Implementation of FAPEC decorrelation stages for IQ and water column data

Degree Thesis
In partial fulfilment of the requirements for the degree in
Telecommunications Engineering

Author
Aniol Martí Espelt

Advisors
Ferran de Cabrera Estanyol
Jordi Portell i de Mora
Jaume Riba Sagarra

Barcelona, June 2021

Abstract

Recent technological advances have resulted in massive data collection, also in hazard environments such as the sea or space. In these situations, both computing power and downlink bandwidth are quite limited. For this reason, efficient data compression becomes a basic technology.

In this thesis we propose two new stages for the FAPEC data compressor: the first one is specially designed to decorrelate data from Radio Frequency (RF) signals; the second one is intended to preprocess multibeam water column data collected by Kongsberg Maritime echosounders.

In order to evaluate the algorithms we have developed, we propose using negentropy, a measure invariant to linear maps which gives us the distance from our data to gaussianity. Finally, we compare the presented stages with other well-known compression algorithms such as FLAC (audio compression) and GZIP (general purpose compression).

Resum

Els avenços tecnològics dels darrers anys han portat a recollir grans quantitats de dades, també en entorns extrems com el mar o l'espai. En aquests ambients, tant la capacitat dels ordinadors de bord com l'amplada de banda del canal de comunicacions sol ser molt limitat. És per això que la compressió eficient de dades resulta ser fonamental.

En aquest treball proposem dues noves etapes de preprocessat pel compressor FAPEC: la primera està dissenyada per decorrelar dades provinents de senyals de radiofreqüència (RF); la segona per dades de columna d'aigua recollides per ecosondes submarines de Kongsberg Maritime.

Per tal d'avaluar el comportament dels algoritmes desenvolupats, proposem utilitzar la neguentropia, una mesura invariant a aplicacions lineals que ens aporta la distància d'un conjunt de dades respecte a la gaussianitat. Finalment, comparem les etapes amb altres algoritmes coneguts com el FLAC (compressor d'àudio) o el GZIP (compressor de propòsit general).

Agraïments

Ai, els agraïments. Els agraïments són, amb diferència, la part més difícil de tot el treball. Amb això no pretenc pas treure valor a aquest treball, només faltaria, però els agraïments els llegeix tothom: tant qui busca el contingut del treball com qui només vol xafardejar què has fet. Per això són importants, i també perquè no pots deixar-te ningú. Tanmateix, on traces la línia? En Jaime, que em va servir una cervesa quan estava fart d'escriure, ha d'anar als agraïments? No ho sé, però decidir això és molt feixuc, i després de quatre mesos de treballar, estudiar i fer aquest treball doncs el meu electroencefalograma és paral·lel a l'eix de les abscisses. Així que, a continuació, donaré les gràcies a les persones que han estat més a prop i que he empipat més aquests mesos.

Primer de tot vull donar les gràcies als meus tres directors: en Ferran de Cabrera, en Jordi Portell i en Jaume Riba. A en Jordi per donar-me la possibilitat de fer aquest treball, a en Jaume i a en Ferran per totes les observacions i aportacions teòriques que han fet, i a tots tres per les revisions, reunions i dubtes que m'han resolt. Gràcies.

A part d'ells, també mereixen ser en aquests agraïments en Marc Vilà, que ha assistit a moltes de les reunions; en David Amblàs, sense el qual molts resultats no haurien estat possibles i l'Orestes Mas, que m'ha ajudat amb molts detalls del LaTeX i el disseny del document.

Deixant de banda l'àmbit acadèmic i professional he de donar les gràcies a molts amics que han estat allà per fer una cervesa, jugar al Counter Strike, anar al cine o per, simplement, parlar i desconnectar.

I ara, Jaime, posa'm una altra gerra.

Contents

1 Introduction **1**
1.1 Motivation and goals . . . 1
1.2 Thesis outline and organization . . . 1
1.3 State of the art . . . 2
1.4 Project planning and costs . . . 2
1.4.1 Gantt diagram . . . 2
1.4.2 Budget and costs . . . 2

2 Entropy coding **3**
2.1 Basic concepts on information theory . . . 3
2.1.1 Shannon entropy . . . 3
2.1.2 Differential entropy . . . 4
2.1.3 Negentropy . . . 4
2.2 Coding techniques . . . 4
2.2.1 Golomb coding . . . 5

3 The FAPEC data compressor **6**
3.1 Background . . . 6
3.2 General overview . . . 6
3.3 Entropy coder . . . 7
3.4 Current FAPEC implementation . . . 8

