or mixed signals where music is played in addition to the speech. This is accomplished by including three different types of encoder techniques:

- ACELP based coding for speech signals, adding some improvements to the features that AMR-WB provided.

- Modified Discrete Cosine Transform (MDCT) based coding for signals that contain music.
- Discontinuous Transmission (DTX) coding for background noise in absence of speech.

As figure 1 shows, the switching system before the encoding algorithms has to decide whether the frame contains speech, music or noise. However, the main feature of this switching system is that it manages to go from one algorithm to another making the transition inaudible. The combination of these algorithms when they perform best is one of the reasons why EVS provides a great quality for mixed signals at low bitrates.

The pre-processing block includes some algorithms to prepare the signal for the encoding. These algorithms include, among others, a high-pass (HP) filter at 20 Hz to eliminate low frequency noise and the resampling of the signal to the sampling rate supported by the encoder (12.8 kHz or 16 kHz). Some of these preprocessing tools extract information of the signal for the switching task, like a signal activity detector and the extraction of the LPC coefficients that are used in the ACELP and MDCT core encoders.

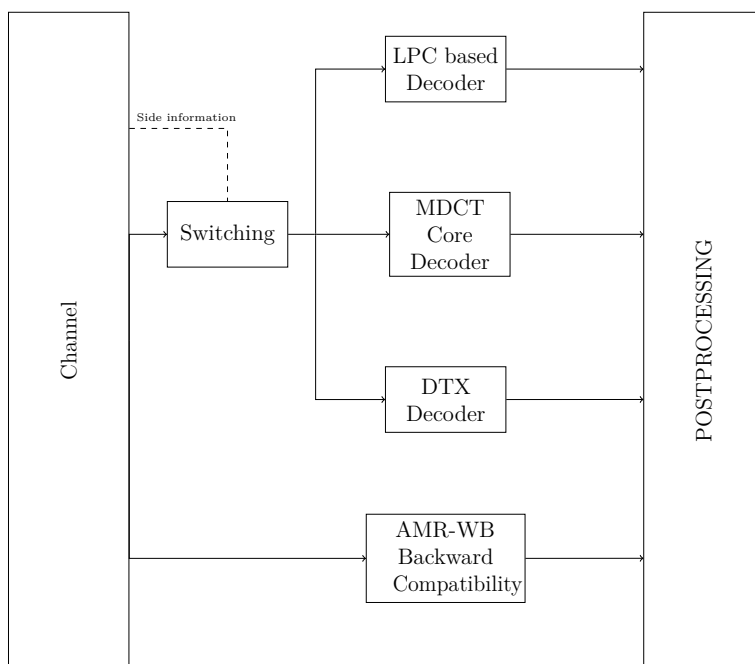


Figure 2: Block diagram of the EVS decoder.

The ACELP core used for speech signals inherits its characteristics from the AMR-WB standard [9]. However it adds some improvements with respect to the previous version, such as the use of different dedicated LPC coding modes for different types of speech, compatibility with a higher sampling rate (16 kHz against the previous 12.8 kHz), the use of formant enhancement and a bass post-filter.

The MDCT core encoder used for music and noisy signals transforms the signal using MDCT [10] to transform the signal before encoding it. This mode includes three versions of MDCT based encoding, Low-Rate/High-rate High Quality-MDCT coding [11], an advanced version of G.719 [12] and MDCT-based TCX [13]. In this thesis work, the working mode of the EVS is forced to use the TCX version, based on Transform Coded Excitation (TCX) techniques [14]. An entropy encoder is then used to quantize and encode the transformed signal.

The decoder structure in EVS is very similar to the encoder (figure 2). The side information provided by the encoder contains the type of coding used; therefore, the switching mechanism only has to read the bitstream and select the corresponding decoder.

The post-processing block compensates for some of the unwanted side-effects of the processing applied in the encoder. ACELP post-processing algorithms include, amongst others, a bass post-filter, formant enhancement and resamples the signal to the desired sampling rate.

The mixed encoding scheme also allows the codec to work at very low bitrates, reaching 5.9 kbps depending on the speech signal. In [8, 13], the performance of this codec is shown with respect to the AMR-WB+, where the improvement is clearly noticeable. When operating on pure speech signals, EVS working in WB mode at 9.6 kbps yields better results than AMR-WB+ at 23.85 kbps, and it outperforms AMR-WB+ considerably at higher bitrates. The result is similar when the signal is mixed with music.

3 Theoretical Background and Motivation

ACELP coders are widely used nowadays because they provide high quality encoding with small bit consumption. The problem that they are presented with, is how best to search the codebook library, in order to obtain the element that yields the smallest error criterion. However, the samples in the LPC residual are highly correlated with each other. This renders many efficient search algorithms suboptimal, which results in an exhaustive search evaluating every possible codebook as the only optimal algorithm.

Nowadays, speech coding systems try to model the manner in which humans generate speech signals. CELP based codecs try to separate the spectral shaping effect of the vocal tract from the speech signal modelling it as a FIR filter, which can be then easily encoded. In order to understand how codecs work, an overview on the human speech production system can be helpful.

3.1 Voice Generation System

The process of speech generation starts in the lungs, which push the air, generating an airflow. When the air travels through the larynx, the vocal folds tighten and vibrate generating a quasi-periodic series of pulses. The vibration frequency of the vocal folds is known as the fundamental frequency. The rest of the elements in the vocal tract work as obstructions to the airflow in order to shape the signal that exhibits the desired sound.

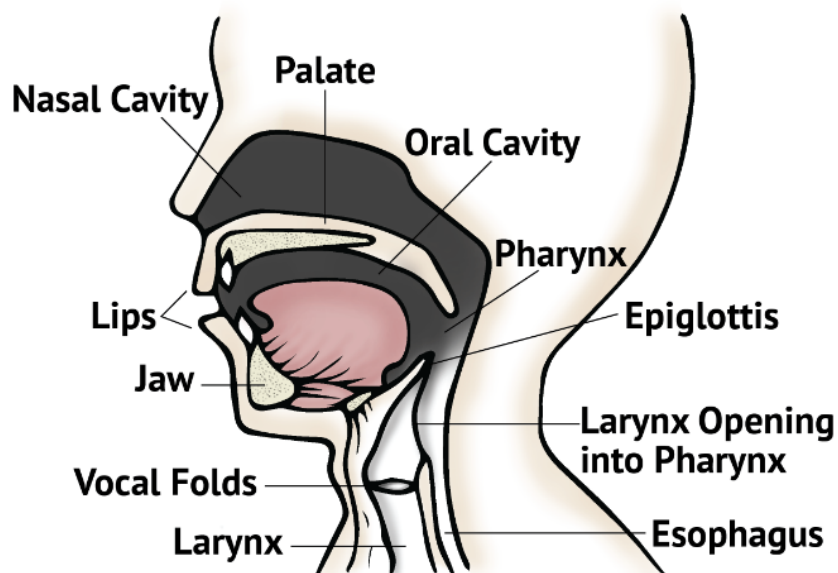


Figure 3: Representation of the vocal tract showing its most significant parts. "Bäckström, T., Lecture material from ELEC-E5500 Speech processing, Aalto University, 2015", Copyright Tom Bäckström, reproduced with permission.

It is possible to classify sounds in two different groups depending on the way they are generated, voiced and unvoiced.

- For voiced sounds, the vocal folds tighten so they can vibrate when the air travels through. This produces the pseudo-periodic series of pulses which is perceived by humans as a tonal sound. The vocal tract does not close completely at any point, it just adopts different ways to reshape the airflow without any obstacles inside the tract. The different sounds are then defined by the frequencies that resonate in the system as shown in section 3.2.
- Unvoiced sounds are generated by loose vocal folds that generate a turbulent airflow which then gets obstructed by the constriction of some part of the vocal tract. The different obstacles that the airflow can find on his way are the tongue, teeth and lips.

The production of voiced sounds and their classification are explained in section 3.2. Therefore, a revision of the unvoiced sounds will be presented here.

Sound	Production	Example
Plosive	Airflow completely stopped for a moment and abruptly released.	p erson, c at, h at
Nasal	Mouth partially or completely closed. Airflow travels through the nose.	n ose, m ore
Fricative	Small opening in the vocal tract. Noisy sound is generated.	f ace, h alf
Affricative	Begin with a stop and turns into a fricative sound	ch ase
Tremulant	Single or repeated constrictions in the vocal tract.	r ace
Liquid	Similar to fricatives, but less noisy.	j oint, w ater
Lateral	The tongue obstructs the airflow. The air travels on both sides of the tongue	l ateral

Table 1: Classification of unvoiced sounds

As stated before, unvoiced sounds are produced when there is a partial or complete obstruction of the vocal tract preventing the airflow to travel naturally through it. These sounds can be classified depending on the obstruction produced and which section of the vocal tract produces it. Table 1 shows a simple classification of these sounds and a small description of how they are produced.

3.2 Linear Prediction

As stated in section 3.1, when humans produce voiced sounds, the vocal folds generate a tonal sound which travels through the components of the vocal tract. This process can be modelled as a succession of tubes of different lengths and diameters which

behave as a waveguide. As [15] shows, when the width of the waveguide changes, some of the signal's energy gets reflected back. This happens with every diameter change and can be represented as a sequence of reflection coefficients. Figure 4 shows a simple representation of this model, every intersection between tubes can be interpreted as a reflection/transmission coefficient that will characterize the vocal tract.

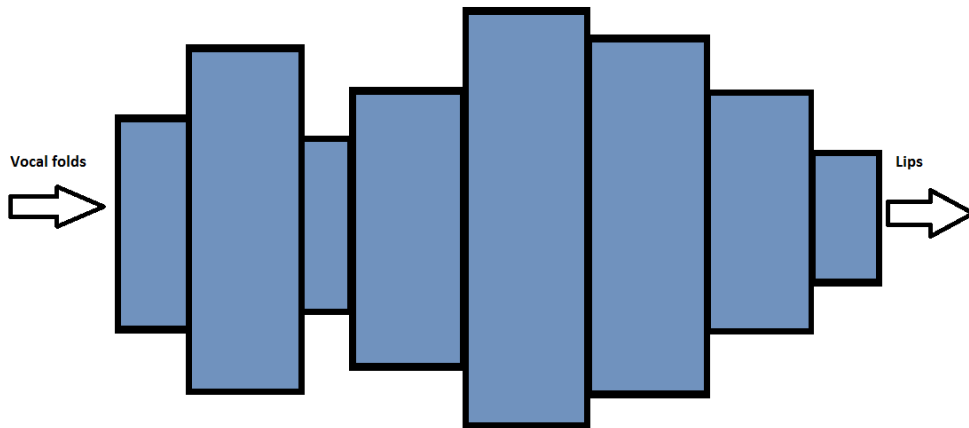


Figure 4: Tube model of the vocal tract

These reflections are dependent on the the frequency of the signal and the result is a shaping in the spectrum of the voice signal. The dominant frequencies after the effect of the vocal tract are called formants, and they define the sound of the phoneme pronounced. The frequencies of different voiced phonemes can be seen in table 2. It is important not to confuse the formants, which define the phoneme that is being pronounced, with the fundamental frequency of the sound, which represents the frequency at which the vocal folds vibrate and change from one person to another. This sequence of reflection coefficients can be represented with Lineal Prediction coefficients and work as a filter where the frequency response is the envelope of the signal. [16] defines linear prediction as a mathematical operation that models a signal in order to predict the value of the sample $x(n)$ using a linear combination of the N most recent past samples. This can be expressed as

$$\hat{x}_N(n) = - \sum_{i=1}^N a_i x(n-i) \quad (1)$$

where a_i are the linear prediction coefficients, the whole operation can be considered as an FIR filter, $A_N(z)$, defined as

Phoneme	F_1 (Hz)	F_2 (Hz)	F_3 (Hz)
/iy/	270	2290	3010
/oo/	300	870	2240
/i/	390	1990	2550
/u/	440	1020	2240
/er/	490	1350	1690
/uh/	520	1190	2390
/e/	530	1840	2480
/ow/	570	840	2410
/ae/	660	1720	2410
/a/	730	1090	2440

Table 2: Voiced phonemes and the frequencies of their first three formants

$$A_N(z) = 1 + \sum_{i=1}^N a_i z^{-i}. \quad (2)$$

For large values of N , most of the spectral information of $x(n)$ is contained in the linear prediction coefficients and the resulting prediction error is a pseudo-random signal with an almost white spectrum, this error is commonly referred as the linear prediction residual. This is the characteristic that LPC based codecs exploit in order to compress the data. When filtering the signal with the linear prediction filter defined in (2), it is possible to obtain the residual signal, such that only the LPC coefficients and the residual signal need to be encoded.

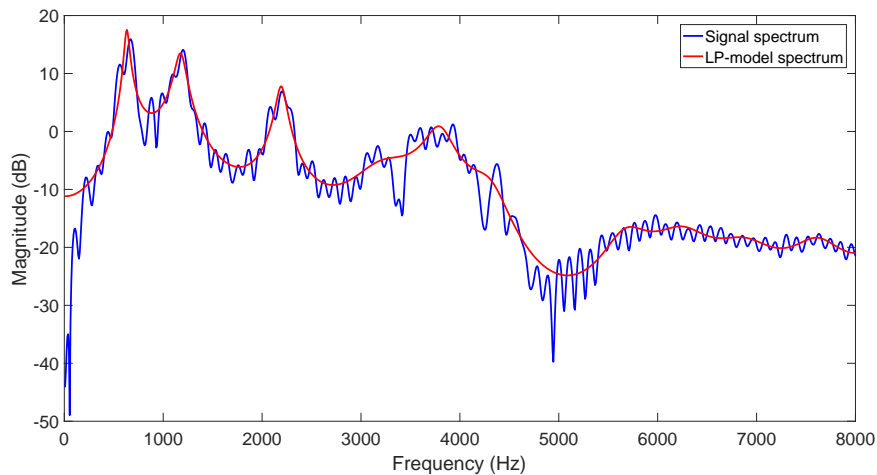


Figure 5: Spectrum of the phoneme /a/. In blue: The spectrum of the speech signal. In red: Spectrum of the Linear Prediction coefficients with order 24

Figure 5 shows the spectrum of the phoneme /a/ pronounced by a male voice in blue. The red line represents the spectral envelope of the sound generated by a 24th order Linear Prediction filter.

One of the most common methods to estimate the LPC coefficients of a signal is the autocorrelation method.

3.2.1 Autocorrelation method and Levinson-Durbin recursion

The error signal, e , can be generated by first estimating a sample $\hat{x}(n)$ using linear prediction and then expressing it as

$$e(n) = x(n) - \hat{x}(n). \quad (3)$$

To estimate the optimum LPC coefficients it is necessary to select them so that they minimize this error. In the case of the autocorrelation method, the expected value of the squared error $E[e^2(n)]$ should be minimized. Defining the autocorrelation of a signal $x(n)$ as:

$$R(i) = E[x(n)x(n-i)] = \sum_{n=-\infty}^{\infty} x(n)x(n-i) \quad (4)$$

Writing $\hat{x}(n)$ in terms of $x(n)$ as in equation 1 and using the autocorrelation formula, the error minimization equation can be defined as:

$$\sum_{i=1}^N a_i R(j-i) = -R(j) \quad \text{For } 0 \leq j \leq N. \quad (5)$$

Considering that $R(i)$ is an even function $R(i) = R(-i)$, equation 5 represents a system of equations whose matrix form is:

$$\begin{bmatrix} R(0) & R(1) & R(2) & \cdots & R(N-1) \\ R(1) & R(0) & R(1) & \cdots & R(N-2) \\ R(2) & R(1) & R(0) & \cdots & R(N-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R(N-1) & R(N-2) & R(N-3) & \cdots & R(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} R(1) \\ R(2) \\ R(3) \\ \vdots \\ R(N) \end{bmatrix} \quad (6)$$

Since the autocorrelation matrix is a Toeplitz matrix, the solution of the system of equations can be found with $O(n^2)$ complexity using the Levinson-Durbin recursive algorithm.

Figure 6 explains the Levinson-Durbin algorithm.

1. Estimate the first LPC coefficient.
2. Enter the loop.
3. Estimate the next LPC coefficient as the prediction error using the previous coefficients.
4. Correct the previous coefficients.
5. Next iteration until all the coefficients are calculated.

Data: Levinson-Durbin recursive algorithm
Result: LPC Coefficients (a_i) of the signal $x(n)$

$$E_0 = R(0);$$

$$a_{11} = k_1 = \frac{R(1)}{E_0};$$

$$E_1 = E_0(1 - k_1^2);$$

for $j = 2:N$ **do**

$k_j = \frac{R(j) - \sum_{i=1}^{j-1} a_{i,(j-1)} R(j-1)}{E_{(j-1)}};$
$a_{j,j} = k_j;$
for $i=1:(j-1)$ do
$a_{i,j} = a_{i,(j-1)} - k_j a_{(j-i),(j-1)};$
$E_j = E_{j-1}(1 - k_j^2);$
end

end

Figure 6: Algorithm of the Levinson-Durbin recursion

3.3 ACELP

Algebraic Codebook Excited Linear Prediction (ACELP) [2, 3, 17] is a speech coding paradigm that derives from CELP, in which the residual of the linear prediction analysis is represented with an entry in a predefined codebook. This entry is then transmitted to the decoder, which extracts the corresponding codebook to be used as excitation. ACELP uses this same structure including algebraic codebooks [2] to represent the LPC residual, which meant a significant step obtaining a substantial reduction of the codebook.

In figure 7, a more precise representation of an ACELP encoder can be seen:

1. The input signal is analysed in order to extract the LPC coefficients of the frame. These coefficients are quantized, as they also have to be transmitted to the decoder.
2. The stored previous excitation is then filtered through the LPC coefficients.
3. This reconstructed signal is subtracted from the input and the error signal resulting, known as innovation, will be encoded and transmitted.
4. The perceptual weighting filter shapes the innovation signal according to the characteristics of the human hearing, in order to move the coding noise to frequency bands where it will be more difficult to be perceived. The representation of the excitation in this perceptual domain is called target vector.
5. The codebook search is performed using an error minimization criterion in order to find the codebook with the highest resemblance to the target.

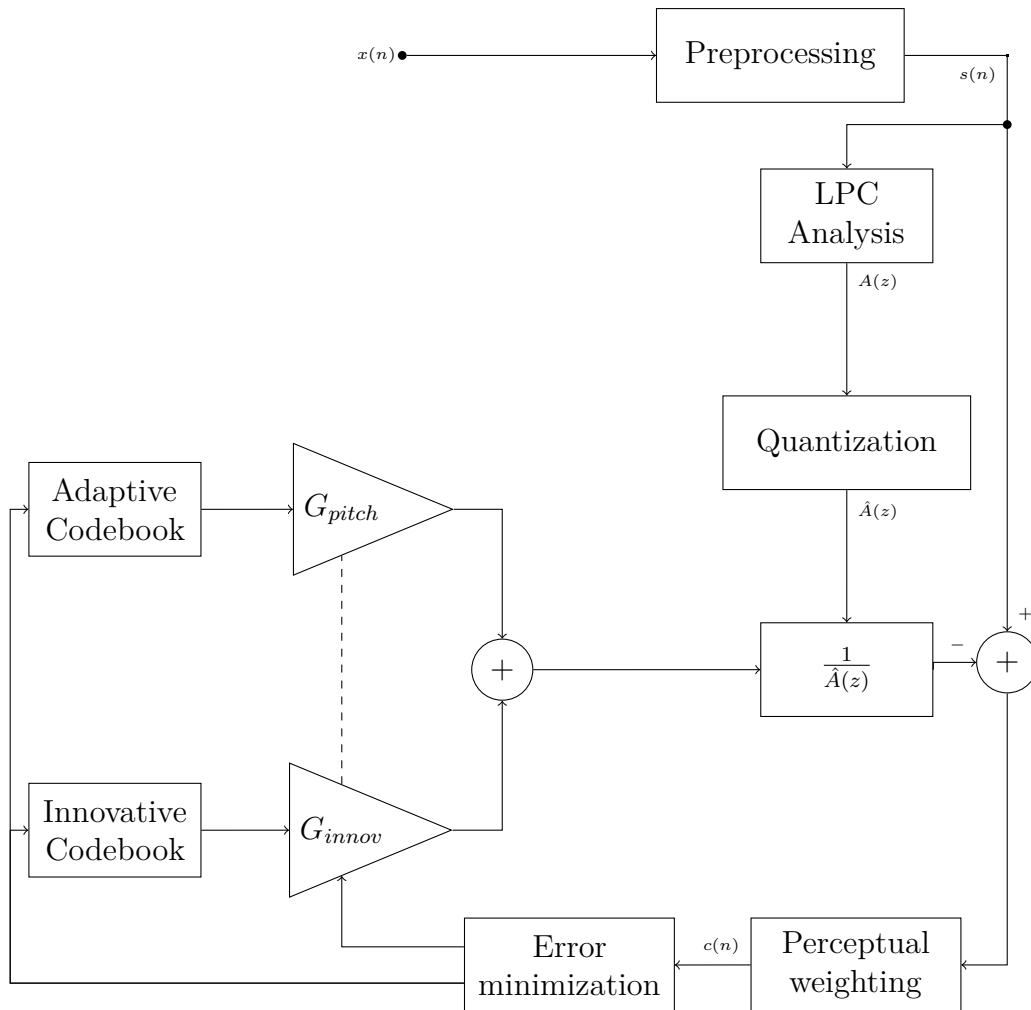


Figure 7: Block diagram of the ACELP encoder

6. Two different codebooks are used to represent the signal. One of them (Adaptive Codebook) contains the tonal components that define the pitch information in voiced sounds. The other is an algebraic codebook that represents the unvoiced sounds.
7. Both codebooks are then scaled by their respective gains and added to be used in the next frame's processing.

The data that is finally included in the transmitted bitstream are:

- The quantized LPC coefficients.
- The gains of the codebooks.
- The pitch delay for the adaptive codebook.
- The index of the innovative codebook.

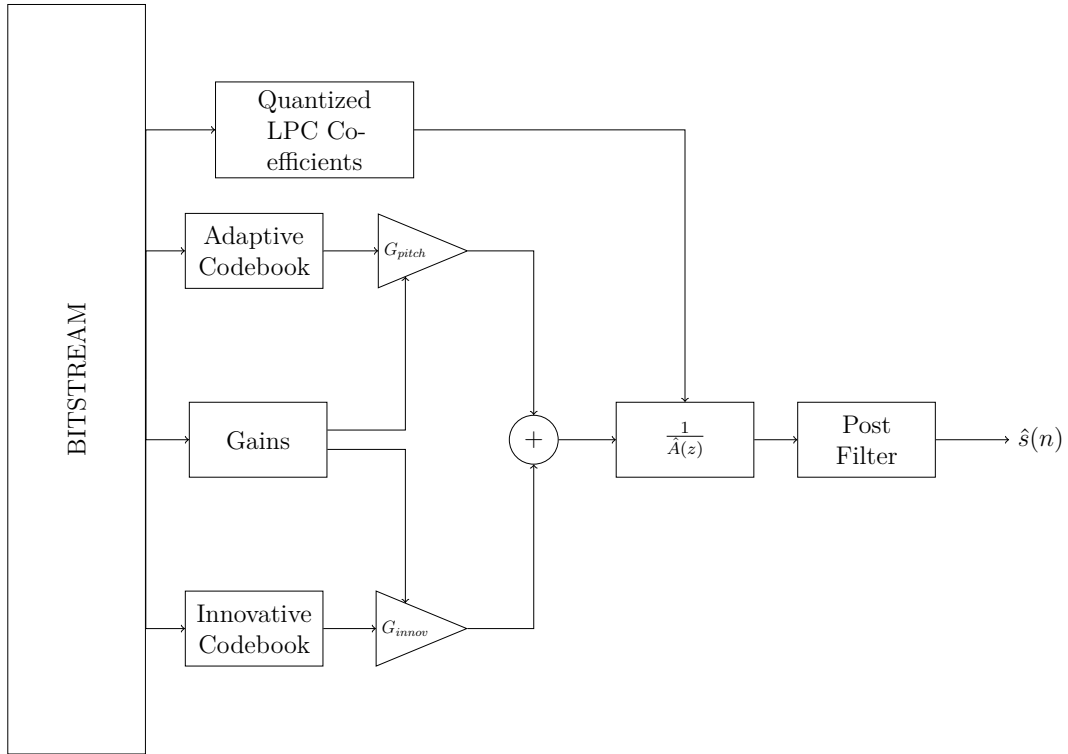


Figure 8: Block diagram of the ACELP decoder.