4 Development of new stages **9**
4.1 Requirements for FAPEC stages . . . 9
4.2 Evaluation of stages . . . 10
4.2.1 Information metrics . . . 10
4.2.2 Performance metrics . . . 11

5 Wave stage **12**
5.1 Introduction to IQ data . . . 13
5.2 Motivation and applications of the Wave stage . . . 13
5.3 Design . . . 14
5.3.1 Requirements and specifications . . . 14
5.3.2 Algorithm design . . . 14
5.4 Results . . . 17

6 KMALL stage **19**
6.1 The KMALL data format . . . 20
6.2 Design . . . 22
6.2.1 Data analysis . . . 22
6.2.2 Algorithm design . . . 23
6.3 Results . . . 26

7 Conclusions and future work **28**

Bibliography **30**

A Proof of Equation 2.5 **32**

B Additional Wave histograms **33**

C Additional KMALL histograms **34**

Acronyms **35**

List of Figures

1.1 Project Gantt diagram. 2

3.1 The FAPEC data compressor structure. 7

4.1 Histograms of a file compressed with Wave. 10

5.1 Illustration of radio occultation (RO). 12
5.2 Flowchart of the Wave preprocessing stage. 16
5.3 Histograms of *Formigues* by Manel, showing the range reduction. . . 17
5.4 Comparison of compressors which shows that FAPEC is the fastest. . 18

6.1 Conception of multibeam sonar on NOAA Ship NANCY FOSTER. . 19
6.2 MWC datagram as described by Kongsberg (410224 Revision H). . . 22
6.3 Raw representation of MWC amplitude from a EM 2040 echosounder. 23
6.4 Raw representation of MWC phase from a EM 2040 echosounder. . . 24
6.5 Flowchart of the KMALL preprocessing stage. 25
6.6 Histograms showing the range reduction of MWC data from a EM 304. 26
6.7 Comparison of compressors showing that FAPEC performs better both in time and ratio than GZIP. 27

B.1 Histograms of *Els entusiasmats* by Manel. 33
B.2 Histograms of *Time* by Pink Floyd. 33
B.3 Histograms *Money* by Pink Floyd. 33

C.1 Histograms of EM 712 retrieved MWC data. 34
C.2 Histograms of EM 2040 retrieved MWC data (with phase). 34
C.3 Histograms of EM 2040 retrieved MWC data. 34

List of Tables

2.1 Rice codes of integers between 0 and 5 and suffix from 0 to 3 bits. . . 5

5.1 Negentropies of audio files. It shows the performance of the Wave stage. 18
5.2 Euclidean distances from Figure 5.4. 18

6.1 Compatibility datagrams of the KMALL format. 20
6.2 Installation and runtime datagrams of the KMALL format. 21
6.3 External sensor output datagrams of the KMALL format. 21
6.4 File datagrams of the KMALL format. 21
6.5 Multibeam datagrams of the KMALL format. 21
6.6 Negentropies before and after preprocessing of files holding MWC data. 26
6.7 Euclidean distances from Figure 6.7. 27

Chapter 1
Introduction

1.1 Motivation and goals

Data compression plays a key role in remote sensing, specially in hazardous environments, where computing power and data links are limited. Two examples of these environments are deep space and offshore. In addition, some technologies used to capture data are proprietary and the data structure may not be optimal for general purpose compressors such as GZIP.

This thesis is part of two strategic projects of DAPCOM Data Services, a spin-off company of the UPC and UB. The first project is intended to compress Radio Frequency (RF) data, mainly from SDR devices; the second one is focused on compressing the water column information contained in the KMALL format, a new data structure from the multinational Kongsberg Maritime.

We aim to develop two FAPEC stages to perform better, in terms of compression ratio and speed, than already existing algorithms. Specifically, we will compare the first stage with FLAC, a well-known audio encoder, and GZIP. The second stage will be only compared with GZIP, as we have not been able to find any specific compressor for such format. Additionally, we also want to propose a universal metric to reliably quantify the performance of new stages in terms of information redundancy.

1.2 Thesis outline and organization

This thesis starts in chapter 2 with a theoretical approach to information theory. There, we describe some basic concepts which will be used later, such as differential entropy or Golomb coding.

In chapter 3 we describe the key aspects of the FAPEC data compression framework and the current features and stages. Then, in chapter 4 we state the general requirements and specifications for the new stages. In this chapter we also propose some metrics to evaluate preprocessing stages.