Figure 8 shows the ACELP decoder structure, which is much simpler than the encoder. The transmitted bitstream contains the information of the codebook gains and the quantized LPC coefficients as well as the codebook index. The decoder structure is very similar to the last part of the encoder, where the signal is decoded to obtain the innovation:

1. The pitch delay and the innovative codebook index are extracted from the bitstream and the corresponding codebooks will form the frame excitation.
2. Each codebook is scaled by its own gain and both are added to form the total excitation.
3. This excitation is filtered by the quantised LPC coefficients to obtain the output signal.
4. This output signal is usually taken through a post-processing filter in order to eliminate unwanted artefacts.

3.4 TCX

The transformation of an audio signal to an alternative domain, is a common first step when analysing and processing the signal, as the new domain might be able to take advantage of some features that the original domain can not provide. As shown in [18, 19, 20, 21], there are many authors that apply these techniques into audio

coding. These kind of techniques are based on Transform Coding (TCX) shown in [6, 14]. These coding techniques are based on transforming the residual of the LPC filtering in order to use the spectral properties of this residual in the encoder.

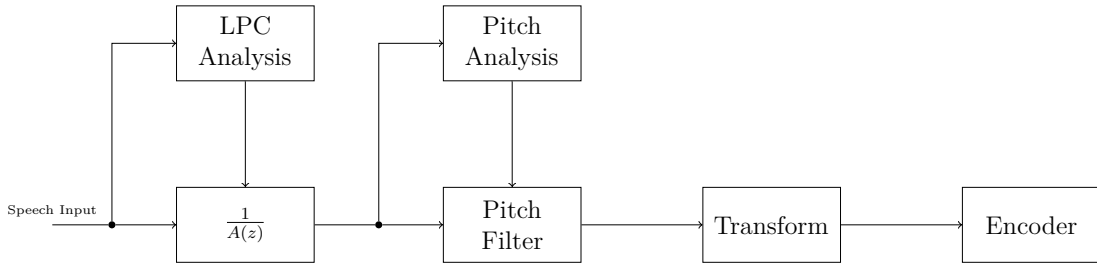


Figure 9: Block diagram of a TCX encoder

Figure 9 shows the basic structure of the TCX encoder. Firstly, the LPC coefficients are used to extract the spectral envelope information of the speech signal, then the pitch analysis removes the fundamental frequency information remaining in the residual. Finally, the resulting signal is transformed to the frequency domain to be encoded. The most common transformation used in this method is the DCT, as it is more robust to compression artefacts.

Analogous to the encoder part, figure 10 represents the structure of a simple TCX decoder. The information of the excitation contained in the bitstream is in the transform domain, therefore the decoding algorithm needs to invert the transformation in order to reconstruct the LPC residual. The influence of the pitch and the spectral envelope are then added again and finally a post-filter reduces the unwanted artefacts, as described in section 3.3.

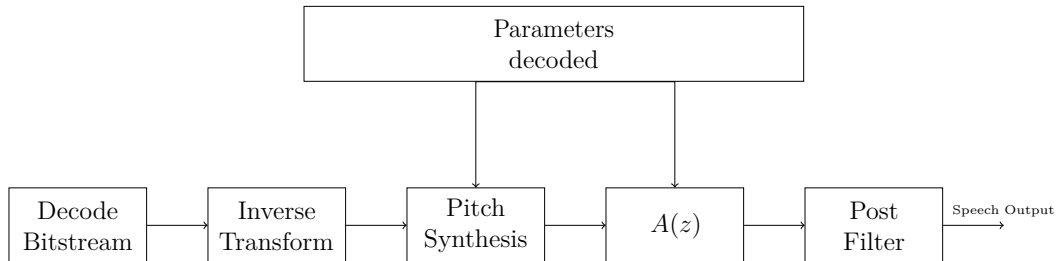


Figure 10: Block diagram of a TCX decoder

The algorithms implemented in this thesis follow the scheme of TCX as they transform the target vector of the innovative codebook into a different domain where the encoding will be more efficient. However, as section 2 shows, the EVS codec already implements a TCX based encoder [13] so its functioning will be explained in order to gain an understanding of the difference between it and the objective of this thesis.

As can be seen in figure 11, the LPC coefficients and the pitch information are extracted from the signal; however, this information is not removed from it. Working at low bitrates, the LPC coefficients and the pitch information are used to determine

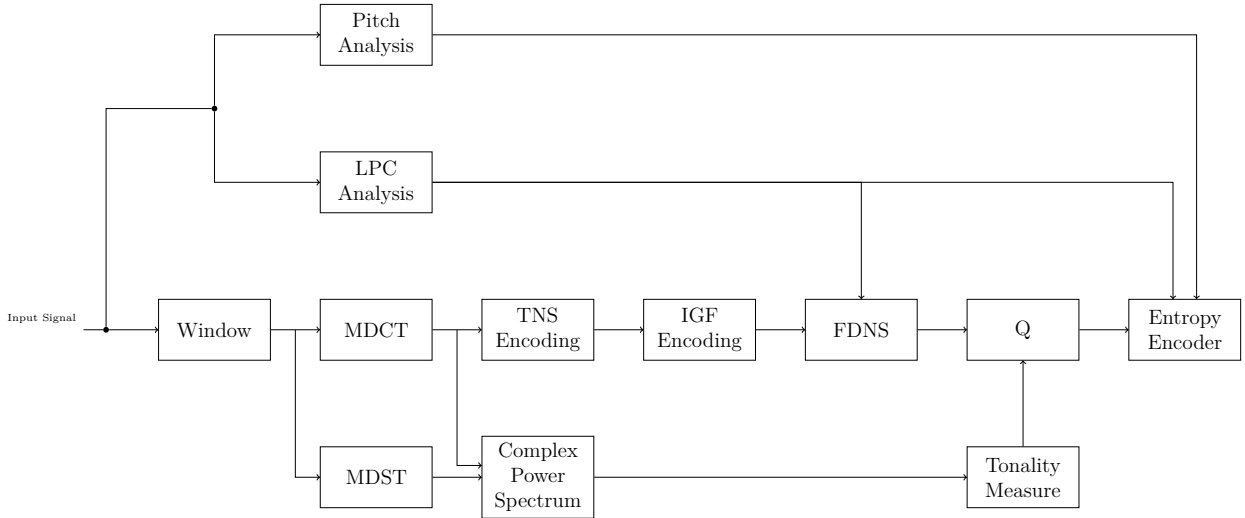


Figure 11: Structure of the MDCT-based TCX encoder in EVS

the probability models for the entropy coding, as explained in section 4.6. The LPC coefficients are also used to estimate the quantisation noise shaping model applied in the frequency domain. At higher bitrates, the MDCT-based TCX uses a context based arithmetic encoder.

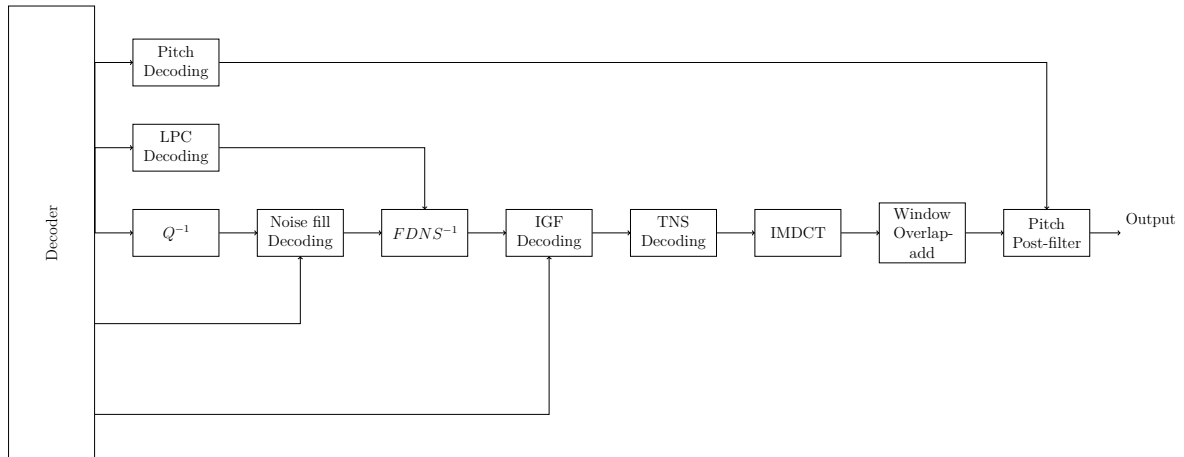


Figure 12: Structure of the MDCT-based TCX decoder in EVS

The decoder structure, as presented in figure 12, follows the same pattern as the inverse TCX, where all the processing applied to the input signal in the encoder must be inverted and applied in the opposite order, in order to reconstruct the original signal.

3.5 Transformations

LPC based codecs parametrize the spectral envelope and the fundamental frequency information and subtract it from the signal leaving an excitation signal, which has a

behaviour similar to noise. However, the effect of the fundamental frequency and the spectral envelope is not completely removed so the excitation signal will still have some degree of correlation between its samples.

This correlation between samples in a frame is a problem for many encoding algorithms. These systems consider that the samples in the target vector of the innovative codebook are decorrelated, which makes their performance suboptimal in terms of bit consumption. ACELP codecs nowadays use more complex encoding algorithms that try to take advantage of this correlation in order to perform a better quantisation of the target vector and search for the corresponding codebook, but this translates into a higher computational complexity.

The objective of this thesis is to investigate the use of other algorithms that transform the innovative target into a different domain in which the samples are decorrelated. This way, it will be possible to use a simpler encoder once the samples are decorrelated. In the decoding part it will only be necessary to invert the transformation to get back the encoded residual.

Four different transformations are considered in this thesis depending on their properties. The first transformation is the Vandermonde decomposition on Toeplitz matrices as it is explained in [5, 22]. DFT and DCT are two other transformations and the last one will be the Karhunen-Loeve transformation applied as the Singular Value Decomposition.

On the one hand, SVD provides a complete decorrelation of the signal because, by definition, it can transform the autocorrelation matrix of the excitation into a diagonal matrix. However, singular values do not have any physical representation of the signal.

On the other hand, DFT is not supposed to decorrelate the signal in the way SVD does, but it directly represents the values of the frequency bands of the signal.

Vandermonde decomposition joins these two characteristics. It establishes a direct relation between the time domain signal and its frequency representation, but it warps the frequency bands in order to minimize the correlation between them.

3.5.1 Discrete Fourier Transform

One of the most common and basic transformations used in signal processing is the Fourier transform. It is based on the idea that every signal can be represented with an infinite number of harmonics and allows us to decompose a signal into its frequency components.

In the world of digital signal processing, this transformation has a discrete counterpart, the DFT. It represents a finite-length discrete signal in the frequency domain using a finite number of uniformly distributed frequency bands. This transformation is represented by an N-by-M matrix which is multiplied with the signal. This matrix is

$$W = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \dots & \omega^{(M-1)} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{2(M-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \dots & \omega^{3(M-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{(N-1)} & \omega^{2(N-1)} & \omega^{3(N-1)} & \dots & \omega^{(N-1)(M-1)} \end{bmatrix}, \quad (7)$$

where $\omega = e^{-\frac{2\pi i}{N}}$ is a primitive Nth root of unity in which $i = \sqrt{-1}$. The dimensions N-by-M of the matrix represent the number of frequency bands in the transformation and the length of the signal, respectively. According to this definition, every row of the matrix can be interpreted as a rotation around the unit circle at a frequency $f = \frac{n \cdot f_s}{2N}$, where f_s is the sampling frequency of the signal and n is the row number in the matrix.

To apply the transformation it is only necessary to multiply the matrix by the signal in the form of a column vector:

$$\begin{pmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ \vdots \\ X(N-1) \end{pmatrix} = W \cdot \begin{pmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ \vdots \\ x(N-1) \end{pmatrix} \quad (8)$$

However, this transformation can become a tedious operation if the length of the signal is too large since after all, it is a multiplication by a matrix so its complexity would be $O(n^2)$.

What makes the Fourier Transform so famous as a DSP tool is its computationally efficient variant. The Fast Fourier Transform(FFT), is an algorithm that takes advantage of the redundancy in the DFT matrix multiplications to reduce its complexity to $O(n \cdot \log(n))$. FFT avoids the matrix multiplication by factorizing the matrix into a product of sparse factors which consist largely of zeros.

The most famous algorithm used to perform the FFT is the Cooley-Tukey and uses the strategy of divide and conquer, which divides the transformation into two $\frac{N}{2}$ point DFT transformations.

As figure 13 shows, to divide the DFT into two $\frac{N}{2}$ transforms, it is necessary to reorder the input samples into even- and odd-position samples and apply the transformation to each group. These samples are then scaled and summed as it is shown in the picture, which is referred to as a butterfly structure.

This secondary DFTs can also be divided into smaller transformations, which results in multiple 2-point DFT transformations. Considering the multiple divisions by 2, the maximum efficiency will be reached if the length of the transformation is a power of 2. That is why it has become a common practice to adapt the length of the FFTs to this rule.

Every time a transformation is simplified, it is necessary to consider the reordering of the elements and their scaling. Considering figure 13 example, the final 8-points

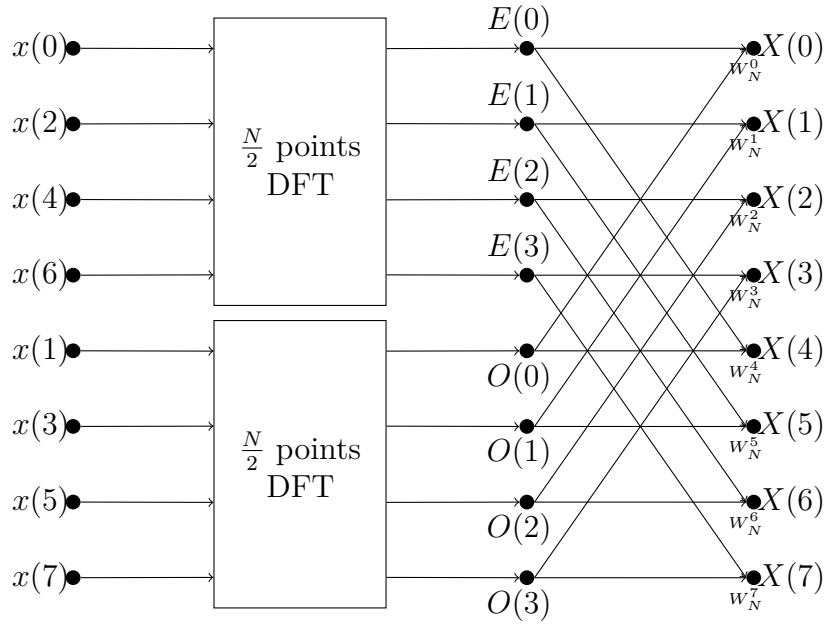


Figure 13: Block diagram of an N -point DFT divided into 2 $\frac{N}{2}$ -point DFTs.

FFT can be seen in figure 14:

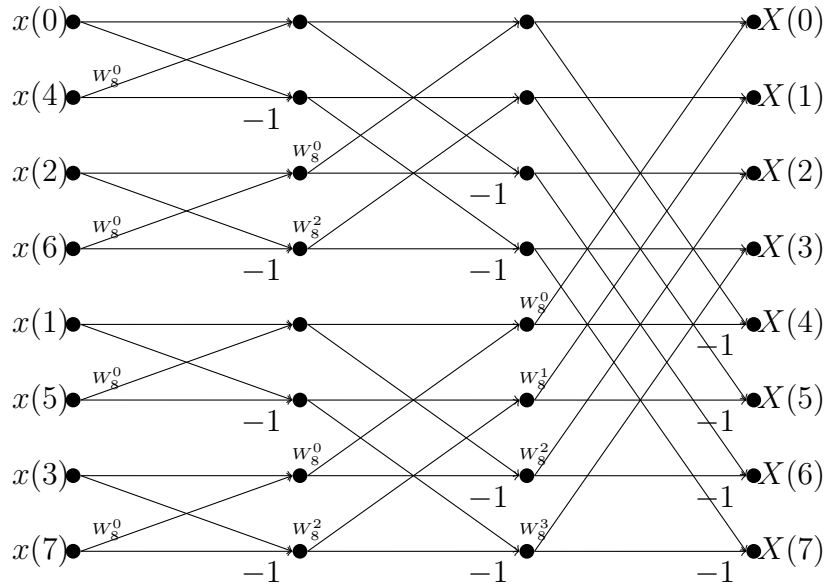


Figure 14: Block diagram of an 8-point FFT.

The scaling factor W_8^n represents the corresponding element in the DFT matrix $W_N^n = e^{-j\frac{2\pi n}{N}}$.

3.5.2 Discrete Cosine Transform

The Discrete Cosine Transform is another transformation widely used in signal processing. It is based in the same principle as the Discrete Fourier Transform, which is representing a signal by the combination of multiple tonal components at different frequencies. However, these tonal components are represented with real cosine functions oscillating at different frequencies instead of complex exponential functions.

One of the main differences of DCT when compared to the DFT is that the frequency representation of the signal has only real values, which represent the amplitude of the cosine components at different frequencies.

There exist various versions of the DCT, which take advantage of different properties of the signal. The most famous of them are DCT-II/III and DCT-IV.

$$X(k) = \sum_{n=0}^{N-1} x(n) \cos \left[\frac{\pi \left(n + \frac{1}{2} \right) k}{N} \right] \quad \text{For } k = 0, \dots, N - 1; \quad (9)$$

$$X(k) = \frac{x(0)}{2} + \sum_{n=1}^{N-1} x(n) \cos \left[\frac{\pi \left(k + \frac{1}{2} \right) n}{N} \right] \quad \text{For } k = 0, \dots, N - 1; \quad (10)$$

Equations 9 and 10 define the DCT-II and DCT-III respectively. These two transforms are the most common version of DCT which will be used in this thesis. Both transformations are always represented as a pair, as version III computes the inverse transformation of version II. Some scaling factors have to be considered to make one version the exact inverse of the other; however, this will be elaborated upon further in section 4.

$$X(k) = \sum_{n=0}^{N-1} x(n) \cos \left[\frac{\pi}{N} \left(n + \frac{1}{2} \right) \left(k + \frac{1}{2} \right) \right] \quad \text{For } k = 0, \dots, N - 1; \quad (11)$$

Equation 11 shows how the DCT-IV is calculated. This version of the DCT is special, as its transformation matrix is orthogonal, which makes the transformation its own inverse. This version of DCT is the basis for the Modified DCT, which is the transformation used in the EVS codec for frequency domain coding.

Although DFT is a very powerful tool for signal visualization and analysis, DCT provides a better performance in compression tasks due to its high degree of spectral compaction. DCT tends to distribute the energy of the signal to a more reduced number of coefficients than the DFT. This means that the most significant part of the signal will be present in a smaller number of coefficients, which reduces the amount of data that needs to be stored.

3.5.3 Singular Value Decomposition

SVD is a technique widely used nowadays in applications that deal with very large amounts of data, such as pattern recognition. This happens because the transforma-

tion represents an expansion of the original data in a coordinate system where its covariance matrix is diagonal.

$$A = U\Lambda V^H \quad (12)$$

The SVD can be expressed as it is shown in equation 12, in which a matrix A with dimensions $N - by - M$ is decomposed into the product of three matrices. Matrix U has dimensions $N - by - N$ and its columns are the left singular vectors; matrix V is $M - by - M$ and its columns are the right singular vectors and Λ is a diagonal matrix with the same dimensions as A , the values of the diagonal are all non-negative real numbers known as singular values.

In some special cases, if A is a square matrix that fulfils certain conditions, this decomposition is identical to the also famous eigenvalue decomposition. This is the case of this thesis, where the matrices on which the decomposition will be applied are autocorrelation matrices. The eigenvalue decomposition can be expressed as:

$$A = U\Lambda U^H \quad (13)$$

As equation 13 shows, the result of the decomposition is a matrix U , whose columns are orthogonal vectors with norm equal to 1 referred as eigenvectors, and Λ is a diagonal matrix containing non-negative real values (eigenvalues).

As it was stated before, the goal of this decomposition is to transform a set of data to another coordinate system where its covariance matrix is diagonal. This means that a signal is transformed in such a way that its values are decorrelated. The eigenvectors in U represent the orthonormal basis of the new coordinate system in which the signal is decorrelated. To transform the signal to the new domain it is only necessary to multiply it by the eigenvector matrix.

The inverse of the transformation is performed by multiplying the transformed vector by the inverse transformation matrix. An interesting feature of this decomposition is that, considering the new basis orthonormal, the eigenvectors in the transformation matrix will be linearly independent. The inverse matrix of an orthogonal matrix is its own transpose, which makes it easy to find the matrix that inverts the transformation.

This decomposition provides a transformation where the values of the signal will have the maximum decorrelation. However, unlike DFT or DCT, the new coordinate system of this transformation is quite abstract and does not have a direct physical interpretation.

The last transformation studied in this thesis tries to combine these two features so it is possible to have a physical representation of the signal and, at the same time, obtain a vector with decorrelated values.

3.5.4 Vandermonde Decomposition

According to [23], it is possible to define a Vandermonde matrix V as

$$V^H = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \alpha_1^3 & \cdots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \alpha_2^3 & \cdots & \alpha_2^{n-1} \\ 1 & \alpha_3 & \alpha_3^2 & \alpha_3^3 & \cdots & \alpha_3^{n-1} \\ 1 & \alpha_4 & \alpha_4^2 & \alpha_4^3 & \cdots & \alpha_4^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \alpha_n^3 & \cdots & \alpha_n^{n-1} \end{bmatrix} \quad (14)$$

where

$$\alpha_i \neq \alpha_j \quad \forall \quad i \neq j. \quad (15)$$

Vandermonde matrices are useful for polynomial interpolation, because they are equivalent to evaluating the values of a polynomial $a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}$ at the different points α_i . Therefore, solving the system of linear equations $Vu = y$ gives the values of the coefficients u_i , which belong to the polynomial.

$$P(x) = \sum_{j=0}^{n-1} u_j x^j \quad (16)$$

However, [24, 25] propose a different application for Vandermonde matrices, in which a Hankel matrix can be decomposed into a product of a diagonal and a Vandermonde matrix.

$$H = \begin{bmatrix} s_1 & s_2 & s_3 & s_4 & s_5 & s_6 \\ s_2 & s_3 & s_4 & s_5 & s_6 & s_7 \\ s_3 & s_4 & s_5 & s_6 & s_7 & s_8 \\ s_4 & s_5 & s_6 & s_7 & s_8 & s_9 \\ s_5 & s_6 & s_7 & s_8 & s_9 & s_{10} \\ s_6 & s_7 & s_8 & s_9 & s_{10} & s_{11} \end{bmatrix} \quad (17)$$

$$H = V^H D V \quad (18)$$

Equation 17 represents a Hankel matrix, which can be defined as a square matrix where every ascending skew-diagonal from left to right has a constant value. The Vandermonde decomposition is then defined in equation 18, where V is a Vandermonde matrix as described above, and D represents a diagonal matrix.

$$T = \begin{bmatrix} s_6 & s_7 & s_8 & s_9 & s_{10} & s_{11} \\ s_5 & s_6 & s_7 & s_8 & s_9 & s_{10} \\ s_4 & s_5 & s_6 & s_7 & s_8 & s_9 \\ s_3 & s_4 & s_5 & s_6 & s_7 & s_8 \\ s_2 & s_3 & s_4 & s_5 & s_6 & s_7 \\ s_1 & s_2 & s_3 & s_4 & s_5 & s_6 \end{bmatrix} \quad (19)$$

Analogously, [22] proves that this transformation can be extended to Toeplitz matrices. A Toeplitz matrix can be defined as in equation 19, and it can be seen as a Hankel matrix flipped upside-down. Therefore, the decomposition of Toeplitz matrices can be performed in a similar way as with the Hankel matrices:

$$T = V_x^H D V_x \quad (20)$$

Where V_x is defined by scalars $(v_k^*)^{-1}$. However, as it is stated before, the objective of this transformation is to decorrelate the values of the innovative target vector. This can be translated as turning autocorrelation matrix into a diagonal matrix. These matrices represent a special type of Toeplitz matrices because they are hermitian. Because of this, the Vandermonde decomposition of an autocorrelation matrix can be expressed as:

$$R_{xx} = V^H D V \quad (21)$$

The Vandermonde matrix described in 14 used as the transformation matrix such that

$$\begin{pmatrix} X(\Omega_0) \\ X(\Omega_1) \\ X(\Omega_2) \\ X(\Omega_3) \\ \vdots \\ X(\Omega_{N-1}) \end{pmatrix} = V^H \cdot \begin{pmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ \vdots \\ x(N-1) \end{pmatrix} \quad (22)$$

Upon closer inspection to the structure of a Vandermonde matrix in equation 14 and the DFT transformation matrix 7, it is possible to identify some similarities. Namely, that the DFT is a special case of Vandermonde matrix where the coefficients $\alpha_i = e^{-\frac{2\pi i}{N}}$. Therefore, it is possible to find a Vandermonde matrix applicable to the decomposition and whose coefficient values are located on the unit circle representing frequency bands as explained in section 3.5.1.

This Vandermonde matrix provides a warped spectral representation of the signal. Considering that the objective of the Vandermonde decomposition is to transform the autocorrelation matrix into a diagonal matrix, the spectral warping represents the frequency components where the signal is most uncorrelated. Figure 15 shows the frequency components of a Vandermonde transformation compared to the DFT frequency bands. The figure shows the frequency band in terms of phase values around the unit circle. To determine the frequency value that is necessary, the following formula is used

$$f_n = \angle \omega_n \frac{f_s}{2\pi} \quad (23)$$

where f_s is the sampling frequency of the signal and $\angle \omega_n$ represents the phase of the coefficient α_n in the matrix defined in 14. In DFT, $\angle \omega_n = n \frac{2\pi}{N}$ is uniformly distributed.

As the previous formula says, the relation between the phase of the points on the unit circle and the frequency of the corresponding band are proportional. Looking at figure 15, the warping effect is noticeable, which leaves some points closer to their neighbours while some areas have a lower density. This means that the frequency resolution will be higher in some areas of the spectrum and lower in others.

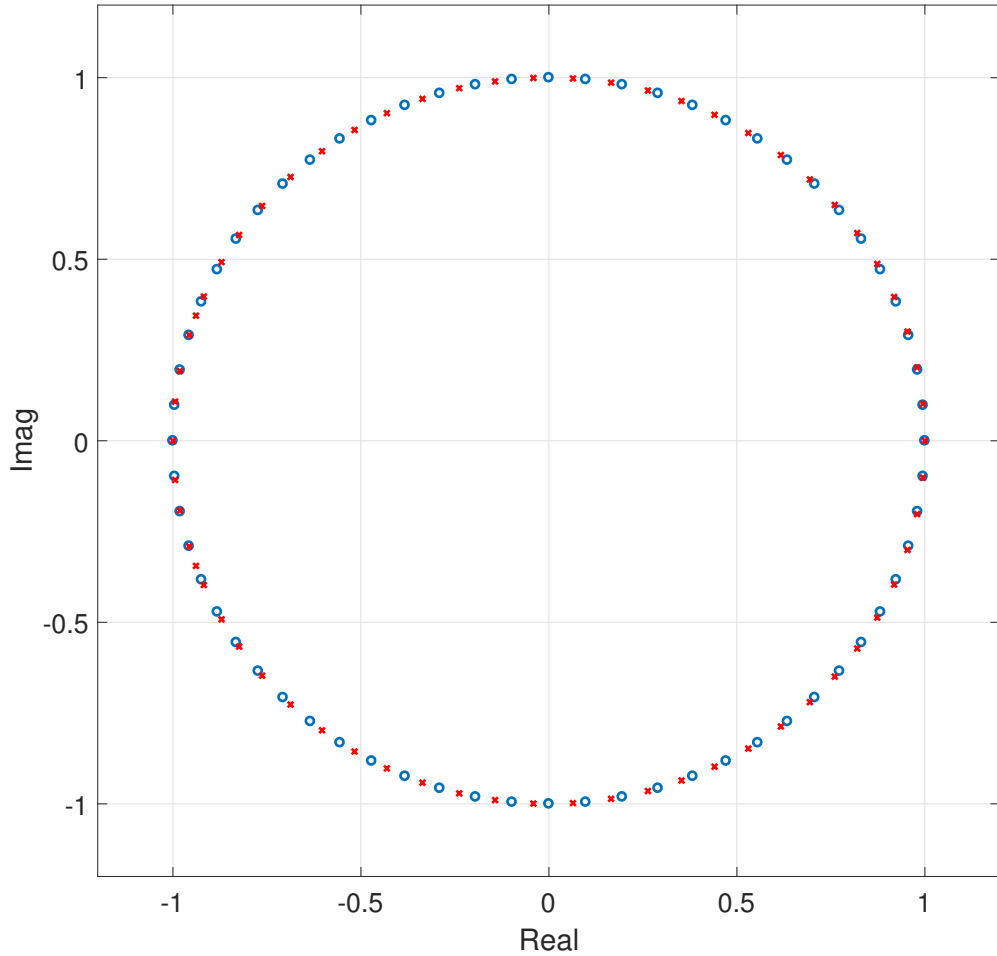


Figure 15: Frequency bands at which the DFT takes values compared to the values of the Vandermonde decomposition in the same subframe. Blue circles: DFT bands. Red crosses: Vandermonde decomposition.

As a summary, both transformations have a direct representation in the frequency domain. However, as [26] states, the basis function of the DFT can be represented as $f(n) = e^{-i\omega(n)}$, where $\omega(n) = \frac{2\pi k}{N}$ for $0 \leq k \leq N - 1$ for a signal of length N , while the Vandermonde can be represented by $\alpha_{\Omega}(n) = e^{-i\Omega_k n}$ where $\Omega_k \in [-\pi, \pi]$ and whose values are distributed at the N frequencies that coincide with the Vandermonde decomposition conditions.

Considering the decorrelation features of the Vandermonde decomposition, its similarity with the Eigenvalue decomposition can be identified. This can be seen in the structure of the decompositions in equations 13 and 21, which both transform a square matrix into a product of a diagonal matrix and a square matrix from both sides.

However, apart from the structure of the decomposition, they have little in

common. Eigenvalue decomposition is a tool used in dimensionality reduction of data sets that may contain any kind of data, this means that the transformation can be applied to any kind of matrix, obtaining always the aforementioned decomposition. Vandermonde, however, must be applied to a Toeplitz hermitian matrix.

As explained before, Vandermonde decompositions have a direct relationship with the spectral representation of the signal where its coefficients are warped frequency components adapted to the decorrelation task. Whereas, Eigenvalue decomposition transforms the signal into a domain where its samples are decorrelated; however, the Eigenvalues does not have a direct physical representation.

The final difference between Vandermonde and the Eigenvalue decomposition is the transformation matrix. While the Eigenvector matrix is orthogonal, which means that its columns (Eigenvectors) are linearly independent and their norm is equal to 1, the Vandermonde matrix is composed by linearly independent columns which, notwithstanding, do not form an orthonormal base. The orthogonality of the Eigenvalue matrix means that the inverse of the matrix is its complex-conjugate transpose, making it easier to calculate than a normal matrix. This does not happen in the Vandermonde decomposition.

4 Implementation

This section deals with the implementation of the four transformations and explains the mechanisms used in detail. The different transformations have been integrated in the EVS codec commercial framework[27], using C as a programming language and Visual Studio 2015 as the development environment. Implementing some of the transformations required some external libraries, such as the Linear Algebra Package (LAPACK)[28], to be linked to the project. The working mode of the codec operates with a sample rate of 16 kHz for the input and output signals, and the encoder/decoder bitrates are 16.4, 24.4 and 32 kbps.

The algorithms have been implemented in a modified version of the EVS codec [29]. As stated in section 2.1, EVS implements two modes for ACELP encoding and three different MDCT-based encoders which are selected depending on the encoding bitrate, amongst other parameters. The version used in this thesis forces the use of ACELP mode 2 at all bitrates and MDCT-based TCX coding using symmetric windows for the frequency domain encoding.

As expressed before, the goal of this work is to transform the excitation of the LPC filtering, after removing the contribution of the fundamental frequency, into a different domain where the samples are decorrelated in order to obtain a more efficient encoder. According to this, the integration of the transformation algorithms will be carried out within the ACELP structure. This algorithm is implemented in the EVS codec and some elements have to be modified. The main structure of the ACELP encoder will be explained in order to provide a better understanding of the changes carried out.

4.1 Integration in EVS

The objective of the thesis is to study different algorithms that may provide some improvements over the current ACELP encoding system, while keeping the LPC structure. To do this, the algorithms were integrated in the EVS codec using the ACELP mode so it is necessary to understand in detail the structure of the ACELP encoder.

Working in ACELP mode for 16 kHz sampling frequency, EVS windows the signal to calculate the LPC coefficients using an asymmetric window with 25ms duration. The processed frames have a length of 20 ms. These frames are then divided into subframes of 64 samples length, which means that every frame will contain 5 subframes of 4 ms. The encoding task performs subframe-by-subframe processing in a closed loop where the effect of previous subframes is considered. The work of this thesis is focused inside the subframe loop so the length of the signals will be 64 samples. From now on, the subframes inside the closed loop will be referred to as frames considering that the work is carried out only inside this loop.

Figure 16 represents the block diagram of the ACELP encoder subframe loop present in EVS. The red dotted boxes represent the places where some modifications have been performed in order to integrate the transformations.

1. The speech signal is analysed and the 16 LPC coefficients $A(z)$ are extracted

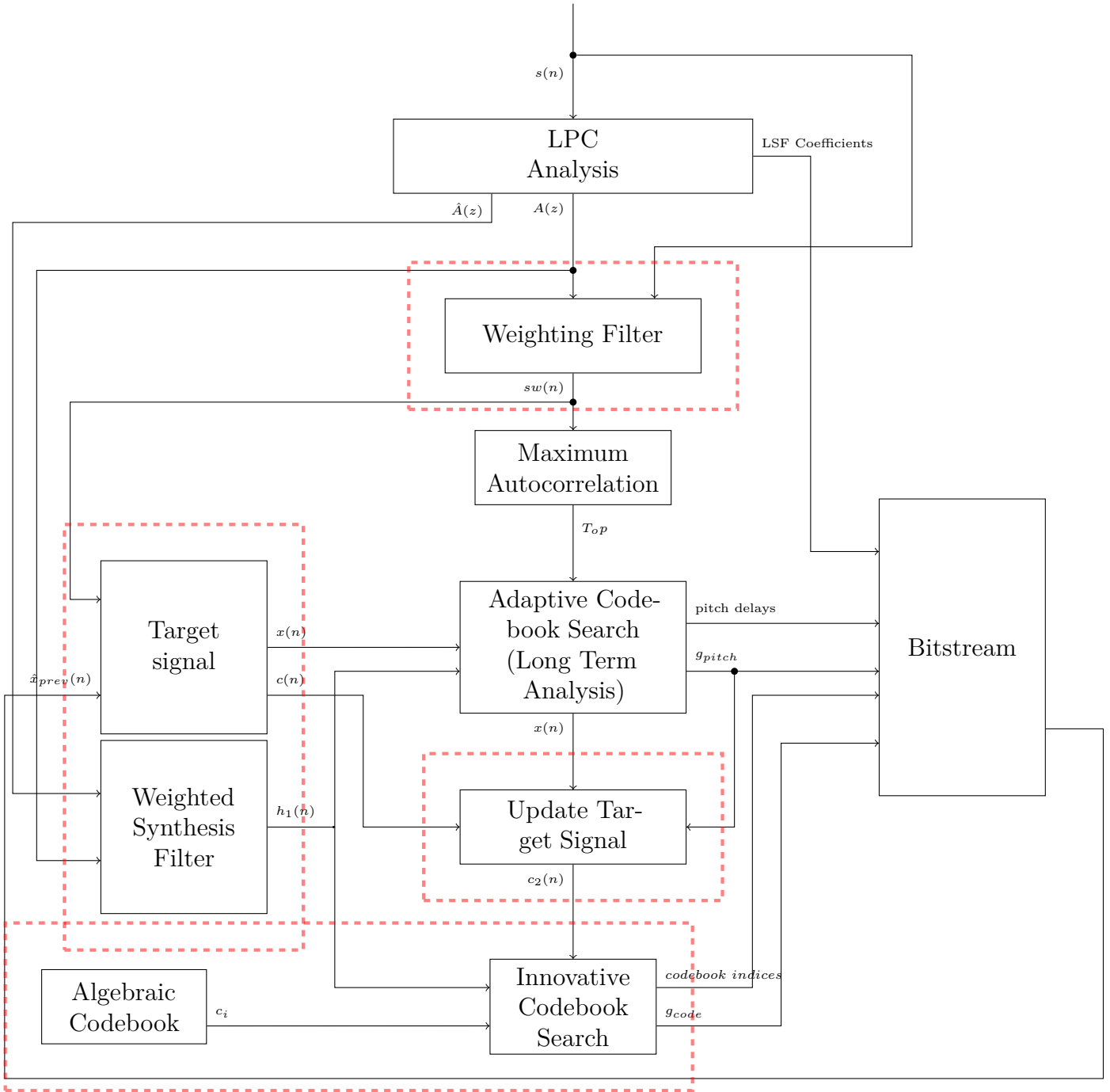


Figure 16: ACELP block diagram in detail.

and quantised $\hat{A}(z)$. The LSF quantised coefficients, which represent the LPC filter for its transmission, are sent into the bitstream.

2. The signal is filtered through a perceptual weighting filter derived from the LPC coefficients.
3. Using the perceptually weighted signal and the previous excitation, two target

vectors are obtained. The target vector for the adaptive codebook search ($x(n)$), and the innovative codebook search ($c(n)$), which are calculated as explained in section 4.2.

4. The LPC coefficients are interpolated, obtaining then the impulse response of the weighted synthesis filter ($h_1(n)$).
5. $h_1(n)$ and the target for the pitch search ($x(n)$) are used in a LTP analysis to estimate the pitch delay (Adaptive codebook) and its corresponding gain. The gain and the pitch delay are introduced into the bitstream. The maximum in the autocorrelation of the weighted signal defines the open loop delay (T_{op}), which limits the search range of the optimal adaptive codebook.
6. In order to remove the influence of the pitch from the innovative target ($c(n)$), the pitch target, scaled by its corresponding gain, is subtracted from it. $c_2(n)$ will be known as the updated innovative vector.
7. The search of the innovative codebook (\hat{c}) is performed using the innovative target and the information from the synthesis filter. The codebook and its gain are encoded in the bitstream and sent to the receiver.
8. Finally, the excitation is reconstructed from the adaptive and innovative codebooks, multiplied by their corresponding gains and stored to be used in the calculation of the target vector for the next subframe.

The goal of the thesis is to try different transformations that decorrelate the updated innovative target vector in order to simplify the innovative codebook search. This means that the main changes will be carried out in step 7, where the encoding algorithm is replaced by its frequency domain version.

In order to apply the envelope based arithmetic encoder, the spectral envelope used in the encoder and decoder has to be the same, which means that the quantised LPC coefficients have to be used to calculate the impulse response of the synthesis filter and the target vectors.

After an initial implementation of the transformation algorithms, a tilt filter will be used on the spectral envelope in order to adapt the distribution of the bits in the encoder, this will make the quantisation noise less perceivable. As this tilt filter is implemented as a de-emphasis filter, the pre-emphasis and de-emphasis effects will be explained to understand the effect of this filter.

4.2 Target Vector

As stated before, the codebook search is performed in the perceptually weighted domain, and that is why it is necessary to represent the excitation for every codebook in this domain. The representations of the excitations in the perceptually weighted domain are the target vectors

As it is shown in figure 16, two target vectors are generated for the codebook search. One of them is used in the pitch analysis to generate the adaptive codebook and the other one is used in the innovative codebook search.

The target signal for the adaptive codebook $x(n)$ is computed by subtracting the zero input response of the weighted synthesis filter

$$W(z)H(z) = \frac{A\left(\frac{z}{\gamma_1}\right)}{\hat{A}(z)(1 - \alpha z^{-1})} \quad (24)$$

where $\alpha = 0.68$ is a de-emphasis factor used to compensate the pre-emphasis applied to the speech signal in the pre-processing explained in section 2.1. The coefficients of $A\left(\frac{z}{\gamma_1}\right)$ can be defined as $a_\gamma(i) = \gamma_1^i a(i)$ for $i = 0, \dots, N - 1$.

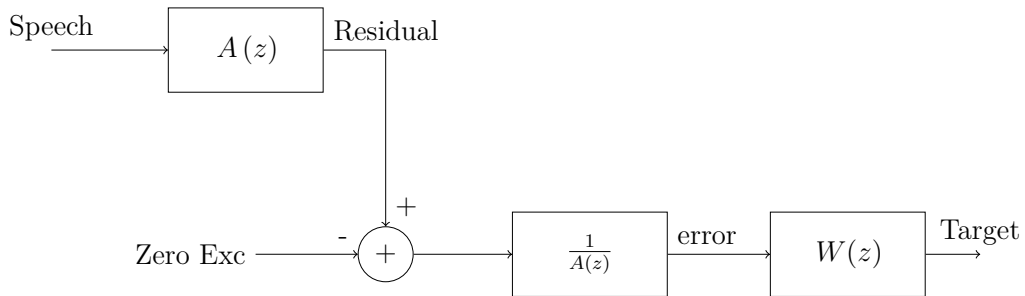


Figure 17: Block diagram of the target vector computation.

Figure 17 shows the block diagram of the target vector computation [27, 29], where the speech signal is filtered through the LPC coefficients, the zero input excitation is subtracted from the resulting residual and the signal is then reconstructed and weighted, obtaining the target vector.

The target vector for the innovative codebook $c(n)$ is the adaptive target vector represented in the residual domain. This means that the target vector is filtered through the linear prediction filter $A\left(\frac{z}{\gamma_1}\right)$ in order to eliminate the effect of the spectral envelope.

To calculate the impulse response of the weighted synthesis filter, the coefficients of $A\left(\frac{z}{\gamma_1}\right)$ are interpolated in order to obtain an array with the same length as the subframe. Being an IIR filter, it is not possible to represent all the coefficients of its impulse response, so only the 64 samples available are calculated and the rest is assumed to be zero. This can be achieved by filtering the LPC coefficients through the two filters $\frac{1}{\hat{A}(z)}$ and the de-emphasis filter as it is shown in equation 24.

The encoder used in this thesis uses the response of the weighted synthesis filter to estimate the probability distribution of the samples in the vector in order to allocate the available bits that will encode the signal. The decoder, as a result, needs to know this probability distribution to reconstruct the encoded signal. However, the decoder possesses only the quantized version of the LPC coefficients.

This is why, in order to have an identical synthesis filter in the encoder and decoder, it is necessary to calculate the synthesis filter using the quantised version of the LPC coefficients. The coefficients $\hat{A}\left(\frac{z}{\gamma_1}\right)$ are obtained by applying the same operation used to calculate the unquantised coefficients.

The weighted synthesis filter used in this thesis work looks like

$$W(z)H(z) = \frac{\hat{A}(\frac{z}{\gamma_1})}{\hat{A}(z)(1 - \alpha z^{-1})} \quad (25)$$

so it can be perfectly reconstructed using the elements available in the decoder with the quantized LPC coefficients extracted from the bitstream. This affects the estimation of the innovative target signal, which is calculated using the quantized synthesis filter.

4.3 Target Update

As it is explained in section 4.2, the calculation of the target vector for the innovative codebook search still contains the influence of the fundamental frequency, as only the spectral envelope was subtracted by the LPC filtering.

Considering that the pitch information is already encoded in the adaptive codebook, it is not necessary to keep that information in the innovative codebook. This is why the target update block uses the adaptive codebook and the pitch gain to subtract the pitch information from the innovative target vector.

$$c_2(n) = c(n) - \gamma_{pitch}\hat{x}_{adap}(n) \quad (26)$$

where $c_2(n)$ represents the updated target vector that will be the input of the innovative codebook search.

4.4 Pre-emphasis, de-emphasis and tilt filter

Pre-emphasis and de-emphasis are two complementary operations used in telecommunications to raise the energy of some frequency bands with respect to others. Pre-emphasis is used to rise the high frequencies and take the lower frequencies down, while de-emphasis has the opposite effect. These two effects are represented as:

$$H_{Preemph}(z) = 1 - \alpha z^{-1} \quad (27)$$

$$H_{Deemph}(z) = \frac{1}{1 - \alpha z^{-1}} \quad (28)$$

The application of these filters in telecommunications has the purpose of increasing the energy of the zones that are more sensitive to noise by pre-emphasising the signal. The signal is then processed according to its final application and, finally, it is de-emphasised in order to compensate the effect of the pre-emphasis. This can be seen in the EVS codec, where the signal is pre-emphasised in the preprocessing block of the encoder and de-emphasised using the same α coefficient at the end of the decoder.

Figure 18 shows the magnitude response of a pre-emphasis filter according to equation 27 with $\alpha = 0.3$. The opposite effect can be seen in figure 19, which represents a de-emphasis filter with the same coefficient.