The next two chapters correspond to the two preprocessing stages we have developed. The outline for both chapters 5 and 6 is almost the same: first, we introduce the data format. Then, we propose an algorithm, and finally, we evaluate it using the metrics from chapter 4.

Finally, the last chapter consists in concluding the thesis and describing some possible future lines to work with.

The LaTeX source code and the Python scripts used to evaluate the stages can be found in the GitHub repository: `https://github.com/aniolm9/bsc-thesis-fapec`.

1.3 State of the art

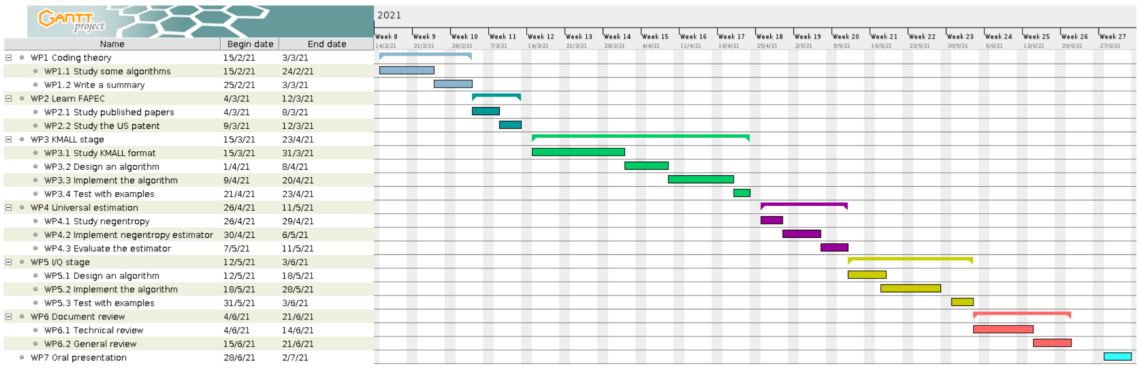
Compressing RF data using FLAC is not new. For instance, there is a paper [NA13] where FLAC is applied on IQ data. This paper together with reports from some DAPCOM clients may serve as a basis to develop a new algorithm.

On the other hand, the KMALL stage has been mainly based on the stage that DAPCOM developed in 2019 for the old water column data structure [Por+19]. As the KMALL format was released in 2019, very few literature exists, but we have found a paper [Mos+13] where the authors propose a modification of Huffman coding to compress MWC data in older formats. Although the entropy analysis they do is of much interest, the proposed algorithm is mostly based on plain entropy coding, so it is not very useful in our case.

Finally, as regards to the universal metric to evaluate stages, we have found some interesting papers [Com94] [HO00] where the authors explore ways to measure the distance of some data to gaussianity.

1.4 Project planning and costs

1.4.1 Gantt diagram



Name	Begin date	End date
WP1 Coding theory	15/2/21	3/3/21
WP1.1 Study some algorithms	15/2/21	24/2/21
WP1.2 Write a summary	25/2/21	3/3/21
WP2 Learn FAPEC	4/3/21	12/3/21
WP2.1 Study published papers	4/3/21	8/3/21
WP2.2 Study the US patent	9/3/21	12/3/21
WP3 KMALL stage	15/3/21	23/4/21
WP3.1 Study KMALL format	15/3/21	31/3/21
WP3.2 Design an algorithm	1/4/21	8/4/21
WP3.3 Implement the algorithm	9/4/21	20/4/21
WP3.4 Test with examples	21/4/21	23/4/21
WP4 Universal estimation	26/4/21	11/5/21
WP4.1 Study negentropy	26/4/21	29/4/21
WP4.2 Implement negentropy estimator	30/4/21	6/5/21
WP4.3 Evaluate the estimator	7/5/21	11/5/21
WP5 I/Q stage	12/5/21	3/6/21
WP5.1 Design an algorithm	12/5/21	18/5/21
WP5.2 Implement the algorithm	18/5/21	28/5/21
WP5.3 Test with examples	31/5/21	3/6/21
WP6 Document review	4/6/21	21/6/21
WP6.1 Technical review	4/6/21	14/6/21
WP6.2 General review	15/6/21	21/6/21
WP7 Oral presentation	28/6/21	2/7/21

Figure 1.1: Project Gantt diagram.

1.4.2 Budget and costs

This thesis consists in the design, implementation and evaluation of two stages which will be included in a bigger framework, FAPEC. In this situation, a straightforward financial study is not possible and going further is clearly out of scope.

However, this thesis has been developed within a cooperation agreement between the Polytechnic University of Catalonia (UPC) and DAPCOM Data Services, with a cost of 3816€.

Chapter 2
Entropy coding

Entropy coding is a lossless data compression scheme based on symbols probability. This concept was first described by Claude E. Shannon in 1948 in his paper *A Mathematical Theory of Communication* [Sha48].

Giving an extensive explanation of entropy coding would be too lengthy and outside the scope. Therefore, in this chapter we will just define the most basic equations in information theory and we will also describe some well-known coding algorithms that will be relevant later. The interested reader should see reference [CT06].

2.1 Basic concepts on information theory

2.1.1 Shannon entropy

Given a discrete random variable X with probability function $p(x)$, its Shannon entropy is defined as:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \cdot \log_2 p(x_i). \tag{2.1}$$

This result represents the average level of information (or uncertainty) of the random variable X, expressed in *bits*. Sometimes entropy is defined using the natural logarithm, therefore expressed in *nats*.

Shannon's source coding theorem

This theorem establishes the theoretical limits to lossless data compression, and a different meaning to the Shannon entropy defined above. Formally:

Theorem 2.1 *Let L_n^* be the expected codeword length per symbol of an optimal n-th order lossless data compression code (in bits/symbol). Let $(X_1, X_2, \dots, X_n)$ be a sequence of symbols from a stochastic process X. Then,*

$$\frac{H(X_1, X_2, \dots, X_n)}{n} \leq L_n^* < \frac{H(X_1, X_2, \dots, X_n)}{n} + \frac{1}{n}. \tag{2.2}$$

If X is a stationary stochastic process,

$$\lim_{n\to\infty} L_n^* = H(X). \tag{2.3}$$

The previous theorem reveals a new interpretation of Shannon entropy: it is the average number of bits per symbol required to encode it.

2.1.2 Differential entropy

Given a continuous random variable X with a density $f(x)$ supported in S and a cumulative distribution $F(x)$, its differential entropy (in *nats*) is defined as:

$$h(X) = -\int_S f(x) \ln f(x) dx. \tag{2.4}$$

In some cases computing an explicit probability density function is too hard or even impossible. In these cases, an equivalent definition in terms of the quantile function $Q(p) = F^{-1}(p)$ can be used (see appendix A):

$$h(X) = \int_0^1 \ln Q'(p) dp. \tag{2.5}$$

This equation is specially useful in some entropy estimation techniques [Vas76].

2.1.3 Negentropy

Negentropy measures the difference in entropy between a given distribution and the normal distribution with same variance. Thus, negentropy provides an indicator of normality and it is invariant by any linear invertible change of coordinates [Com94].

Let $G \sim \mathcal{N}(\mu, \sigma^2)$. Let X be a continuous random variable with variance σ^2. Then, negentropy is defined as:

$$J(X) = h(G) - h(X) \geq 0. \tag{2.6}$$

A normal distribution is used because, among all the real-valued distributions supported on $\mathbb{R}$ with variance σ^2, $\mathcal{N}(\mu, \sigma^2)$ has maximum entropy:

$$h(G) = \frac{1}{2} \ln\left(2\sigma^2\pi\right) + \frac{1}{2} = \ln\left(\sqrt{2\pi e\sigma^2}\right). \tag{2.7}$$

2.2 Coding techniques

There exist several techniques to assign bits to symbols. In general, one would be interested in a coding scheme which gives codewords with a mean length as close as possible to the Shannon limit. However, sometimes achieving the optimality might be too demanding, so other faster but suboptimal techniques have been developed.

In this section we will describe Golomb coding, a known technique used by several standards such as FLAC [Coa01] or CCSDS 121.0 [Spa20]. Besides, the FAPEC entropy coder [Por+18] is highly inspired in Rice codes [RP71], a subset of the Golomb codes [Gol66]. For this reason, we will only focus on Golomb coding.

Although we are only interested in Golomb codes, it is worth to mention that there exist other entropy coding techniques which are more common than Golomb, for instance Huffman coding [CT06], arithmetic coding [Mac02] or, more recently, the Asymmetric Numeral System [Dud+15].