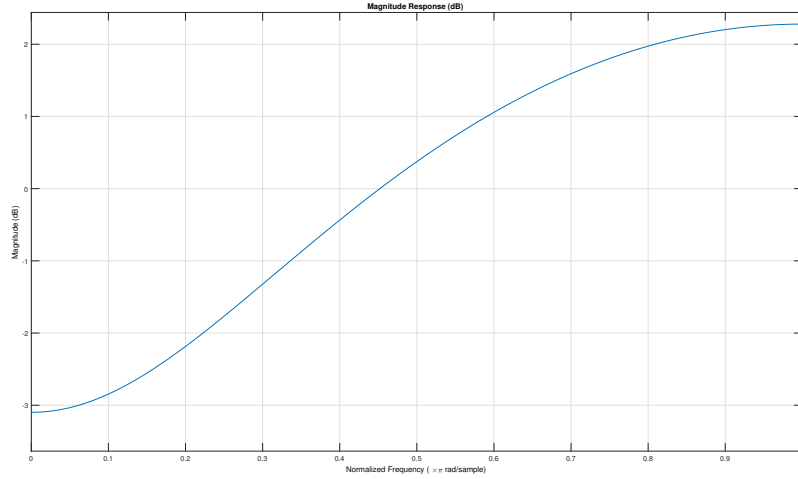


Figure 18: Magnitude response of a Pre-emphasis filter with $\alpha = 0.3$

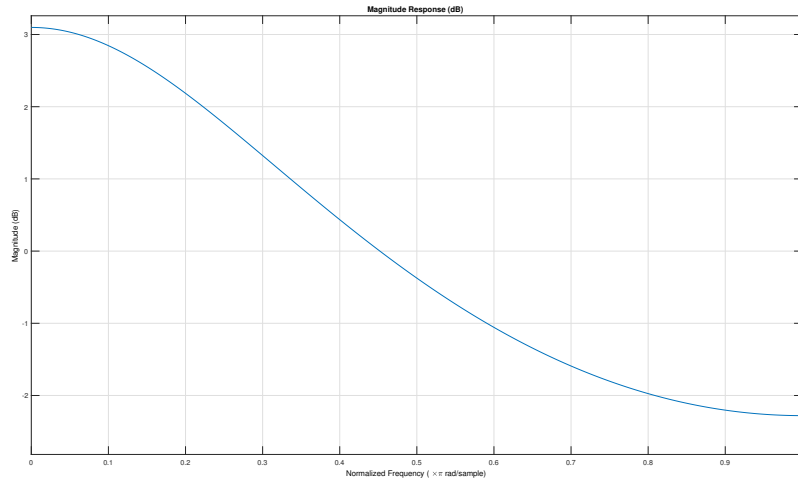


Figure 19: Magnitude response of a De-emphasis filter with $\alpha = 0.3$

The application of pre-emphasis and de-emphasis in this thesis is, however, to modify the response of the synthesis filter in order to optimise the probability model of the envelope based arithmetic encoder, which is explained in detail later.

In an early implementation of the Vandermonde decomposition, the steps followed were as explained in [5], where a pre-emphasis is applied to the synthesis filter before the encoding process. However, this implementation carries some numerical instability when integrated in the EVS codec. This is why, as a first attempt, the impulse response was not modified.

In the right side of figure 20, strong noise components can be distinguished in the low frequencies. Considering that the number of bits assigned to each frequency band depends on the spectral envelope, it was decided to apply a tilt filter to the

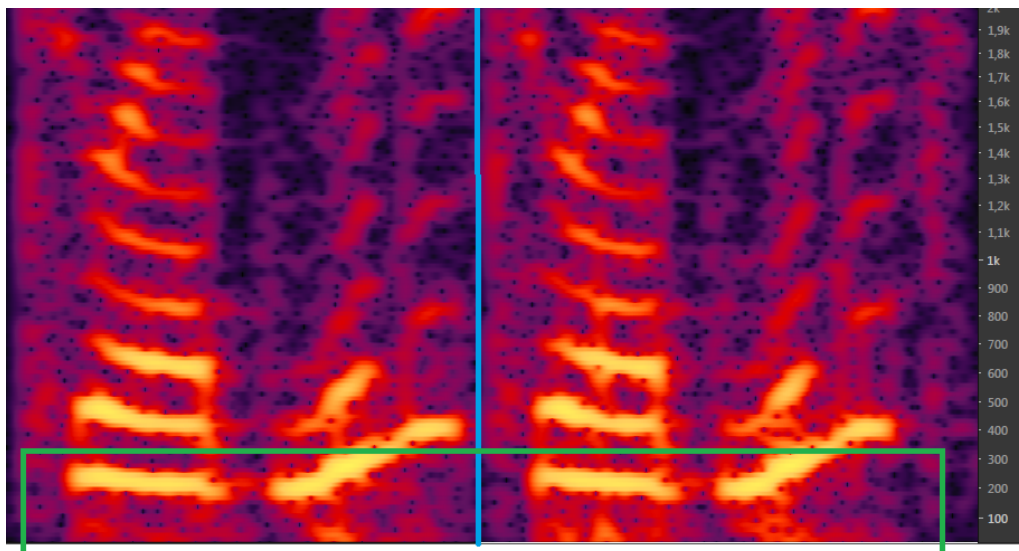


Figure 20: Spectrogram of signal encoded with Vandermonde transformation. Left: Using tilt filter. Right: No tilt filter.

synthesis filter. This filter will be a de-emphasis with a small coefficient in order to slightly reduce the high frequencies of the synthesis filter. This can be translated as giving more importance to the low frequencies in the probability model, assigning more bits to them, which will reduce the quantisation noise.

The left side of figure 20 shows the spectrogram of the signal with the effect of the tilt filter, in this case, the noise has been considerably reduced and it is no longer perceivable.

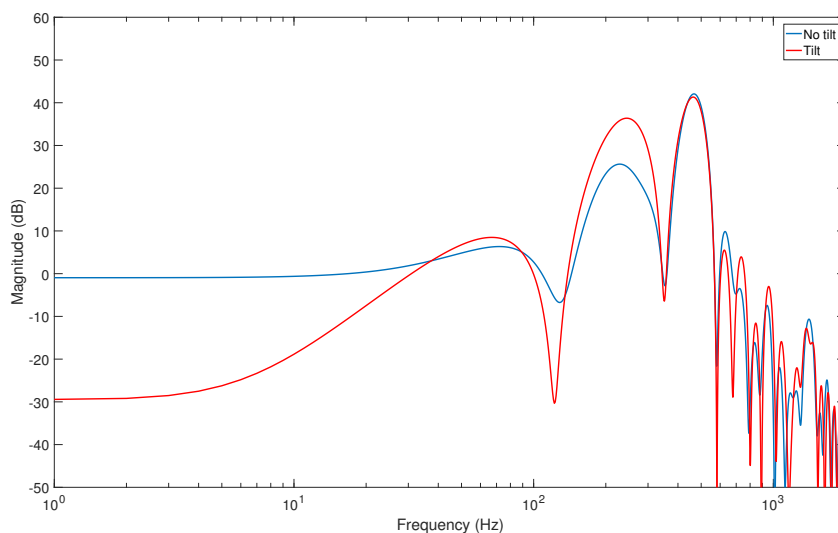


Figure 21: Spectrum of a frame encoded using the tilt filter and without using it

A clearer representation of the effect of the tilt filter can be seen in 21, where

a frame of the signal is represented in the frequency domain. Note that the noise below 50 Hz is as much as 30 dB lower in the version with the tilt filter compared to the version without it.

As stated before, this tilt filter is applied to the impulse response of the weighted synthesis filter before the innovative codebook search. The tilt coefficient is a fixed number so it is possible to reproduce it in the decoder. The resulting block diagram can be seen in figure 22.

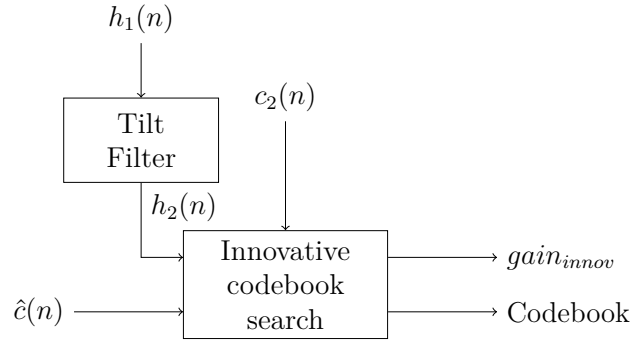


Figure 22: Integration of the tilt filter in the ACELP structure

4.5 Residual transformation

After obtaining the target vectors and processing the impulse response of the synthesis filter appropriately, it is necessary to apply the transformations that were explained in section 3 in order to decorrelate the samples of the target vector, which is the objective of the thesis.

Now the implementation of every transformation will be clarified and comments will be given in terms of the complexity of the algorithm regarding the complexity analysis.

DFT and DCT are signal independent transformations in terms of its transformation matrix. This matrix is constant and does not have to be calculated for every subframe, which allows for fast implementations. Vandermonde and Eigenvalue decomposition, on the other hand, need to calculate the transformation matrix at every subframe; this is why the different implementations of the transformations will be compared in pairs depending on this signal dependency.

4.5.1 DFT

As it is stated before, there is little point in implementing the conventional DFT transform as its fast version, the FFT, provides the same results with much lower complexity. The EVS framework already implements tools to perform the FFT efficiently, so it will not be necessary to implement the largest part of the transformation.

The integration of the DFT in the innovative codebook search is represented in figure 23. Where $c(n)$ is the target vector for the innovative codebook search and $h_1(n)$ is the impulse response of the weighted synthesis filter.

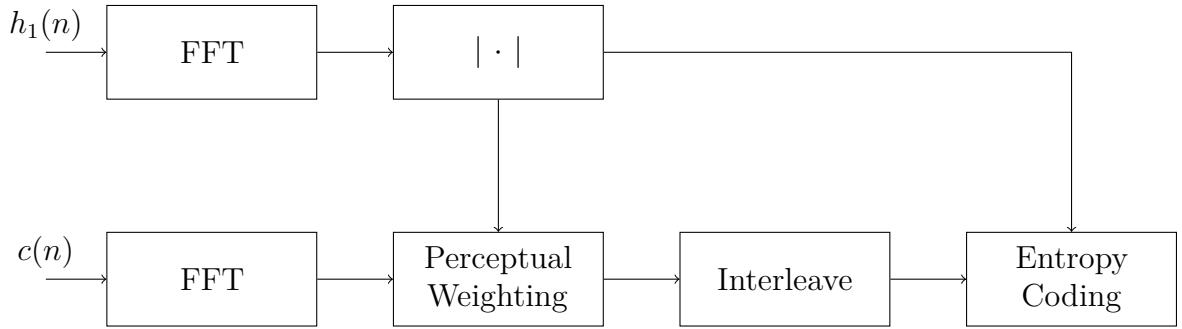


Figure 23: Block diagram of the frequency-domain CELP (FD-CELP) encoder using DFT

Firstly, it is necessary to transform both $c(n)$ and $h_1(n)$ into the frequency domain. The transformations calculate two arrays of complex values and the length of the subframe. The synthesis filter coefficients will be used as a scaling factor for the perceptual weighting and the envelope based arithmetic encoder so its absolute value is calculated.

The target vector is then weighted in the frequency domain using the perceptual model of the synthesis filter: $Y(n) = \sqrt{|H_1(n)|}X(n)$, where $H_1(n)$ represents the transformed response of the synthesis filter. This weighting shapes the signal with the envelope of the synthesis filter.

As stated before, the values in the vector are complex, however, the representation of a real signal in the frequency domain using FFT is a complex-conjugate hermitian function. This allows the removal of half of the values in the vector that can be reconstructed later and, considering that complex values can not be used in the encoding process, the remaining values are interleaved in a 64-length alternating real and complex values of the signal spectrum.

Finally, the target vector is fed to the envelope based encoder, which uses the envelope of the synthesis filter to estimate the probabilities of each symbol in order to distribute the available bits for each sample, as explained later.

The decoder structure is represented in figure 24. Working with the envelope based encoder, the first step is to calculate the envelope of the synthesis filter by performing its DFT and calculating the absolute value of the 64 samples. The synthesis filter has been calculated using the LPC coefficients sent in the bitstream.

The envelope is then used in the decoder to estimate how the bits were distributed in the encoder and to extract the transformed excitation. As the transformed values were sent interleaving their real and imaginary parts, the next step will be the reconstruction of the transformed innovative target vector. Joining the real and imaginary part of every sample will result in a 32-sample vector. As commented before, thanks to the properties of the DFT of a real signal, the other 32 samples are built by mirroring the coefficients' position with respect to the last element and negating the value of the imaginary parts.

It is also necessary to invert the effect of the perceptual weighting applied in the encoding, so the envelope of the synthesis filter will be used as follows: $y(n) = \frac{x(n)}{\sqrt{|H_1(n)|}}$.

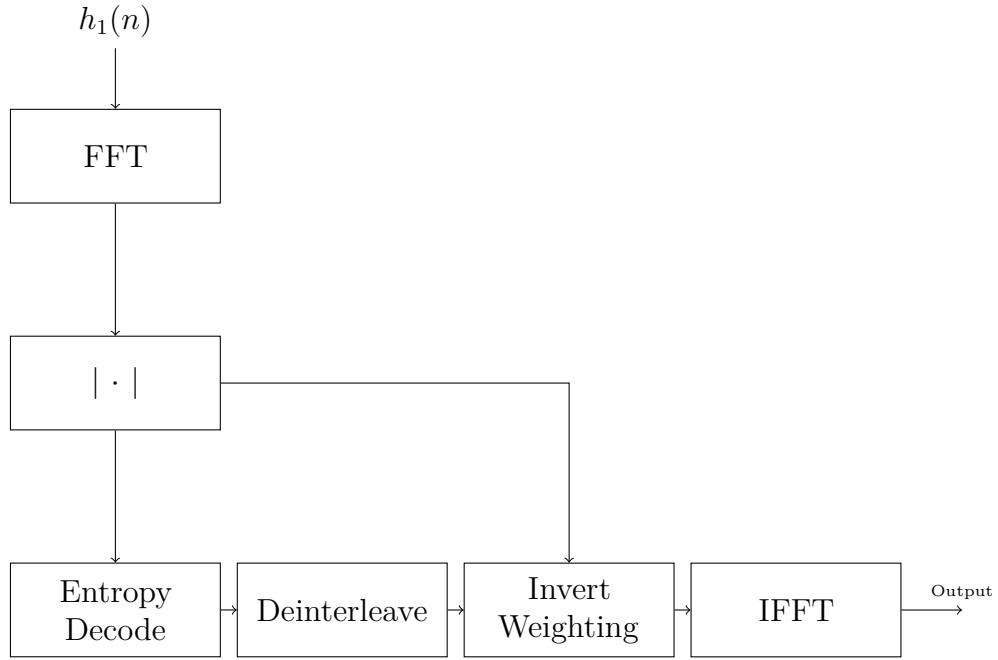


Figure 24: Block diagram of the FD-CELP decoder using DFT

Finally, the IFFT transforms the excitation vector back to the time domain so the output will be the quantised excitation that will be used to reconstruct the signal.

4.5.2 DCT

As explained in the theoretical background, the DCT transformation has different implementations depending on its properties. The most common versions are the DCT II and III, which are mutually inverted, and the DCT-IV, which is its own inverted version.

The difference between using one or the other depends on their boundary conditions. For example, DCT-IV and MDCT perform optimally when the signal is periodic and there are no discontinuities in the boundaries. DCT-II is not so strict with the signal analysed.

EVS already implements a fast method for the DCT-IV, which is used in the MDCT based TCX. This is why the first implementation of the transformation that has been tested uses this method. However, the frame analysed is probably not periodic so it does not assure that the boundary conditions are achieved.

DCT-II has also been implemented in order to compare the effect of the boundary conditions proving that the result is noticeably better when this version is used. However, in this case only the direct application of the transformation has been done so the complexity of the encoding grows.

Equations 9 and 10 represent the equations of the DCT II and III respectively. The first will be implemented for the direct transformation and the second will reverse it. However, in order to invert the transformation correctly, it is necessary to consider some scaling factor in both of the transformations.

In DCT-II, the first term $x(0)$ has to be multiplied by $\frac{1}{\sqrt{2}}$ and then, all the coefficients, including $x(0)$, will be scaled by $\sqrt{\frac{2}{N}}$. The same scaling applies to DCT-III where, instead of dividing $x(0)$ by 2, it has to be divided by $\sqrt{2}$ and then all the coefficients are scaled by $\sqrt{\frac{2}{N}}$. The resulting equations will be:

$$X(k) = \sqrt{\frac{2}{N}} \left\{ \frac{x(0)}{\sqrt{2}} + \sum_{n=1}^{N-1} x(n) \cos \left[\frac{\pi \left(n + \frac{1}{2} \right) k}{N} \right] \right\} \quad \text{For } k = 0, \dots, N-1; \quad (29)$$

$$X(k) = \sqrt{\frac{2}{N}} \left\{ \frac{x(0)}{\sqrt{2}} + \sum_{n=1}^{N-1} x(n) \cos \left[\frac{\pi \left(k + \frac{1}{2} \right) n}{N} \right] \right\} \quad \text{For } k = 0, \dots, N-1; \quad (30)$$

The encoding and decoding scheme is similar to the one described for the DFT, however, the coefficients obtained by the DCT are already real numbers and there is no mirroring in the spectrum, so there will be some differences in the implementation.

It can be observed in figure 25 that the structure is simpler if one is to consider that there is no need to interleave the coefficients of the transformation.

As stated previously, the target vector and the impulse response of the synthesis filter are transformed to the DCT domain. Even though DCT provides a vector of real values, they can still be under 0, so it is necessary to set them all positive to form the spectral envelope.

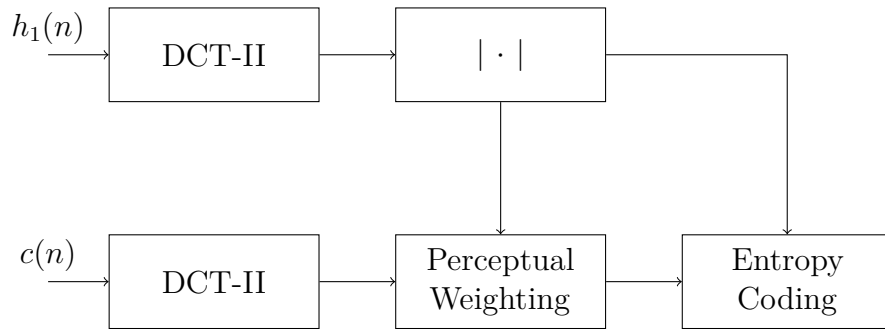


Figure 25: Block diagram of the FD-CELP encoder using DCT

The envelope is used the same way as before to perceptually weight the coefficients of the transformed target, so $Y(n) = \sqrt{|H_1(n)|}X(n)$. The interleaving is no longer necessary and all the coefficients of the signal spectrum are fed into the entropy coder.

Figure 26 shows the structure of the decoder, again noticeably similar to the structure of the DFT decoder except for the deinterleaving process, considering that the values are already real values and represent the whole spectrum of the signal.

The steps are in the same order as with the DFT:

1. Transform the impulse response of the synthesis filter extracted from the bitstream.

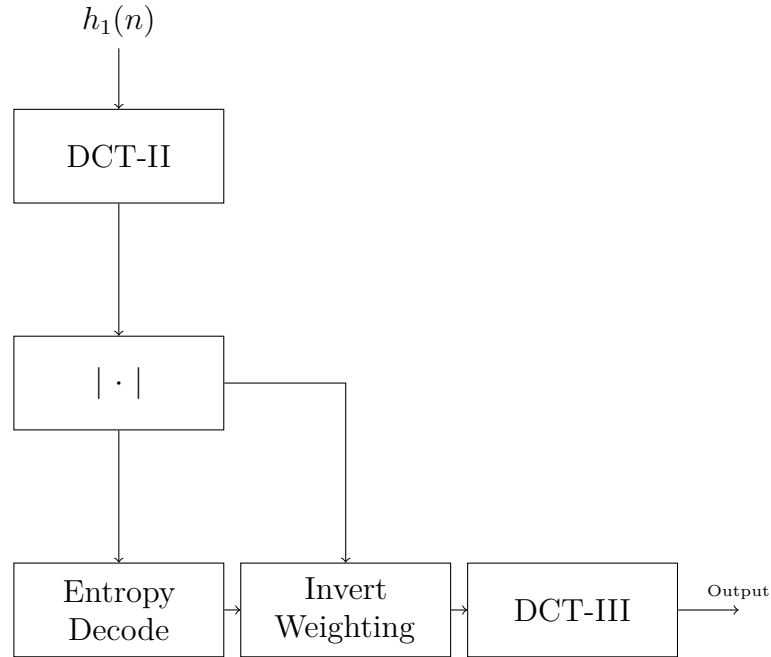


Figure 26: Block diagram of the FD-CELP decoder using DCT

2. Calculate the absolute value of the synthesis filter coefficients and use the spectral envelope to decode the transformed excitation coefficients.
3. Invert the effect of the perceptual weighting from the encoder by dividing the excitation by the square root of the spectral envelope coefficients.
4. Reverse the transformation to obtain the quantised excitation in the time domain.

4.5.3 Vandermonde decomposition

The previous section explains the theory of the Vandermonde decomposition, however, it does not explain how to obtain the corresponding transformation matrices V and D needed in the process.

For the implementation of this transformation, the scheme from [5, 22, 26, 30] is followed. First of all it will be necessary to find the matrix V that satisfies $R_{xx} = V^H D V$.

Given a vector x , which will be the impulse response of the weighted synthesis filter in the encoder implementation, the steps to follow in order to find its V matrix are:

1. Calculate the autocorrelation of the signal x as $R_{xx} = E[xx^H]$.
2. Obtain the linear prediction coefficients of the autocorrelation matrix using the Levinson-Durbin algorithm.
3. Find the roots of $\hat{A}(z)$ adapting them to the unit circle.

The goal of the transformation is to diagonalize the autocorrelation matrix, so the first step will be to obtain the autocorrelation matrix of a signal defined in step 1. The autocorrelation matrix can be obtained by performing a convolution of the vector x with himself. This will result in a vector with length $2N-1$ with the following structure:

$$R_x = \begin{bmatrix} r(N-1) & \dots & r(1) & r(0) & r(1) & \dots & r(N-1) \end{bmatrix} \quad (31)$$

Where $r(n) = \sum_{k=0}^{N-1} x(k)x(n-k)$. Knowing that R_{xx} is a Toeplitz matrix, this vector is enough to represent it.

After calculating the autocorrelation of the vector x , the linear prediction coefficients of R_{xx} are obtained using the Levinson-Durbin algorithm. This results in $\hat{A}(z) = 1 + \sum_{k=1}^{N-1} \hat{\alpha}_k z^{-k}$.

The roots of this polynomial are the coefficients of the Vandermonde matrix v_n . The article in [30] suggests to transform the linear prediction sequence into a tridiagonal matrix and then apply the QR algorithm to find its eigenvalues, which will match with the roots of $\hat{A}(z)$. In this case, the method used involves generating an oversampled Fourier transform of the linear prediction coefficients and looking for the zeros within it.

After making sure that all the zeros are properly distributed around the unit circle and correcting their module to the unit, the last zero is forced to be in the Nyquist frequency according to the spectral representation of the Vandermonde decomposition.

Once the V matrix has been obtained, the diagonal matrix D can be calculated by multiplying the autocorrelation vector with the inverse Vandermonde matrix V^{-1} . An algorithm is proposed in [31] to invert a Vandermonde matrix with a complexity of $O(n^2)$, so the final calculation for the D matrix will be represented as:

$$D = V^{-H} R_{xx} V^{-1} \quad (32)$$

or, considering R_x as the autocorrelation vector:

$$diag(D) = V^{-H} R_x \quad (33)$$

The diagonal of the matrix D represents the scaling factors for every frequency band represented by the transformation, while the V matrix will be used as the transformation matrix as explained in the previous sections. The expression of the transformation will be:

$$y = \sqrt{D} V^H x \quad (34)$$

As mentioned before, the Vandermonde decomposition is a signal dependent transformation. This means that the transformation matrices have to be calculated every subframe in order to adapt the transformation to its decorrelation properties.

Figure 27 shows the structure of the encoder using the Vandermonde transformation. Firstly, the decomposition has to be applied on the synthesis filter to obtain the correct V and D matrices for the spectral information of the corresponding subframe.

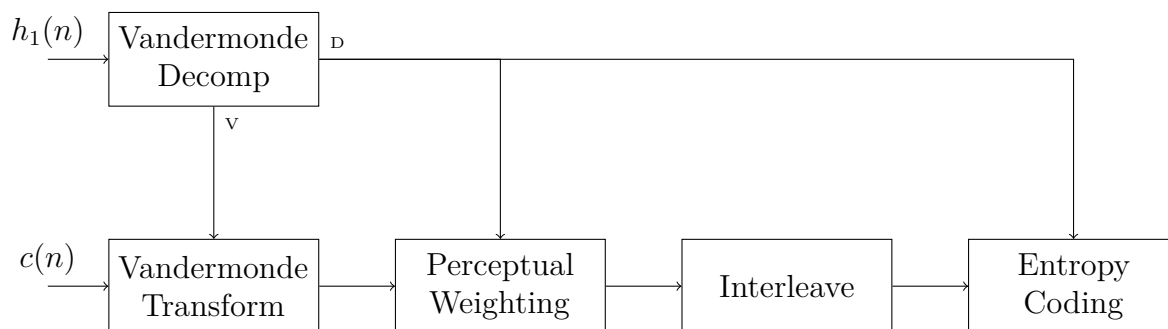


Figure 27: Block diagram of the FD-CELP encoder using Vandermonde decomposition

The V matrix is then used to transform the target vector to the warped frequency domain and the scaling factors from D work as the perceptual model in the weighting task.

The transformation using Vandermonde matrices produces complex values as it is a warped version of the DFT. This means that interleaving will be necessary to store the real and imaginary part of every component. However, like the DFT, the components are mirrored after the Nyquist frequency so only half of the array needs to be stored.

Finally, the envelope based encoder uses the coefficients of D as an envelope to assign the number of bits assigned to every sample.

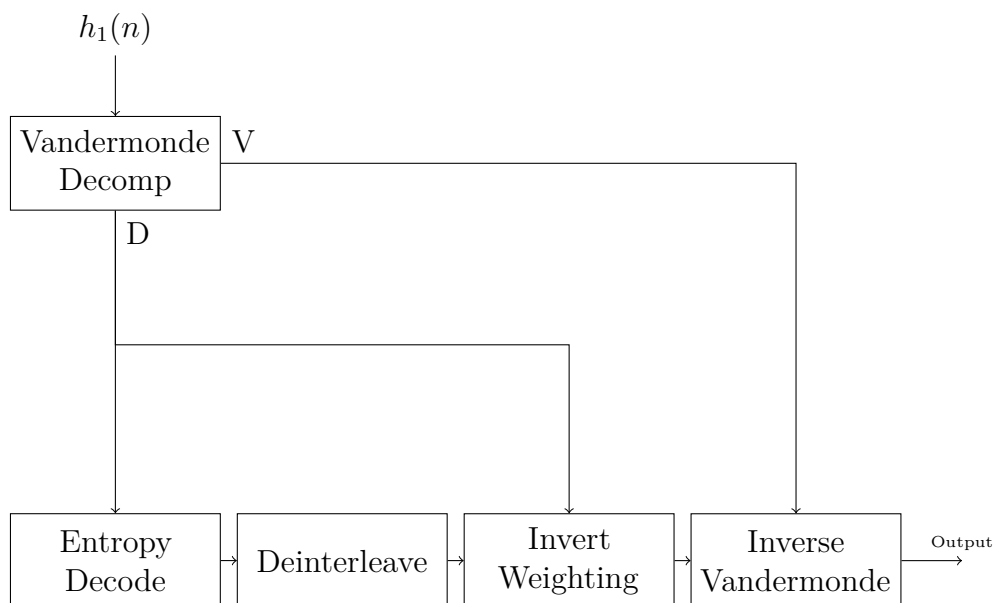


Figure 28: Block diagram of the FD-CELP decoder using Vandermonde decomposition

The decoder part is implemented the same way as the DFT, however, note that it is still necessary to generate the V and D matrices from the decoded LPC coefficients. The inverse transformation is applied, with the inverse scaling, as

$$x = V^{-H} \sqrt{D^{-1}} y \quad (35)$$

4.5.4 Eigenvalue decomposition

Eigenvalue decomposition is the only transformation that does not have a physical interpretation of its coefficients as the previous three algorithms do. However, its purpose is to transform the data to a space where the autocorrelation matrix of the samples is diagonal, which means that the samples are decorrelated.

There are multiple algorithms to calculate this decomposition, however none of them are currently implemented in C and implementing them from scratch would be beyond the scope of this thesis. However, it is possible to utilise external libraries.

The library that will be used in this section is LAPACK. It has multiple algebraic operations implemented in Fortran and includes some functionalities that allow compatibility with C. The guide to compiling and including the LAPACK library in a project can be found in Appendix 1.

The LAPACK library includes functions to implement singular value decomposition on matrices with random dimensions and, more specifically, eigenvalue decomposition when the matrix is square.

The default algorithm implemented in this library to calculate the Eigenvalue decomposition is QR. This algorithm has a complexity of $O(n^3)$ but provides the maximum accuracy in the eigenvectors and eigenvalues.

The QR decomposition is named after the denomination of the two matrices resulting from it. Let A be a matrix whose eigenvalues want to be calculated, this matrix can be decomposed into an orthonormal matrix Q and an upper triangular matrix R such that $A = QR$.

This can be turned into an iterative process where the decomposition can be defined for every k_{th} step as $A_k = Q_k R_k$. The update for the next step can be then formed as

$$A_{k+1} = R_k Q_k = Q_k^{-1} Q_k R_k Q_k = Q_k^{-1} A_k Q_k = Q_k^T A_k Q_k \quad (36)$$

Following the iterations, the algorithms ends up converging into a diagonal solution whose values are the eigenvalues of A_0 and the columns of Q are its eigenvectors.

Considering that an external library implemented in a different programming language is used to perform this transformation, it is necessary to check if the transformation is performed correctly. To do this, the eigenvalue decomposition provided by MatLab has been used, taking it as the reference.

This test is done by running the eigenvalue decomposition on a specific subframe using both methods and comparing the values obtained. The results show that the eigenvectors are calculated accurately, coinciding every value of the resulting matrix. The eigenvalue diagonal matrix, however, presents some error in the values after half of the matrix.

[32] describes the matrix of eigenvalues as the variances of the dataset analysed in every dimension. It also defines a very common method used in PCA to reduce the dimensionality of the data.

In PCA, once the eigenvalue decomposition is performed, some dimensions are removed from the analysis because they do not contain any important information. The technique to measure the information contained in an eigenvalue is called the Proportion of Variance (*PoV*). This measure is defined as:

$$PoV_k = \frac{\sum_{i=0}^{k-1} \lambda_i}{\sum_{j=0}^{N-1} \lambda_j} \quad (37)$$

Where k represents the number of eigenvalues evaluated in the analysis considering that they are ordered decreasingly, from highest to lowest.

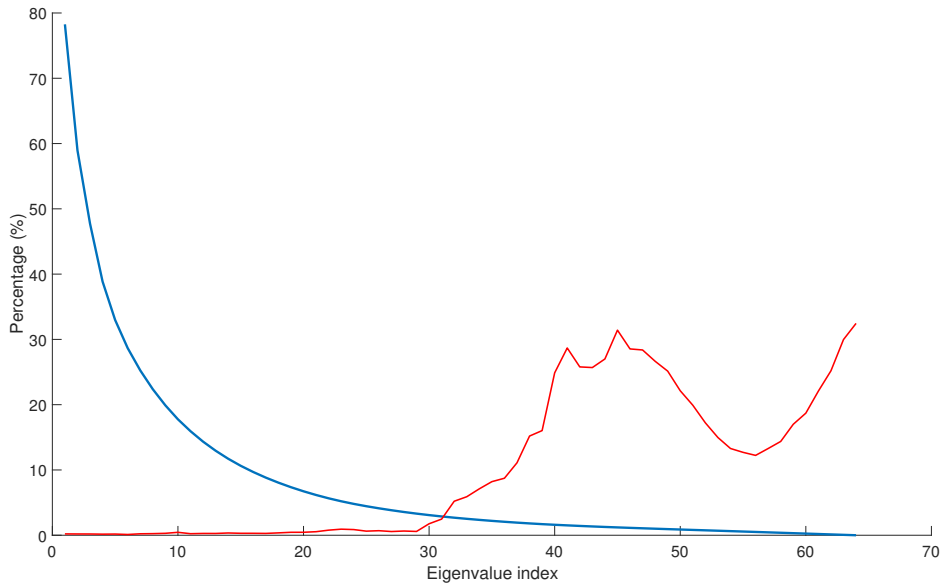


Figure 29: Average Percentage of Variance (*PoV*) and error with respect to MatLab functions. Blue Line: Percentage of information loss based on the *PoV*. Red Line: Error of each eigenvalue with respect to the MatLab function.

In this case, instead of removing eigenvalues from the analysis, some of the values used in the transformation are wrong. The purpose of this test is to evaluate how significant the error of this components is with respect to the correct ones.

Figure 29 represents the *PoV* evaluated as information loss in blue and the relative error as it is shown in equation 38 in red. The horizontal axis contains the index of the eigenvalues in descending order.

$$Err(i) = \frac{|\lambda_{LAPAK}(i) - \lambda_{MatLab}(i)|}{|\lambda_{MatLab}(i)|} \quad (38)$$

Note that the error in the eigenvalues starts to increase after one half of the length of the vector, and even reaches values around 30%. However, 97% of the information is already contained in the first 32 eigenvalues, which means that the error in the smallest values will barely affect the transformation.

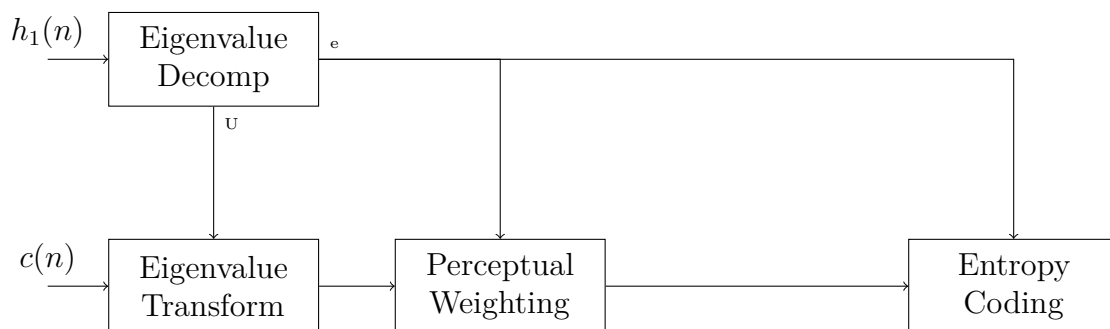


Figure 30: Block diagram of the FD-CELP encoder using Eigenvalue decomposition

The transformation is similar to the Vandermonde decomposition, considering that it is necessary to calculate the transformation matrices for every subframe evaluated. In this case, as mentioned before, the transformation will be performed using the eigenvector matrix, which transports the data into the decorrelated coordinate system, and the eigenvalues will represent the perceptual model for the weighting and the envelope based coding.

In the same way as with the DCT, the eigenvalue decomposition produces only real values if the transformed matrix is real. The eigenvalues are also positive so they can be directly used in the perceptual weighting and the encoder.

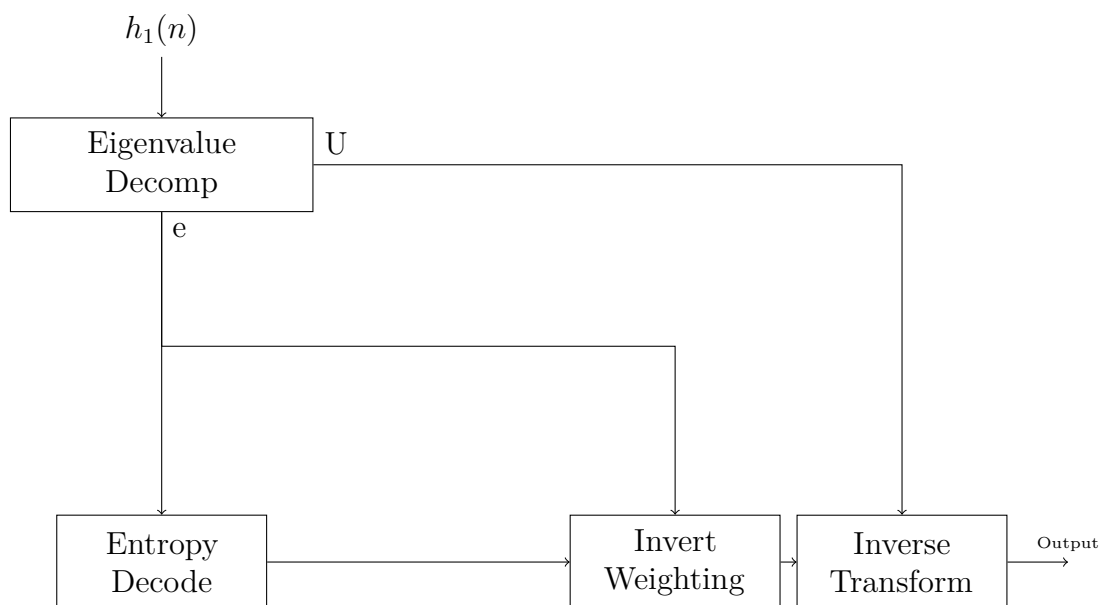


Figure 31: Block diagram of the FD-CELP decoder using Eigenvalue decomposition

4.6 Quantisation and encoding

As shown in figure 16, the encoding of the innovative target vector has two parts, a gain factor γ_{innov} used to quantise the signal to the correct level, and the actual

codebook that minimizes the error criterion. In the case of ACELP codecs, they try to optimize the SNR in the perceptually weighted domain. The target function to be minimized, explained in [5], is:

$$\mu(x, \gamma_{innov}) = \|H(x - \gamma_{innov}\hat{x})\|^2 \quad (39)$$

Where x is the target excitation, \hat{x} represents the quantised excitation and H is the convolution matrix of the weighted synthesis filter. The task of the quantiser is to find the optimum γ_{innov} in order to minimize 39.

However, the estimation of the optimum γ_{innov} assumes that the values of the target vector are decorrelated, which is not a realistic case. This can be translated into an efficiency loss in terms of bit consumption. In the actual ACELP encoding techniques, the optimisation of the SNR is performed using an analysis-by-synthesis algorithm, which compares the target excitation in the perceptually weighted domain to obtain this SNR value. As a result, the only optimum solution to find the correct codebook is to evaluate the SNR value with all the potential codebooks. That is the reason why ACELP codecs nowadays use different algorithms [2, 33] to optimize codebook search, accepting a compromise between quality and complexity.

As it is mentioned before, the objective of the target transformations is to decorrelate its samples, allowing the use of simpler quantisation and encoding of the excitation.

4.6.1 Envelope based arithmetic encoder

The encoder chosen for this task is an envelope based arithmetic encoder described in [1]. This encoding technique interprets the model of the spectral envelope as a probability distribution of the bit consumption necessary to encode each sample.

Considering that this implementation can be seen as a variation of TCX, it is possible to use the envelope model given by the LPC coefficients as the probability model for the encoder. The quantised LPC coefficients are transmitted in the bitstream to the decoder, this means that using the quantised LPC coefficients to define the weighted synthesis filter, the spectral envelope can be directly decoded from the bitstream. This is one of the necessary changes in the generation of the target vector, as explained before.

For a better performance, the target vector is weighted by the spectral envelope from each transformation. This perceptual weighting is the one that appears in the block diagrams of the transformations, so the target vector going into the encoder will be already weighted.

Figure 32 shows the block diagram of the quantisation loop based on what is explained in [1]. Where $x(n)$ represents the transformed target vector and $h_1(n)$ stands for the transformed response of the weighted synthesis filter.

1. A fixed quantisation gain γ is defined in order to scale the target before the quantisation.
2. The scaled target is quantised to the corresponding amplitude levels, limiting its amplitude to a finite number of values.

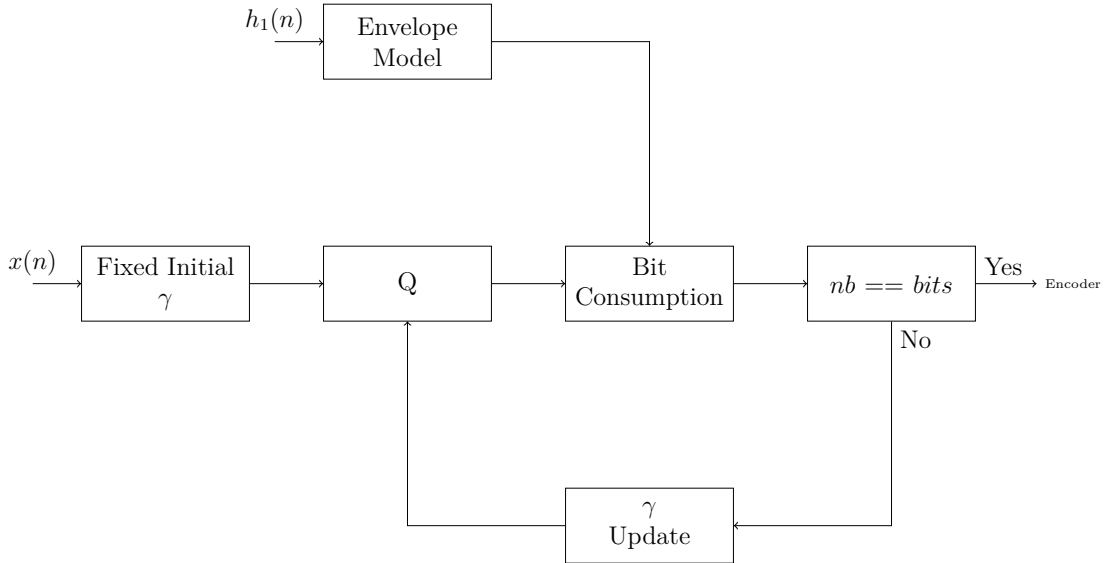


Figure 32: Quantiser structure of the envelope based encoder proposed in [1]

3. The envelope model calculated from the synthesis filter and the quantised target are used to estimate the number of bits necessary to encode the signal $nb = \sum_k -\log_2(p(\hat{y}(k)))$.
4. If the number of bits necessary to the encoding (nb) coincide with the available bits for the subframe ($bits$), the quantised target is sent to the encoder.
5. If the number of bits does not match the available bits, the quantisation gain is updated and the process is repeated with the new gain.

An interesting effect that was observed while implementing this, was that the probability distribution tends to underestimate the number of bits necessary to encode the target. The encoder is meant to start encoding the lower frequencies and encodes all the frequency bands until there are no available bits left. As a result, if the quantisation scales the signal for a higher number of bits than those available, noise filling is used in all the frequency bands for which there were no bits left and were quantized to zero, generating a noisy signal.

As a solution to this problem, it was decided to use the actual encoder in every iteration of the quantisation loop. This way, the number of bits is not an estimation, rather the actual required value and the encoding has a similar complexity to the probability estimation, so it does not increase the computational cost of the algorithm.

Another modification introduced to the quantisation loop to reduce the number of iterations is a previous estimation of the quantisation gain. Considering that the target has been weighted by the spectral envelope, it is possible to estimate the initial gain for the loop in every subframe. The MDCT-based transform coding in the EVS already includes a function (`SQ_gain`) to estimate the number of bits of a quantisation, which may not be optimal for the transform coding of this thesis but provides better results than using a fixed value as before.

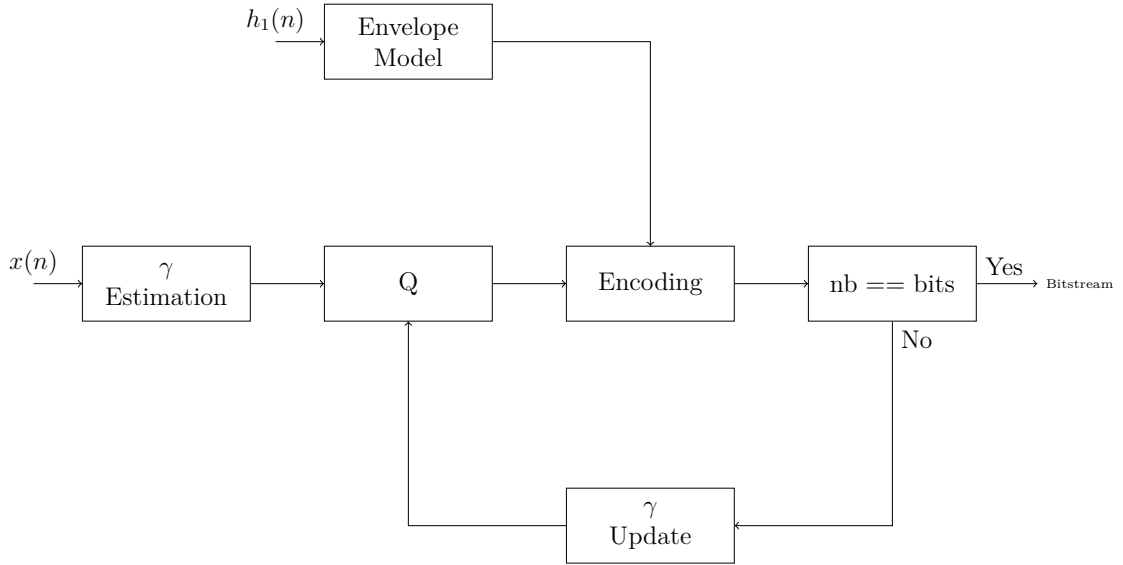


Figure 33: New quantiser structure using the encoder instead of estimating the probabilities

Figure 33 represents the block diagram of the quantisation loop with the new structure proposed for this thesis. In it, the quantisation gain is estimated before entering the loop. Getting a gain value more similar to the right one from the beginning helps to reduce the number of iterations of the loop, which is one of the main sources of complexity.

The second modification is the substitution of the bit estimation with the actual encoder. This gives a more accurate measurement of the number of bits used in order to avoid under- or overestimation of this parameter.

Using the encoder to evaluate the number of bits required also means that once the available number of bits is reached, the target has been optimally encoded, so it is not necessary to run the encoder again.

5 Testing and Results

To evaluate the performance of the different algorithms implemented, multiple tests have been carried out in order to analyse the different properties of each algorithm. Both objective and subjective measurements will be included in this analysis.

The objective characteristics evaluated are the objective quality estimated using a POLQA test, a study of the complexity of each algorithm evaluating the number of operations required in it, and an analysis of the entropy improvement measured by how well the algorithms decorrelate the signals. To estimate the subjective quality of the algorithms, a MUSHRA test has been carried out at different encoding bitrates to evaluate how the signal degradation is perceived.

In order to have a first hint on the direction that the results will take, the perceptual SNR of the different test files has been calculated. The test items used are male and female speech files in three different languages, including clean speech and mixed speech signals with different background noises. The clean sound samples were extracted from [34] and the different background noises were added later on the clean files.