2.2.1 Golomb coding

Golomb codes were first proposed by Solomon W. Golomb in 1966 in his article *Run-Length Encodings* [Gol66]. In this paper, Golomb proposes these codes as a way to encode events which follow a geometric (or exponential) distribution. For these distributions, Golomb codes are optimal [YRM91] [GV75] and fast to calculate. Simplifying, Golomb coding is highly suitable for situations where small values have a bigger probability than large values.

Encoding procedure

Let N be the number to encode. Then, its Golomb code is:

Algorithm 2.1: Golomb encoding procedure

```
Input: $N$
Define: $M, q, r \in \mathbb{N}, \quad k = \lceil \log_2(M) \rceil$
$q \leftarrow \lfloor \frac{N}{M} \rfloor$
$r \leftarrow N - M \cdot q$
code $q$
    write a $q$-length string of 1 (or 0) bits.
    write a 0 (or 1) bit.
end
code $r$
    if $r < 2^k - M$ then
        binary code $r$ using $k - 1$ bits.
    else
        binary code $r + 2^k - M$ using $k$ bits.
    end
end
Output: Golomb code for $N$.
```

For the particular case $M = 2^k, \;\; k \in \mathbb{N}$ the codes are called Rice codes.

Example

In Table 2.1 we provide an example of some Rice codes calculated following the previous algorithm. Notice that the suffix of Rice codes always has length k.

N	$k = 0$	$k = 1$	$k = 2$	$k = 3$
0	1	1 0	1 00	1 000
1	01	1 1	1 01	1 001
2	001	01 0	1 10	1 010
3	0001	01 1	1 11	1 011
4	00001	001 0	01 00	1 100
5	000001	001 1	01 01	1 101

Table 2.1: Rice codes of integers between 0 and 5 and suffix from 0 to 3 bits.

Chapter 3
The FAPEC data compressor

Fully Adaptive Prediction Error Coder (FAPEC) is a versatile and efficient data compressor originally designed for space missions [Por+18]. The main advantages of FAPEC are its high computing performance and its resilience in front of noise or data outliers.

In this chapter we will explore the structure of FAPEC and also its integration with several decorrelation stages such as the two presented in this work. In order to do so, we will first see the origin of FAPEC and its need in space industry.

3.1 Background

Space missions are continuously evolving, hence the amount of data generated is growing. However, onboard computing power and downlink bandwidth are not evolving so fast and they are still quite narrowed. The usual technique to deal with the second limitation is data compression, but it has to be implemented efficiently as we are still limited by the first constraint. Moreover, the chosen algorithm must be reliable, as losing data in remote sensing could have severe consequences.

The Consultative Committee for Space Data Systems (CCSDS) already provides some standards for space data compression. For instance, CCSDS 121.0 for generic lossless data compression or CCSDS 122.0 [Spa17] for image data compression. Although they achieve good compression ratios and their overall performance is remarkable, they can be too complex and demanding for some missions with limited hardware, such as cubesats. Besides, having a unique solution for data compression would be better than having different standards for different data types.

In order to work out the previous limitations, FAPEC was proposed, featuring:

- High computing performance.
- Good coding efficiency for all entropy levels regardless of the statistical outliers.
- Several preprocessing stages to suit different types of data.

3.2 General overview

The FAPEC data compressor is based on two stages: a preprocessing stage followed by the entropy coder. In fact, FAPEC's name comes from this architecture as usually the first stage is some type of predictor which tries to estimate the true samples and generates a prediction error. Then, the error is sent to the entropy coder instead of the original samples (thereof the Prediction Error Coder naming).

Formally, given an input sample x_i and an estimator $\hat{x}_i$, the value sent to the entropy coder is:

$$e_i = x_i - \hat{x}_i. \tag{3.1}$$

Restoring the original value in decompression is straightforward:

$$x_i = \hat{x}_i + e_i. \tag{3.2}$$

Note that in a few stages some flags are also sent to the entropy coder.

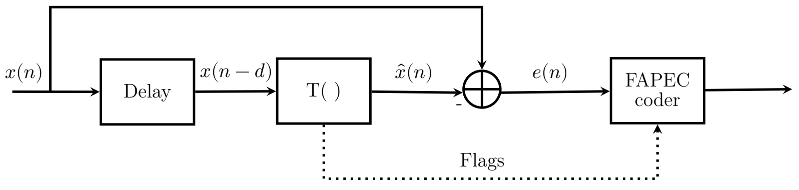


Figure 3.1: The FAPEC data compressor structure.

Figure 3.1 illustrates the general approach, where $T()$ is a generic transformation.