File	ACELP	Vandermonde	DFT	DCT	Eigenvalues
ger M street	10.2803	6.7277	7.6035	7.4145	8.4019
ger M clean	10.0682	6.9867	7.7202	7.5365	8.3867
ger F office	9.7351	6.2791	7.3066	7.0751	7.9666
ger F clean	10.5063	6.6647	7.4471	7.3797	8.2660
fre M music	9.2559	6.1174	6.8396	6.7621	7.5602
fre M clean	10.5778	6.8202	7.6993	7.5772	8.6650
fre F car	7.1981	5.4464	6.1396	6.0686	6.3295
fre F clean	10.0543	6.2025	7.1357	6.9487	7.8501
eng M babble	10.4459	6.5782	7.6678	7.5154	8.8269
eng M clean	10.3658	6.7173	7.6391	7.4264	8.5083
eng F babble	10.6210	6.3490	7.5012	7.3985	8.5871
eng F clean	10.2082	6.3192	7.2705	7.1544	8.1607

Table 3: Average perceptual SNR of the different test files processed with the different algorithms

Table 3 shows the results of the perceptual SNR calculation for all the files analysed. The name of each file shows the language of the speech, the gender of the voice and the background noise type added to the signal.

The perceptual SNR is an objective measurement that calculates the SNR of a processed signal in a perceptually weighted domain in order to obtain a more accurate evaluation of the performance of the processing algorithm. The SNR value is calculated by windowing the signal and calculating the SNR value for every windowed segment. This value allows for the evaluation of the SNR along the whole signal. The average of the SNR values of all the segments is usually calculated and it is called segmental SNR. This averaging is performed on a logarithmic scale, ignoring the values that go to infinity. If the SNR values are calculated applying a perceptual

weighting to the windowed segments, the average value is called perceptual-segmental SNR.

It can be seen that ACELP outperforms all of them. However, comparing the different transformation algorithms, eigenvalue decomposition is, as expected, the best of the the four, followed by DFT and DCT which are almost the same. The last performance is seen in the Vandermonde decomposition.

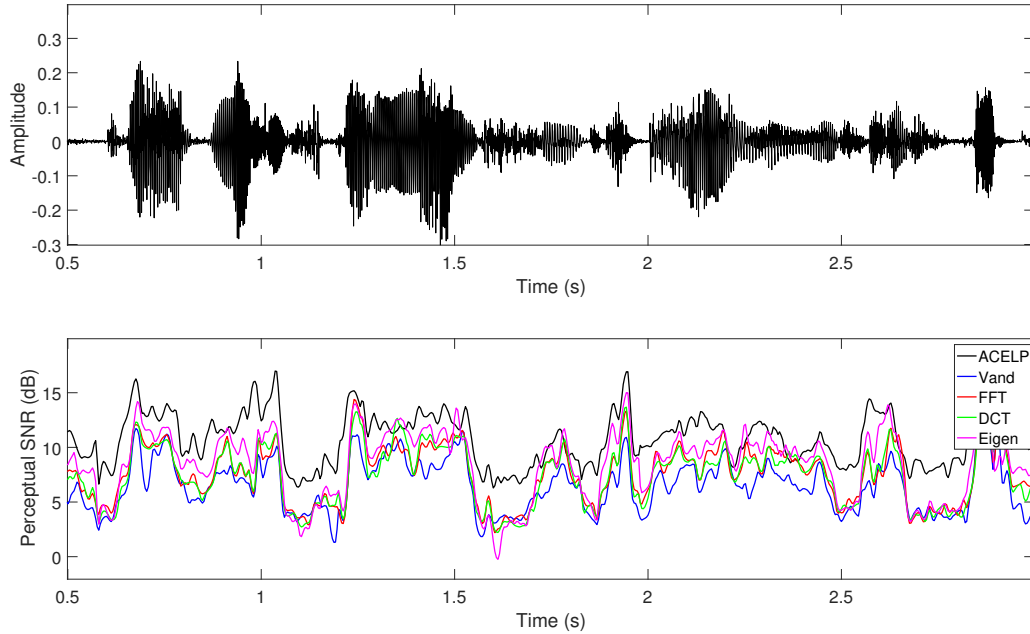


Figure 34: Perceptual SNR of english male noisy speech. Top: Time representation of the signal. Bottom: Perceptual SNR over time.

The evolution of the perceptual SNR over the file is represented in figure 34 in order to analyse the ordering of the files in table 3. Eigenvalue decomposition has a noticeably better quality most of the time while the other three transformations are almost the same. Nevertheless, Vandermonde's SNR drops at certain points like 0.75 s or 1.5 s, which may produce the decrease in the average value of this transformation.

Even though this measurement provides a good starting point for the evaluation of the algorithms, the results are not very reliable, considering that it only compares the energy of the files in a perceptually weighted domain.

5.1 Complexity Study

The main goal of the thesis is to evaluate the different transformation algorithms in order to find a more efficient encoding technique than the actual ACELP implementation. So one of the main evaluation methods for the quality of the algorithms is a complexity study.

In this section, the complexity of the algorithms is estimated by analysing the number of operations present in the encoding task. These operations are then weighted with respect to a reference [35] in order to get a simple value as a result. This resulting value represents the Weighted Million Operations Per Second (WMOPS) and is the measure that will be used to compare the algorithms' complexity.

Note that the eigenvalue decomposition was performed using an external pre-compiled library, therefore it is not possible to know which operations are carried out inside its functions. To compare the complexity of the eigenvalue decomposition, the execution times have been calculated too. Even though it is not a reliable measurement, considering that the conditions inside the machine running can return very different results, these measurements have been done whilst trying to preserve the same conditions between the different algorithms in order to obtain a hint on the complexity of this transformation.

Despite the fact that it is not possible to know which operations are performed inside the pre-compiled library, its documentation provides information about the algorithms used in every step of the eigenvalue decomposition. Knowing the complexity of these algorithms, the complexity of the algorithms will also be evaluated step by step in order to find out the parts where the algorithms may have the highest complexity.

The file evaluated in the complexity study is a mix of multiple sounds from clean speech to music and noise, in order to evaluate the performance of the algorithm in all the possible situations. The representation of the results will be in terms of WMOPS in the worst case scenario.

In a first evaluation of the complexity, the results obtained were the following:

Algorithm	WMOPS
ACELP	48.16
Vandermonde	211.65
DFT	99.57
DCT	105.55

Table 4: Initial complexity study

The results in Table 4 are considerably above the complexity of the ACELP encoder. After analysing every block of the transform encoding algorithms, an extremely high number of WMOPS was concentrated in the quantiser loop. This happened because of the silence subframes, which being all zeros produced an unexpected problem in the quantisation.

As explained before, the quantisation loop tries to find the optimum gain in order to encode the sequence using all the bits available. When the sequence is only composed of zeros, the quantisation loop can never find an optimum gain and iterates until the maximum number of iterations established is reached.

This problem was solved by adding a silence detector before the quantisation, which adds some complexity to the algorithm but solves the problem of the zero quantisation. The complexity after adding the silence detection is:

Algorithm	WMOPS
ACELP	48.16
Vandermonde	194.9
DFT	74.5
DCT	96.8

Table 5: Complexity values with silence detection in the encoder

Table 5 shows how the complexity of the algorithms has been reduced, however, the complexity is still not as low as in the ACELP. This happens due to the estimation of the initial gain, which reduces the number of iterations necessary with respect to the fixed implementation although it is still not enough.

However, the estimation of the gain is done using an algorithm optimized for the MDCT-based transform coding of the EVS. For the final evaluation, an optimized gain estimation will be assumed, limiting the number of iterations to 1 or 2:

Algorithm	WMOPS
ACELP	48.16
Vandermonde	160.9
DFT	49.9
DCT	48.6

Table 6: Complexity values assuming an optimized gain estimation

The result of the last assumption can be seen in Table 6. The algorithms based in DCT and DFT have now reached the level of the ACELP in terms of complexity. This reduction in complexity, leads to the assumption that it could be possible to further optimize the encoder in order to get a better performance than the ACELP.

The complexity of the Vandermonde decomposition is however extremely high. DFT and DCT, as explained before, already have a predefined transformation matrix which allows for a computationally efficient implementation, providing an algorithm with $O(n \log(n))$ complexity. Vandermonde and eigenvalue decompositions, however, have to generate the transformation matrix for every subframe. For the Vandermonde decomposition it is possible to define these steps with their corresponding complexity:

1. Calculate the Vandermonde matrix. $O(n^2)$
2. Inversion of the Vandermonde matrix. $O(n^2)$
3. Matrix multiplication to obtain D. $O(n^2)$
4. Matrix multiplication to apply transformation. $O(n^2)$

And the same steps can be seen in the eigenvalue decomposition:

1. QR algorithm to calculate the matrices. $O(n^3)$

2. The eigenvalue matrix is already calculated, no need to invert the transformation here.
3. The inverse of the Eigenvector matrix is its own transpose.
4. Matrix multiplication to apply transformation. $O(n^2)$

The main problem of the eigenvalue decomposition is the QR algorithm, with a complexity of $O(n^3)$. The matrix multiplication in both algorithms make the complexity grow to unacceptable levels. This can be seen in the execution time of every algorithm:

Algorithm	Time (ms)
ACELP	610
Vandermonde	1457
DFT	875
DCT	907
Eigenvalues	11491

Table 7: Execution time of the original implementation of the algorithms encoding an 8 seconds file

Note that the execution time of the Eigenvalue decomposition is even higher than the length of the file encoded.

As a conclusion, in terms of complexity, DCT and DFT encoding could be viable if some optimisation was carried out in the actual encoder. This optimisation would include the silence detection and the quantisation gain estimation.

In order to make Vandermonde or eigenvalue decomposition competitive with the actual encoding algorithms it would be necessary to optimise the matrix multiplications and the search of these transformation matrices.

5.2 Entropy study

As it is explained before, the objective of the transformations is to eliminate the correlation existing between the samples of the innovative target vector. This is why another measure of the quality of the algorithms will be the degree of decorrelation that the transformations can achieve in the samples of a subframe.

The entropy of a sequence of values represents the amount of information that the sequence contains and, therefore, it is proportional to the minimum number of bits necessary to encode a signal without loss of information. If a sequence is highly decorrelated, its entropy value will be higher than another sequence whose values have some correlation. [36] describes a method to calculate the entropy of a vector $x(n)$ by using its autocorrelation matrix R_{xx}

$$H(R_{xx}) = \frac{1}{2} \ln \left[(e2\pi)^N |R_{xx}| \right] \quad (40)$$

where N represents the size of the data sequence. However, in terms of this thesis, the differential entropy explained in [36] is much more interesting. This measure represents how much the entropy changes from the original sequence of data to the transformed one.

$$A = T^H R_{xx} T \quad (41)$$

Equation 41 represents how to apply the transformations on the autocorrelation matrix of x , where T stands for the transformation matrix of the corresponding transformation. Considering this, the differential entropy will be:

$$\Delta H = H(A) - H(R_{xx}); \quad (42)$$

However, the Vandermonde decomposition and the DFT generate complex values when applied; this results in some problems considering that the determinant of a complex matrix is a complex number and it is necessary to calculate the logarithm of that value.

Although dealing with complex numbers complicates the analysis of the differential entropy, it is possible to interpret the entropy as how decorrelated the signals are. This way, considering A as the transformed correlation matrix, the more diagonal A becomes, the higher the differential entropy will be.

Three subframes have been selected with different sounds produced in order to compare how well the algorithms decorrelate the signals under different conditions. These subframes represent a tonal sound produced in a voiced phoneme, an unvoiced phoneme and just noise without speech on it.

The autocorrelation matrices represented in figures 35-43 are normalized and plotted in a logarithmic scale, which means that 0 dB stands for the maximum correlation. The color scale shown in figure 35 is the reference used in all the images.

Figures 35, 36 and 37 show the autocorrelation matrix of a voiced sound and its corresponding transformations calculated as in equation 41.

The autocorrelation matrix presents a very high correlation with neighbour samples, and it is even possible to appreciate some periodicity as the correlation decreases and grows along the vector.

36 contains the autocorrelation matrices after applying Eigenvalue and Vandermonde decomposition. It is noticeable that the eigenvalue decomposition completely decorrelates the signal, as it is theoretically the optimum tool for the decorrelation task.

Vandermonde decomposition shows a high level of decorrelation in and around the middle of the matrix, which, considering the frequency mapping of this decomposition, correspond to the higher frequency components, while the low frequencies still have a high correlation.

As figure 37 shows, the autocorrelation in the DFT domain looks very similar to that of the Vandermonde, however, the Vandermonde decomposition manages to obtain a higher decorrelation of the samples.

Finally, the DCT transform also performs a good decorrelation of the samples, better than the Vandermonde. as it has still some correlation in the low frequencies.

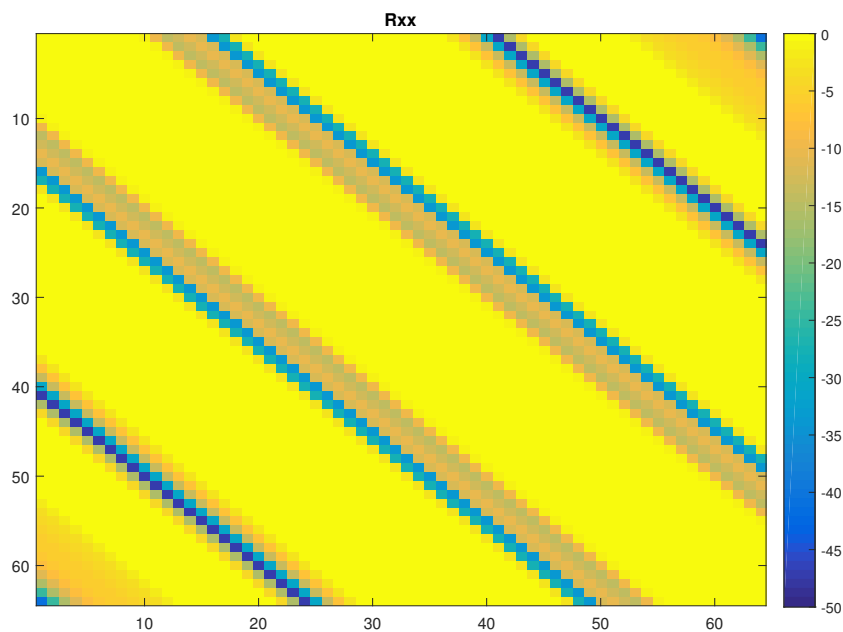


Figure 35: Autocorrelation matrix of a frame containing a voiced sound

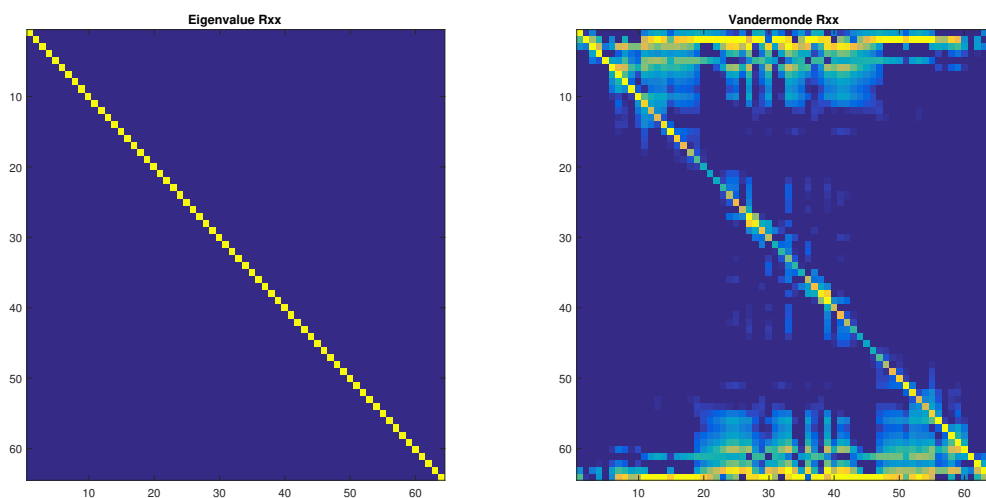


Figure 36: Transformed autocorrelation matrices of a voiced sound. Left: Eigenvalue decomposition. Right: Vandermonde decomposition

However, the effect of the transformations also has to be evaluated in different kinds of sound. Figure 38 shows the autocorrelation matrix of an unvoiced sound. As it is explained in section 3, unvoiced sounds are produced by a turbulent airflow and have a behaviour more similar to noise. Note that in this case, the signal has lost the periodicity present in the voiced sounds and the correlation decreases as it moves

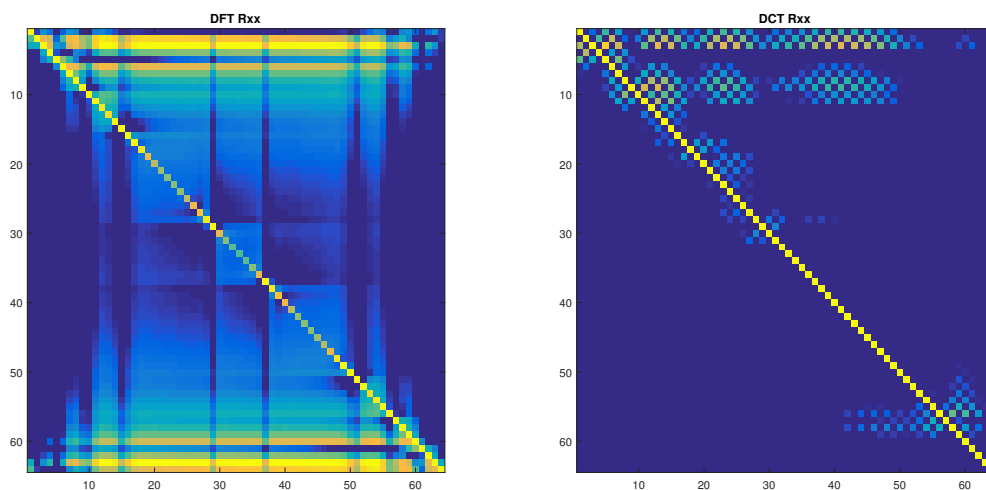


Figure 37: Transformed autocorrelation matrices of a voiced sound. Left: DFT. Right: DCT

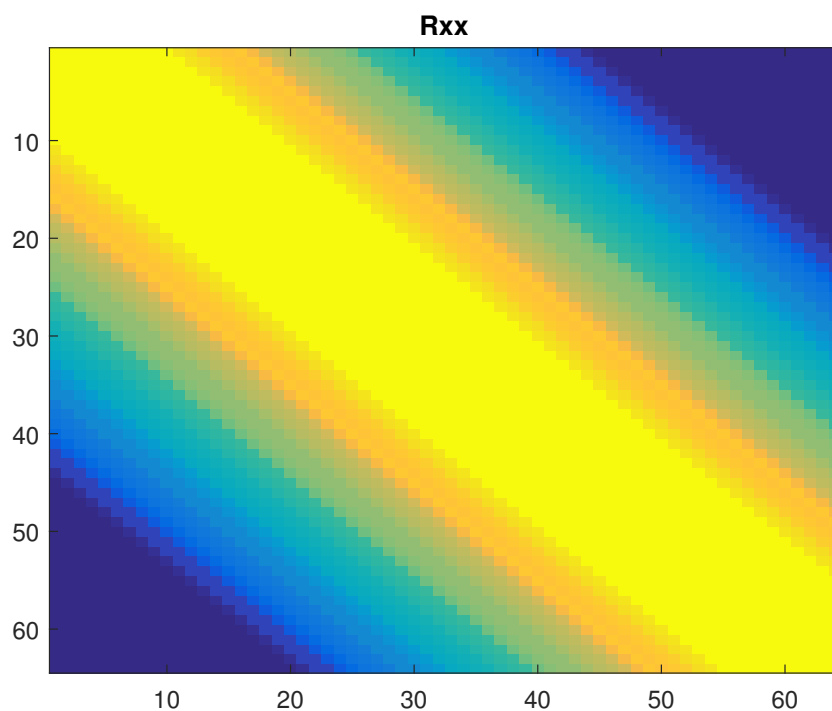


Figure 38: Autocorrelation matrix of a frame containing an unvoiced sound

away from the diagonal. This means that the samples only maintain correlation with their own neighbours.

Eigenvalue decomposition provides the same result as before, as the transformation adapts to completely decorrelate the signal at every subframe.

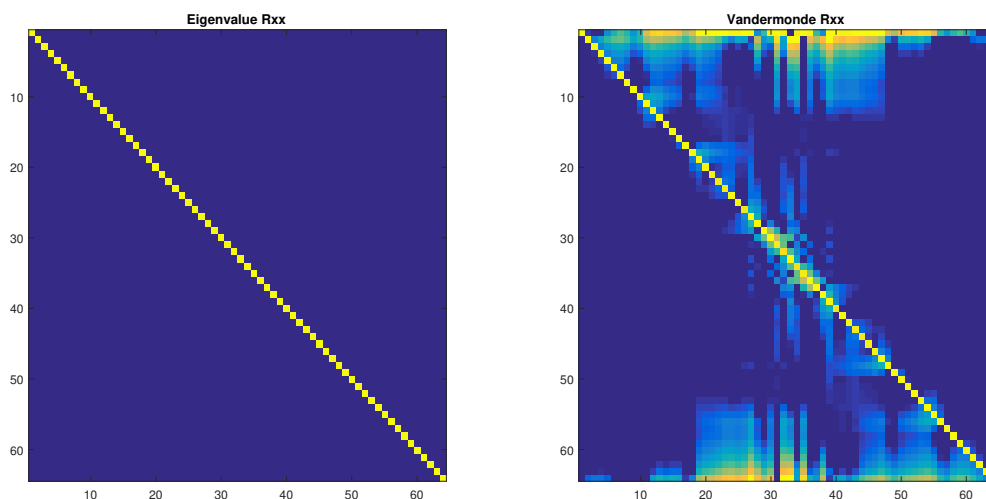


Figure 39: Transformed autocorrelation matrices of an unvoiced sound. Left: Eigenvalue decomposition. Right: Vandermonde decomposition

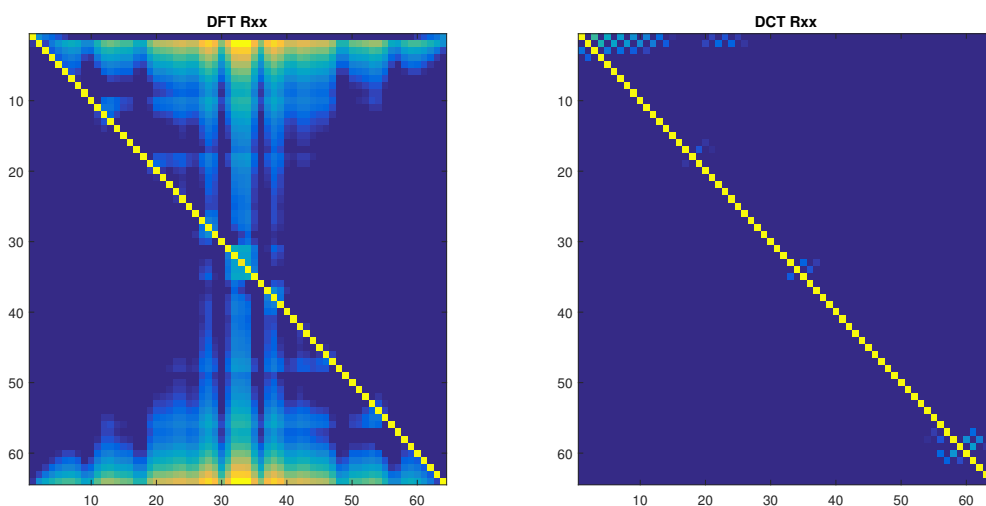


Figure 40: Transformed autocorrelation matrices of an unvoiced sound. Left: DFT. Right: DCT

Vandermonde decomposition has a similar behaviour to the previous case, decorrelating the signal but still keeping some similarities with the DFT representation.

The result of the DFT is now very similar to the eigenvalue decomposition. As the original signal is already slightly decorrelated, the DCT manages to decorrelate it almost completely. This effect is seen in the noise example too.

The signal in the example displayed in figures 41, 42 and 43 is a silence frame where only the background noise can be heard. In this case, the autocorrelation matrix of the frame is much more similar to a diagonal than in the previous cases.

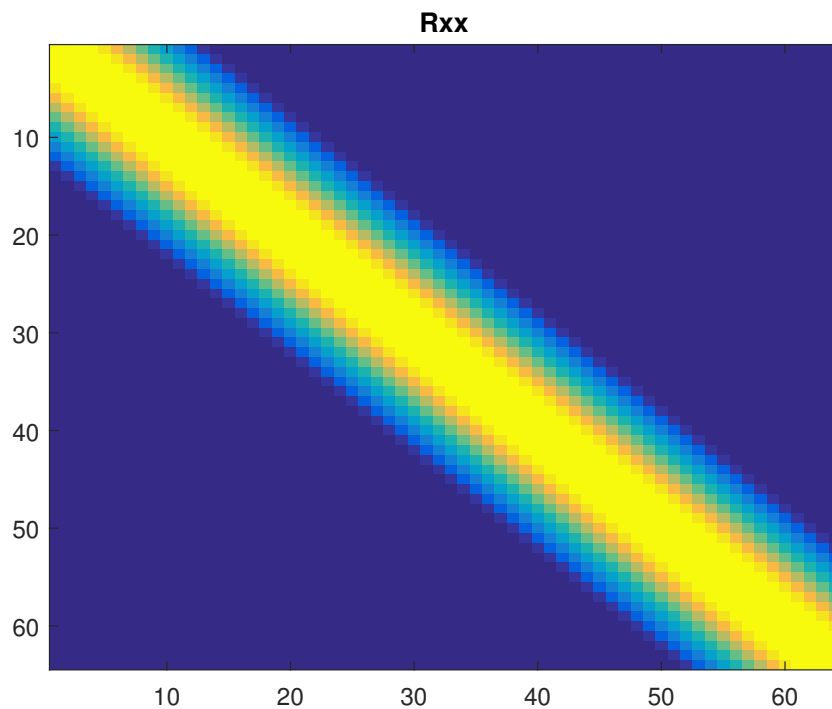


Figure 41: Autocorrelation matrix of a frame containing silence

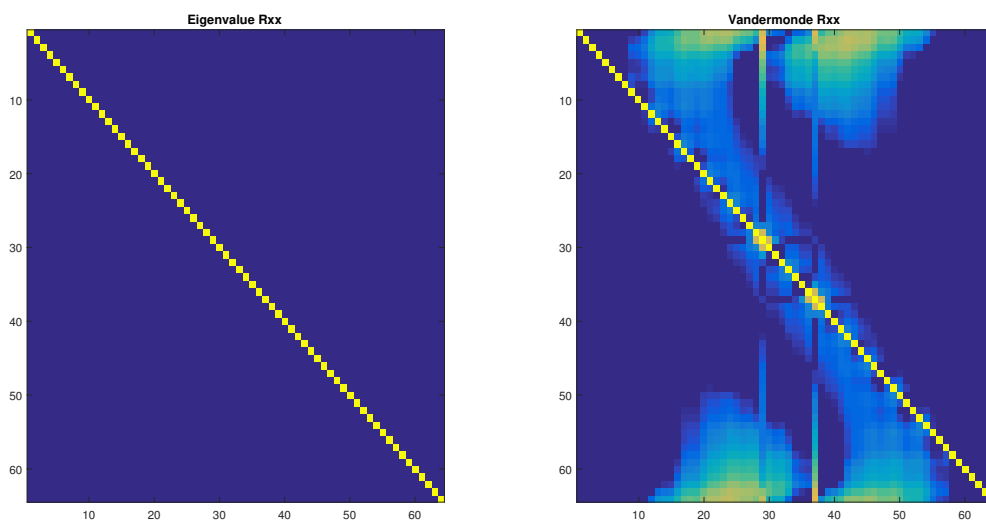


Figure 42: Transformed autocorrelation matrices of silence. Left: Eigenvalue decomposition. Right: Vandermonde decomposition

Eigenvalue decomposition and DCT have a very similar behaviour to the previous case, providing a complete decorrelation of the samples.

Being an already decorrelated signal, the result for DFT and Vandermonde do

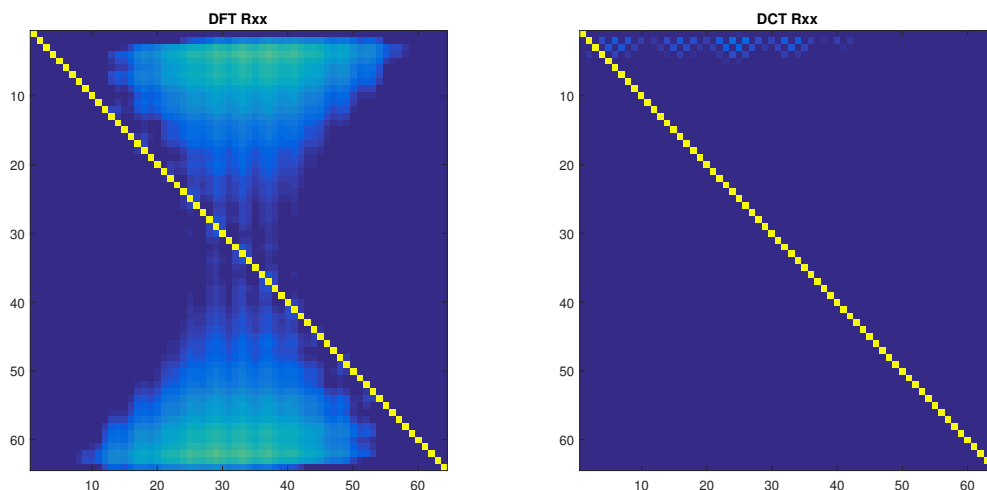


Figure 43: Transformed autocorrelation matrices of silence. Left: DFT. Right: DCT

not have very high values outside the diagonal, however, they are still not completely decorrelating the signal. An interesting effect in Vandermonde is that some of these values are concentrated around the diagonal, which means that those samples are still correlated with their neighbours.

As a conclusion, as it is proven in the theory, the eigenvalue decomposition is the transformation that manages to decorrelate the signals the most. DCT also provides a very high decorrelation ratio, which is why it is more commonly used in compression tasks than the DFT.

DFT, as expected, does not completely decorrelate the signal with a finite number of samples in the transformation. If the length of the transformation grew to infinity, the decorrelation would coincide with the eigenvalue decomposition.

The behaviour of the Vandermonde decomposition is very similar to the DFT considering their resemblance in terms of spectral representation of the signal. It manages to reach a higher decorrelation than the DFT as explained before; however, it is not as high as the eigenvalue decomposition or DCT.

As stated in [30], Vandermonde decomposition suffers some numerical instability when the length of the transformations grows over 64 samples. This numerical instability appears when the roots of the $A(z)$ coefficients are calculated, as explained in section 4.5.3. This is probably the reason why the Vandermonde decomposition autocorrelation matrix looks more similar to the DFT than to the complete decorrelation that it should theoretically perform.

5.3 POLQA test

Perceptual Objective Listening Quality Assessment (POLQA) is an objective test defined in the ITU-T Recommendation P.863 [37]. This algorithm evaluates the quality of audio signals using a perceptual model to reproduce how a person would perceive the sound. POLQA is useful in order to provide fast results on the quality

of the signal without the need for gathering a group of listeners. This algorithm is explained in [38, 39].

The POLQA test will be used to analyse the quality of the four transformation algorithms implemented in this thesis. The ACELP encoder in EVS will be evaluated and used as a reference for the other values. The algorithms will be tested using different bitrates to encode the samples, which are 16.4, 24.4 and 32 kbps, in order to analyse the degradation of the quality when the available number of bits in the encoder is reduced.

It is also possible to establish a relationship between the degree of decorrelation of the target and the quality of the processed signals. As stated before, the encoding becomes more efficient when the samples of the target vector are decorrelated. According to this, the Eigenvalue decomposition should provide the best results, followed by the DCT, Vandermonde and DFT, which will have the worst quality.

The results of the POLQA test are represented using a Mean Opinion Score (MOS) value. MOS is represented in the range 0-5, where 0 represents the worst quality and 5 indicates the best result.

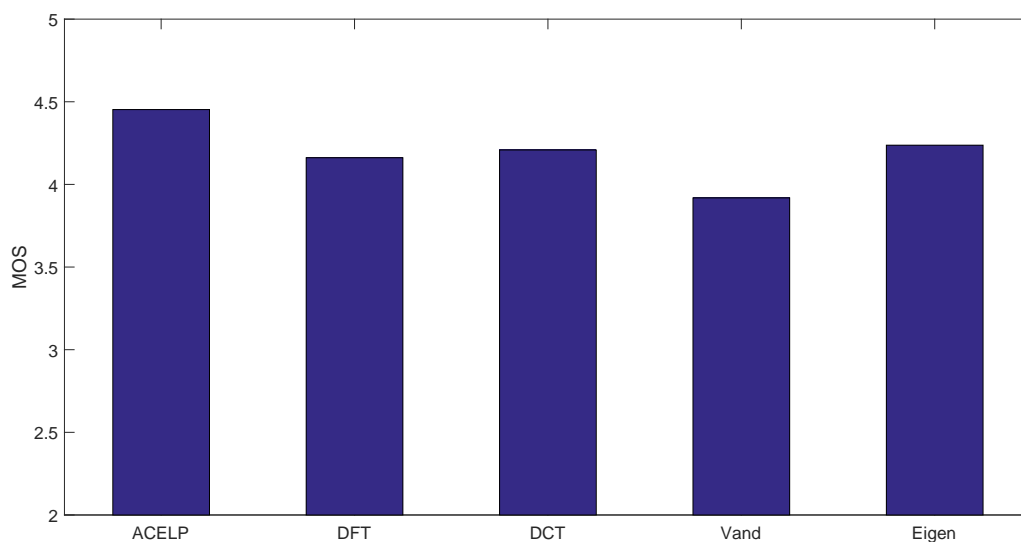


Figure 44: Average POLQA results at 32kbps evaluated on 972 samples

Figure 44 shows the results of the POLQA test after evaluating the samples processed at 32 kbps. It is noticeable that the ACELP implementation is better than the four transformation algorithms, however, all the samples are around 4 MOS, which means that the quality of the samples is considerably high.

As it was expected, considering the decorrelation results, the eigenvalue decomposition has the best performance amongst the transformation algorithms. The eigenvalue decomposition is closely followed by the DCT, as it achieved an almost complete decorrelation of the signal.

DFT and Vandermonde decomposition are in the last positions. However, DFT achieves a better quality than Vandermonde. Section 5.2 states that the Vandermonde

decomposition is not working as it theoretically should due to numerical instabilities caused by the root estimation algorithm. This may be the reason why the quality of the Vandermonde decomposition has a lower value than the rest of the algorithms. Another reason may be the effect of the tilt filter explained in section 4.4, which, in order to reduce the noise in the low frequencies, can cause a slight energy loss in the signal.

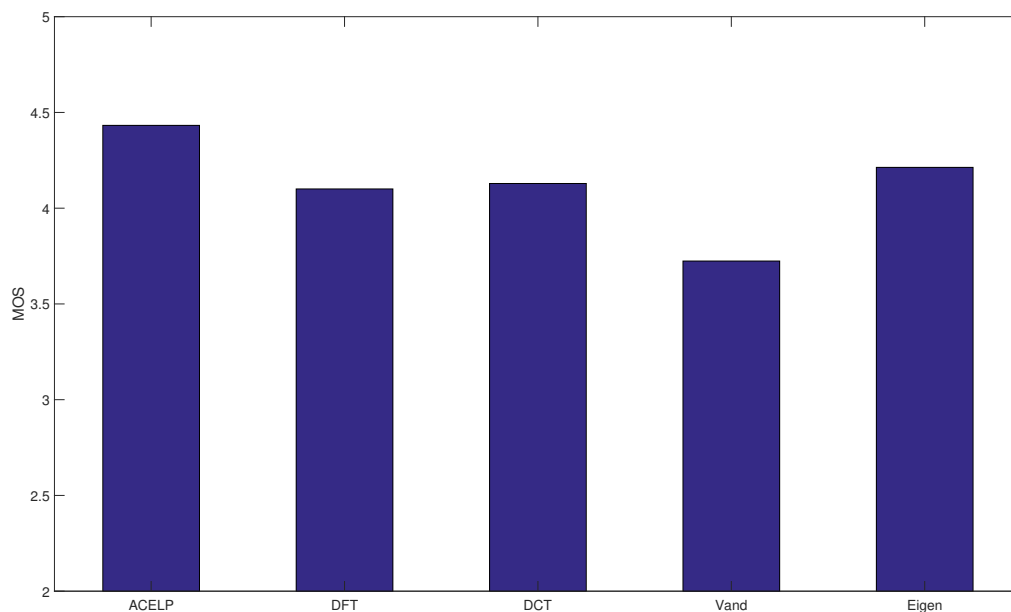


Figure 45: Average POLQA results at 24.4kbps evaluated on 972 samples

As can be seen in figure 45, the quality of the samples decreases from the 32kbps version to the one at 24.4. However, the order of the different algorithms is still the same as before. First, the eigenvalue decomposition, followed by the DCT, DFT and Vandermonde decomposition. ACELP still outperforms all the transformation algorithms.

Figure 46 shows the average MOS score of the different algorithms encoded at 16.4 kbps. The undergoing quality loss is noticeable in the Vandermonde decomposition, which has lost almost 1 MOS against its implementation at 32kbps. Eigenvalue decomposition and DCT have decreased too, laying now behind the DFT, which has not been as degraded.

To evaluate the actual degradation of the signals when the bitrate is reduced, figure 47 represents the average MOS scores of the five algorithms at the three evaluated bitrates.

The good performance of ACELP at low bitrates can be seen in the figure, where the signal quality is barely reduced compared to the highest bitrate. This does not happen with the transformation algorithms. Vandermonde, which has the lowest MOS score of the four algorithms gets significantly degraded from 32 kbps to 16.4,

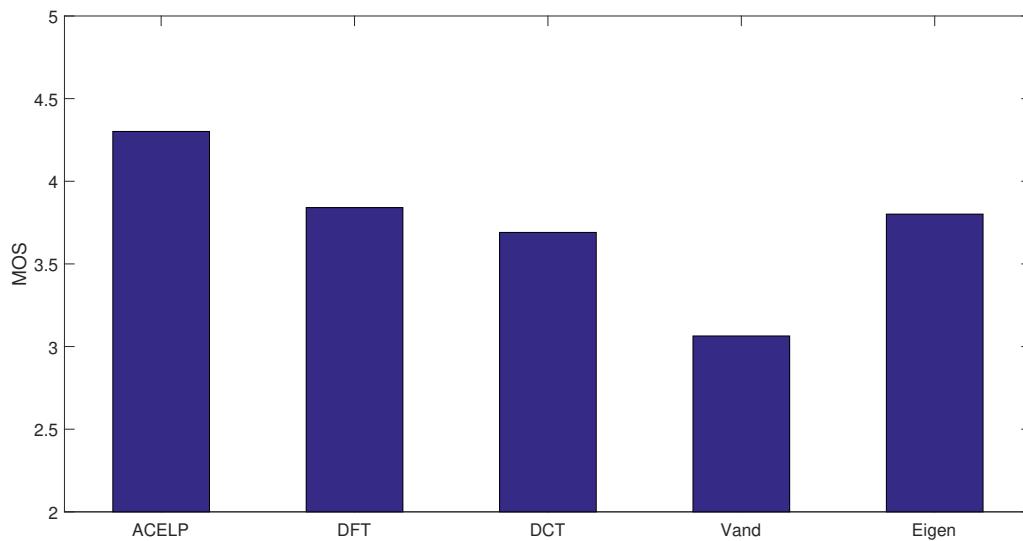


Figure 46: Average POLQA results at 16.4kbps evaluated on 972 samples

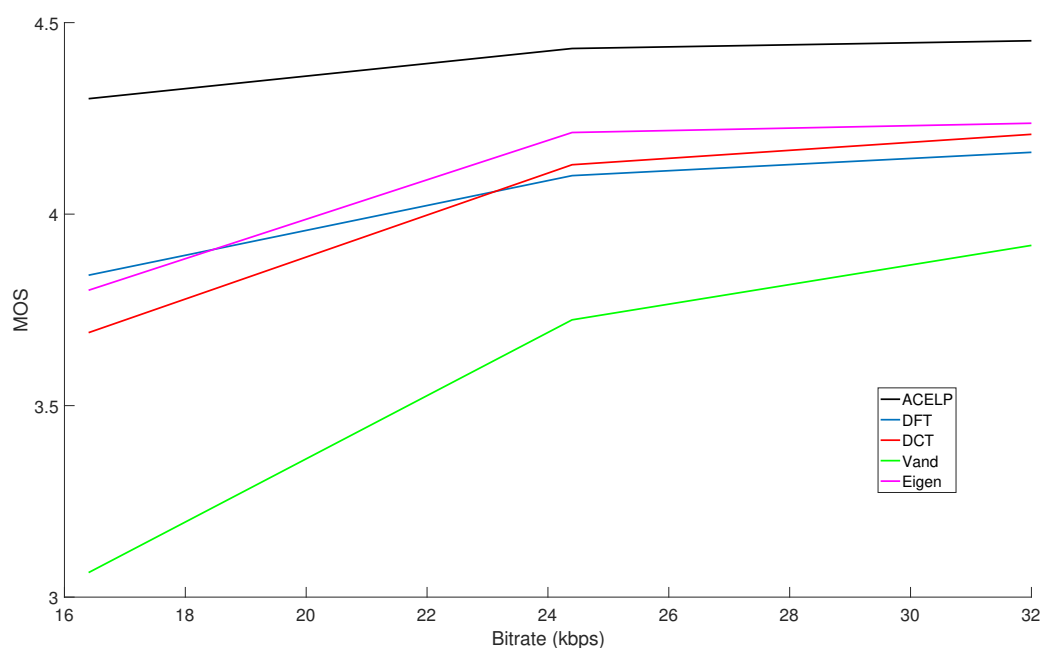


Figure 47: Bitrate against MOS representation of the different algorithms

losing almost 1 MOS point as shown before. Something similar happens to the eigenvalue decomposition and the DCT. This effect is not as steep in DFT, which makes it outperform the previous two at 16.4 kbps.

Notice that the main quality reduction happens when the bitrate decreases from 24.4 to 16.4 kbps, whereas it is not so big between 32 and 24.4. This leads to the

conclusion that the transformation algorithms would not be viable at lower bitrates.

5.4 MUSHRA Test

POLQA is a good test to obtain fast results and get a hint on the perceived quality of the implemented algorithms. However, every person is different and a perceptual model that tries to be a general representation of how humans perceive sounds is not as accurate as a real person. Therefore, a MUSHRA test has been carried out in order to obtain a more accurate measurement of the perceived quality of the algorithms.

The Multiple Stimuli with Hidden Reference and Anchor (MUSHRA) test [40] is a methodology for the evaluation of the subjective quality of an audio signal, which is widely used to evaluate the perceived quality of lossy audio compression algorithms. It is commonly used to compare the quality of different compression algorithms applied on a same sound file. The samples of the multiple algorithms applied on one sound file are called conditions.

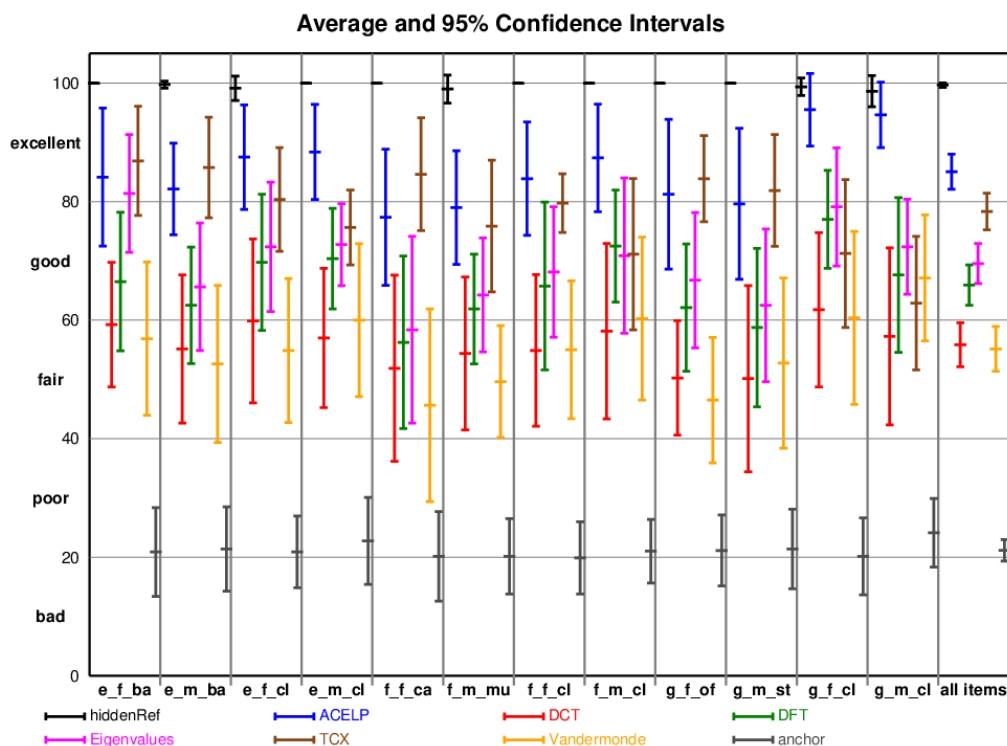


Figure 48: Average MUSHRA results for 32kbps encoding using 8 expert listeners.

This test consists in comparing the different conditions with respect to a reference, evaluating them in terms of how similar they are to this reference. What makes this test so useful for the evaluation of sound quality is the possibility of listening to all the conditions at the same time, allowing to jump from one to another in order to compare them.

The test shows the different conditions at the same time without displaying their names so the rating is objective. The test allows the user to rate the conditions comparing them to the reference on a scale from 0 to 100. In order to estimate an absolute rating independent from the listener, a hidden reference and anchor are introduced with the evaluated conditions. The hidden reference establishes the maximum quality of the evaluation, as it is identical to the reference. The hidden anchor is usually a low-pass version of the reference, filtered at 3.5 kHz, and works as a low quality reference in order to consider the results measured on an absolute scale. This test provides highly reliable results with a small number of listeners.

To evaluate the algorithms implemented in this thesis, twelve items have been used, containing male and female voices in different languages and combining clean and mixed speech, which are shown in table 3. The test has been carried out at two different bitrates, 32 and 16.4 kbps, and 8 expert listeners have evaluated the quality of the samples. The representation of the results is done using Student's T distribution and a confidence interval of 95%.

As the goal of the thesis is to evaluate the quality of the transformation algorithms with respect to the ACELP implementation, the samples have also been processed forcing the EVS to work in ACELP mode. The last condition is formed by samples generated using the EVS in TCX mode, considering that the algorithms of this thesis are also based on the transformation of the signal.

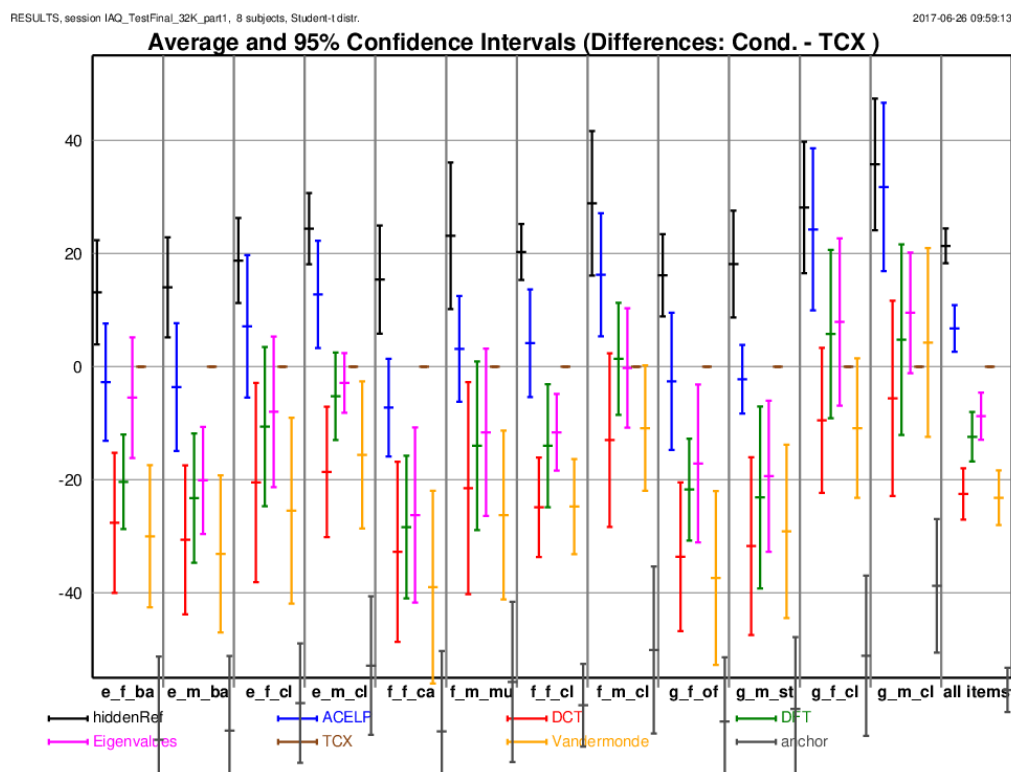


Figure 49: MUSHRA results represented with respect to the EVS-TCX encoding. 32kbps encoding using 8 expert listeners

Figure 48 shows the average results of the test for a 32 kbps bitrate. The order

of the algorithms in terms of their quality is similar to the POLQA results. However, the quality of the DCT transformation is lower than it was predicted by POLQA. Eigenvalue decomposition and DFT are at the same level as TCX in clean speech, but ACELP clearly outperforms all the other algorithms.

The performance of TCX is improved in noisy signals. This can be clearly seen in figure 49, which shows a differential representation of the MUSHRA results with respect to the TCX encoding. Conversely, ACELP is way over the TCX in clean signals and the thesis algorithms are at its same level. The quality of TCX grows in noisy signals reaching the level of ACELP and in some cases outperforming it, leaving the other transformation algorithms behind.

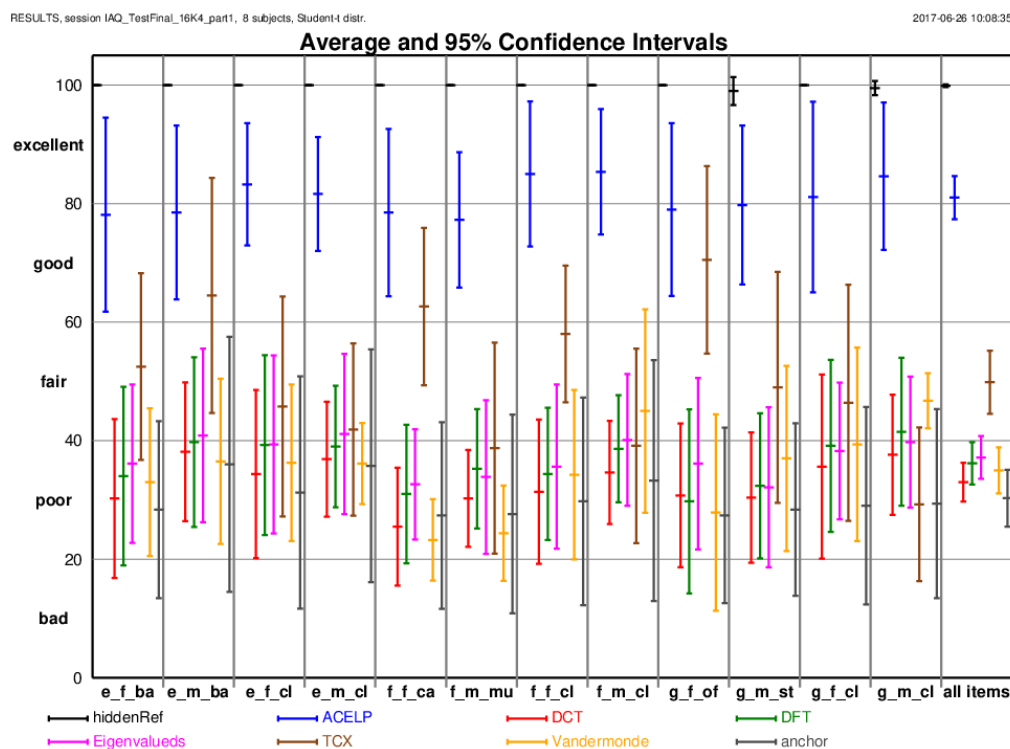


Figure 50: Average MUSHRA results for 16kbps encoding using 8 expert listeners

The same test was performed using encoded signals at 16 kbps. The results are shown in figure 50. Notice that, in this case, the quality of the signals has suffered a significant loss.

ACELP still outperforms all the other algorithms, which is consistent with the results obtained in the POLQA test where the degradation of the ACELP encoder was not as steep as the other algorithms. However, considering the low bitrate, MUSHRA may not be the optimum test to compare the implemented algorithms.

As can be seen in figure 50, the transformation algorithms are usually below the hidden anchor. Considering that this thesis focuses only on speech signals, most of the signal's information will be found in frequencies under 4 kHz. This means that a low-pass filter at 3.5 kHz will not add a significant distortion to the signal.

Therefore, a LP filter does not affect the signal as much as the artefacts produced by the encoder. That is the reason why the hidden anchor presents a better quality making this test inappropriate for the evaluation of low bitrate speech signals.

It can be seen in the test results that the quality of the implemented algorithms is not so different from one to the next. However, considering the reason mentioned above, this measurement may not be reliable.

6 Discussion and Future Lines of Study

With regards to the goal of this thesis, four transformation algorithms have been implemented in order to decorrelate the innovative target vector, which is calculated by subtracting the pitch information from the LPC residual. The first two implemented are the DFT and DCT. These transformations are commonly used in signal processing and their behaviour is independent of the signal analysed. Next, the eigenvalue decomposition was implemented. This is a well known algebraic operation used in dimensionality reduction. Finally, the Vandermonde decomposition was implemented as explained in [22]. Both eigenvalue and Vandermonde decomposition need to calculate the transformation matrix at every processed frame, adapting to the signal in order to decorrelate its samples.

The transformation algorithms were integrated in the EVS codec structure, more specifically replacing the innovative codebook search in ACELP mode with the algorithms explained in section 4.5 and 4.6. In order to insert the new encoding structure, some changes were required, which are explained along section 4.

The objective of the thesis was to analyse alternatives to the ACELP implementation, whose complexity grows too high when the size of the codebook is large. The transformation algorithms focus on reducing the correlation between the samples of the target in order to apply simpler codebook search algorithms avoiding the efficiency loss introduced by this correlation. In order to evaluate the quality of the implemented algorithms, several objective and subjective analysis were carried out.

Firstly, an entropy analysis was carried out, evaluating the different algorithms depending on the degree of decorrelation that they can achieve on the signal. As expected, eigenvalue decomposition perfectly diagonalises the autocorrelation matrix. DCT manages to get a significantly high decorrelation of the signal, even though it is not perfect. As explained in section 3.5.1, DFT is not meant to decorrelate the samples of the signal, which can be observed in the results. Vandermonde decomposition, theoretically, performs a perfect decorrelation of the samples in the target; however, the autocorrelation result looks more similar to the DFT representation than to a diagonal matrix. This may be caused, as stated in [30], by a numerical instability present when the Vandermonde decomposition is calculated using a transformation length equal or higher than 64 samples, which is the length required in this thesis work.

ACELP algorithms need to make a compromise between the quality of the encoding and the complexity of the algorithm. The purpose of the transformations is to allow for the use of simpler encoding algorithms in order to reduce the complexity of this process. For this reason, the complexity of the implemented algorithms will be evaluated with respect to the ACELP encoder.

Note that the algorithms implemented are not optimised and just taking two simple steps, as explained in section 5.1, it was possible to obtain a significant reduction of the complexity. The final results are at the same level as the ACELP for DFT and DCT, which means that with some optimisation it would be possible to achieve a lower complexity than the actual ACELP implementation. Eigenvalue decomposition, as expected, is far above the rest in terms of complexity, considering

that it needs very demanding algorithms to perform the decomposition. Vandermonde obtains an intermediate value between eigenvalue decomposition and DFT/DCT, as it still needs to calculate the transformation matrix for every frame and apply the transformation performing a matrix multiplication.

POLQA and MUSRA tests were carried out in order to evaluate the objective and subjective quality of the processed samples. It was noticed that at low bitrates, the quality of the transformation algorithms decreases so much that they are not viable compared to the ACELP implementation. Working at higher bitrates (32 kbps), the performance of the algorithms is significantly increased. Their quality remains under the ACELP implementation but there is not such a high difference compared to the 16.4 kbps version.

The results of the POLQA test agree with the evaluation of the entropy. As expected, eigenvalue decomposition performs the best, followed by DCT, DFT and Vandermonde. The reason for Vandermonde to be in the last position is, as explained before, the numerical instability of the decomposition process. If this problem could be solved, the quality of the Vandermonde decomposition should be at the same level as the eigenvalue decomposition.

The MUSHRA test shows, however, some different results. In this case, the eigenvalue decomposition, DFT and Vandermonde decomposition maintain the same order as in POLQA. The quality of DCT, however, has dropped to the last position. This is an unexpected result, as DCT has been proven to perform better than DFT, which is supported by the entropy analysis.

As a conclusion, four algorithms were implemented in order to transform the innovative target vector, reducing the correlation between its samples. The algorithms were integrated in the ACELP structure present in the EVS codec allowing the use of an arithmetic encoder, simpler than the encoder used in ACELP. The efficient transformation algorithms, which are DFT and DCT, have a great performance in terms of complexity, which could be reduced considering that the encoder is not optimized. The quality of the samples has an acceptable level and could be improved after the various problems commented before are solved.

Future lines of study could focus on DCT and Vandermonde decomposition, as they are promising transformations in this field considering the fact that they are not optimised in this thesis. The quality of the encoding of the DCT could be improved by finding out what is wrong in the encoding process that makes it perform worse than DFT in the listening tests.

Some improvements can also be done in the Vandermonde decomposition. Firstly, finding a numerically stable algorithm to calculate the transformation matrices in order to raise the quality of the encoding, as the theoretical complete decorrelation of the samples would be obtained. The other problem that Vandermonde decomposition faces is its complexity. More efficient algorithms could be researched in order to reduce the computational requirements of finding the transformation matrix and applying it.

Finally, as it could be noticed in section 5.1, the envelope based arithmetic encoder could be optimised in order to make it more efficient and reduce the complexity of the whole system.

References

- [1] Tom Bäckström and Christian R Helmrich. Arithmetic coding of speech and audio spectra using *tcx* based on linear predictive spectral envelopes. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5127–5131. IEEE, 2015.
- [2] J-P Adoul, P Mabillean, M Delprat, and S Morissette. Fast celp coding based on algebraic codes. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'87.*, volume 12, pages 1957–1960. IEEE, 1987.
- [3] Yi Zhao, Sheng Zhang, and Xiaokang Lin. Two methods of design and implementation of acelp vocoder. In *Signal Processing, Communication and Computing (ICSPCC), 2013 IEEE International Conference on*, pages 1–6. IEEE, 2013.
- [4] Isabel M Trancoso and Bishnu S Atal. Efficient search procedures for selecting the optimum innovation in stochastic coders. *IEEE transactions on acoustics, speech, and signal processing*, 38(3):385–396, 1990.
- [5] Tom Bäckström and Christian R Helmrich. Decorrelated innovative codebooks for acelp using factorization of autocorrelation matrix. In *INTERSPEECH*, pages 2794–2798, 2014.
- [6] Takehiro Moriya and Masaaki Honda. Transform coding of speech using a weighted vector quantizer. *IEEE journal on selected areas in communications*, 6(2):425–431, 1988.
- [7] Stefan Bruhn, Harald Pobloth, Markus Schnell, Bernhard Grill, Jon Gibbs, Lei Miao, Kari Järvinen, Lasse Laaksonen, Noboru Harada, Nobuhiko Naka, et al. Standardization of the new 3gpp evs codec. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5703–5707. IEEE, 2015.
- [8] Martin Dietz, Markus Multrus, Vaclav Eksler, Vladimir Malenovsky, Erik Norvell, Harald Pobloth, Lei Miao, Zhe Wang, Lasse Laaksonen, Adriana Vasilache, et al. Overview of the evs codec architecture. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5698–5702. IEEE, 2015.
- [9] Bruno Bessette, Redwan Salami, Roch Lefebvre, Milan Jelinek, Jani Rotola-Pukkila, Janne Vainio, Hannu Mikkola, and Kari Jarvinen. The adaptive multirate wideband speech codec (amr-wb). *IEEE transactions on speech and audio processing*, 10(8):620–636, 2002.
- [10] Nils Werner and Bernd Edler. Nonuniform orthogonal filterbanks based on mdct analysis/synthesis and time-domain aliasing reduction. *IEEE Signal Processing Letters*, 24(5):589–593, 2017.

- [11] Srikanth Nagisetty, Zongxian Liu, Takuya Kawashima, Hiroyuki Ehara, Xuan Zhou, Bin Wang, Zexin Liu, Lei Miao, Jon Gibbs, Lasse Laaksonen, et al. Low bit rate high-quality mdct audio coding of the 3gpp evs standard. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5883–5887. IEEE, 2015.
- [12] G Recommendation. 719:“low-complexity, full-band audio coding for high-quality, conversational applications”, 2008.
- [13] Guillaume Fuchs, Christian R Helmrich, Goran Marković, Matthias Neusinger, Emmanuel Ravelli, and Takehiro Moriya. Low delay lpc and mdct-based audio coding in the evs codec. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5723–5727. IEEE, 2015.
- [14] T Moriya and H Suda. An 8 kbit/s transform coder for noisy channels. In *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, pages 196–199. IEEE, 1989.
- [15] Nathan Marcuvitz. *Waveguide handbook*. Number 21. Iet, 1951.
- [16] PP Vaidyanathan. The theory of linear prediction. *Synthesis lectures on signal processing*, 2(1):1–184, 2007.
- [17] Cheng-Yu Yeh. Algebraic codebook search strategy for an algebraic code-excited linear-prediction speech coder by means of reduced candidate mechanism and iteration-free pulse replacement. *IET Signal Processing*, 8(9):990–995, 2014.
- [18] Victoria E Sanchez and J-P Adoul. Low-delay wideband speech coding using a new frequency domain approach. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 2, pages 415–418. IEEE, 1993.
- [19] Victoria Sánchez, Antonio J Rubio, Juan M López-Soler, Ángel de la Torre, and Antonio M Peinado. A low-delay transform domain approach to trellis coded quantization. *Speech communication*, 21(3):141–153, 1997.
- [20] Sean A Ramprashad. A multimode transform predictive coder (mtpc) for speech and audio. In *Speech Coding Proceedings, 1999 IEEE Workshop on*, pages 10–12. IEEE, 1999.
- [21] Juin-Hwey Chen and Dongmei Wang. Transform predictive coding of wideband speech signals. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 275–278. IEEE, 1996.
- [22] Tom Backstrom. Vandermonde factorization of toeplitz matrices and applications in filtering and warping. *IEEE Transactions on Signal Processing*, 61(24):6257–6263, 2013.

- [23] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.
- [24] Sven Feldmann and Georg Heinig. Vandermonde factorization and canonical representations of block hankel matrices. *Linear algebra and its applications*, 241:247–278, 1996.
- [25] Daniel L Boley, Franklin T Luk, and David Vandevoorde. Vandermonde factorization of a hankel matrix. *Scientific computing (Hong Kong, 1997)*, pages 27–39, 1998.
- [26] Christian Fischer Pedersen and Tom Bäckström. Sparse time-frequency representation of speech by the vandermonde transform. In *INTERSPEECH*, pages 2248–2252, 2014.
- [27] 3GPP Specification TR 26.445. Codec for enhanced voice services (evs); detailed algorithmic description. v 14.0. 2017.
- [28] Lapack linear algebra package. <http://www.netlib.org/lapack/>. Accessed: 06/03/2017.
- [29] 3GPP Specification TR 26.443. Codec for enhanced voice services (evs); ansi c code (floating-point). v 14.0. 2017.
- [30] Tom Bäckström, Johannes Fischer, and Daniel Boley. Implementation and evaluation of the vandermonde transform. In *Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European*, pages 71–75. IEEE, 2014.
- [31] Ake Björck and Victor Pereyra. Solution of vandermonde systems of equations. *Mathematics of Computation*, 24(112):893–903, 1970.
- [32] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2014.
- [33] C Laflamme, J-P Adoul, HY Su, and S Morissette. On reducing computational complexity of codebook search in celp coder through the use of algebraic codes. In *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pages 177–180. IEEE, 1990.
- [34] Ebu sqam. <https://tech.ebu.ch/publications/sqamcd>. Accessed: 15/04/2017.
- [35] ITU-T. Software tool library: User’s manual, 2009.
- [36] Tom Bäckström. *Speech Coding: Code Excited Linear Prediction*. Springer, 2017.
- [37] ITU-T P. 863. Perceptual objective listening quality assessment. *Int Telecom Union*, 2011.

- [38] Yi Gaoxiong and Zhang Wei. The perceptual objective listening quality assessment algorithm in telecommunication: Introduction of itu-t new metrics polqa. In *Communications in China (ICCC), 2012 1st IEEE International Conference on*, pages 351–355. IEEE, 2012.
- [39] John G Beerends, Christian Schmidmer, Jens Berger, Matthias Obermann, Raphael Ullmann, Joachim Pomy, and Michael Keyhl. Perceptual objective listening quality assessment (polqa), the third generation itu-t standard for end-to-end speech quality measurement part i—temporal alignment. *Journal of the Audio Engineering Society*, 61(6):366–384, 2013.
- [40] ITUR Recommendation. Bs. 1534-1. method for the subjective assessment of intermediate sound quality (mushra). *International Telecommunications Union, Geneva*, 2001.
- [41] Guide for lapack in windows. <http://icl.cs.utk.edu/lapack-for-windows/clapack/index.html>. Accessed: 06/04/2017.

Appendix 1: Building and integration of LAPACK library

In order to implement the eigenvalue decomposition in C, the use of external libraries is required. In this case, the Linear Algebra Package (LAPACK) has been used. This library provides a set of linear algebra operations implemented in Fortran, performed by calls to the Basic Linear Algebra Subprograms (BLAS) library.

BLAS contains a group of low level subroutines to perform linear algebra operations such as vector additions, scalar multiplication or matrix multiplication. Considering that this library is implemented at a very low level, there are multiple implementations depending system's architecture in which the program will run. This makes them very efficient and provide a very fast performance on these operations. However, in order to adapt to any system, the version of BLAS used in this thesis is untuned and, therefore, will be considerably slower than a specific version.

As commented before, LAPACK is programmed in Fortran but it provides an interface that allows its usage in C programs. These libraries include a guide [41] in order to integrate the libraries into Visual Studio; however, some problems were faced while integrating the libraries into the EVS project. For this reason, this appendix explains the steps followed to build and use the LAPACK library inside the EVS Visual Studio solution.

[41] shows multiple ways to build the libraries depending on the version of LAPACK used and its application. It even provides a precompiled set of libraries for Visual Studio 2015, which, however, will not be used in this case due to the project configuration.

In this case, CMake will be used to build the libraries. It is necessary to have a Fortran compiler as well as GCC installed in the computer. In this case, the Minimalist GNU for Windows (MinGW) is used, as it contains GCC as C compiler and includes a Fortran compiler.

Once using CMake, the input directory, containing the source code, and the output directory where the project will be placed have to be set. Clicking in "Configure", MinGW has to be set as generator of the project and the option "Specify native compilers" has to be chosen. Then, it is necessary to select the corresponding C and Fortran compilers, specifying the path inside the MinGW directory to "gcc.exe" and "gfortran.exe" respectively.

Once the configuration has finished and the options for the project have been set, it is only necessary to press "Generate". This will create an executable file that is used build the libraries.

In our case, some source files gave trouble in the building stage due to some undefined data structures. This structures are used to represent complex numbers in the routines that process complex data. This project only works with real signals, so the declaration of those data structures can be removed from the corresponding files (chetri.c, csytri.c, zhetri.c, zsytri.c) and the compilation will run.

Finally, the compilation process may not generate the specific .dll files linked to the corresponding .lib. However, as .dll files contain only the linking information to

the actual libraries `.lib`, it is possible to take the `.dll` files provided by the precompiled files mentioned before. These DLL files have to be inserted in the same directory as the executable file generated by the project solution and the `.lib` files have to be added to the libraries dependencies of the project. [41] also mentions a set of environment variables that need to be set in the project solution.

Once the libraries are built and included in the project, the four header files in the source code have to be included with all the header files in the project.

These are the steps followed to integrate the LAPACK library into the EVS project. After that, it is only necessary to follow the library's documentation [28] in order to use its routines.

Appendix 2: Weights for the operations in the WMOPS estimation

As explained in section 5.1, The complexity analysis of every algorithm is performed in terms of WMOPS [35]. For this method, it is necessary to count the operations performed inside an algorithm and evaluate them depending on their computational load with respect to a basic operation. Every operation is weighted according to this idea and the total number of operations is added in order to obtain the total complexity measured in WMOPS.

This appendix includes a table (8) with the main operations, their weights and examples of what they mean as they are explained in [35].

Operation	Example	Weight
Addition	$a = b + c$	1
Multiplication	$a = b * c$	1
Multiplication + addition	$a+ = b * c$	1
Move	$a = b$	1
Store in array	$a[i] = b[i] + c[i]$	1
Logical	AND, OR, etc.	1
Shift	$a = b >> c$	1
Branch	if, if...else	4
Division	$a = b/c$	18
Square-root	$a = \text{sqrt}(b)$	10
Transcendental	sine, arctan, etc.	25
Function call	$a = \text{func}(b, c, d)$	2 + number of arguments passed and returned
Loop initialization	for(i=0;i<n;i++)	3
Indirect addressing	$a = b.c$	2
Pointer initialization	$a[i]$	1
Exponential	pow, e^n	25
Logarithm	log	25
Conditional test	used in conjunction with BRANCH	2

Table 8: Weights for the different operations analysed in the complexity study