From Theorem 2.1 we know that the entropy is the average number of bits required to encode a block. Therefore, the main purpose of decorrelation stages in data compressors is to reduce the entropy at the coder input. In other words, the preprocessing stage performance is critical in the overall compressor efficiency.

FAPEC features several preprocessing algorithms such as a basic differential, multi-band prediction or even wavelet transform. In this work we introduce two new stages that follow this approach.

3.3 Entropy coder

FAPEC's entropy coder is inspired by the Golomb-Rice coder used in the CCSDS 121.0 standard for lossless data compression. FAPEC aims to perform similarly to Rice codes while fixing some of the problems related to them.

In order to achieve a good adaptation in front of outliers, FAPEC works on data blocks of, typically, 128 samples. The core analyzes every block of prediction errors and determines the optimum coding tables. Finally, it calls the coding kernel, which generates a variable-length code for every given prediction error. Although FAPEC needs larger data blocks compared to CCSDS 121.0, it has the advantage that the maximum code length is limited to less than twice the bit length of the input values.

Previously we have said that the core of FAPEC performs a statistical analysis of the prediction errors. If this analysis determines a very high entropy, FAPEC does not try to encode the values and they are sent to the output as is. On the other hand, the analysis might reveal very low entropy levels (about 90% of the values being -1, 0 or 1). Assuming a probability of $\frac{1}{3}$ for all of them and sequences of 6 values, there are 3^6 possible sequences, which can be encoded using 10 bits ($2^9 < 3^6 < 2^{10}$). The average is 1.66 bits per sample, a 95% of the Shannon limit. Finally, sequences of 5 zeroes or more are also detected and a run length encoding is performed.

3.4 Current FAPEC implementation

The last released version of FAPEC is 19.0, which is currently running in a cubesat constellation. In order to ensure portability between all platforms and a high efficiency, FAPEC is written in ANSI C. It also offers some features which we briefly describe:

- Stand-alone binary with multithreading support.
- Dynamic library that provides a simple API.
- Data chunking.
- Encryption.
- Several preprocessing stages.

Data chunking

FAPEC compresses data in chunks. Chunks are usually between 64 kB and 4 MB, yet the user may set a different value. After every chunk, all algorithms are reset, ensuring that every chunk can be decompressed independently of the others. This feature is compulsory in harsh environments where transmission errors may often occur. Note that data chunks are not the data blocks mentioned earlier (see 3.3), as the former are treated completely independently both in the preprocessing and the coder, and the latter are the length of the adaptive block in the coder.

Encryption

AES with 256-bit keys encryption may be enabled. With this option, every chunk will be encrypted after compression. This feature is only available if the target C compiler and system have the OpenSSL libraries.

Preprocessing stages

FAPEC 19.0 provides several preprocessing stages, specifically: basic delta, prediction-based multi-band images, text, LZW, tabular text or data, genomics, Kongsberg water column and DWT for images.

At this point the reader may not understand why this thesis proposes a new preprocessing stage for water column data if the current version of FAPEC already provides one. The reason is that a new file format containing water column information was released and the current stage does not support it. More details about this format are available in section 6.1.

On the other hand, after looking at all the available stages one notices that audio files do not have a specific stage. This fact motivates the development of a preprocessing algorithm that will be both useful for audio and IQ files.

Chapter 4
Development of new stages

New FAPEC stages must be well documented, efficient and easy to maintain. In order to achieve this, all stages are subject to some general requirements and must be evaluated. In this chapter we will first list the general requirements for new stages. Then, we will present four metrics to evaluate each stage.

4.1 Requirements for FAPEC stages

In this section we will list the general requirements that all FAPEC stages developed by DAPCOM must fulfill. Besides these, other specific requirements for each stage may be defined.

Requirements

1. The stage must be implemented in C.
2. The software shall be provided as a library and as a stand-alone binary.
3. The stage must be able to work with data chunks.
4. The stage must support lossless and, if applicable, lossy compression.
5. Data must not be lost even if the input is corrupted (i.e. delta fallback).
6. Compression ratio shall be better than that of Gzip.
7. Compression speed shall be better than that of Gzip.
8. The software shall be tested with a representative dataset.
9. The design and implementation must be documented.

Specifications

The following specifications shall be fulfilled with an Intel Core i7-10710U CPU.

1. The occupied memory with one thread shall not exceed 64 MB, assuming chunks of 1 MB.
2. FAPEC data chunks shall have a size between 128 kB and 4 MB.
3. Decompression time cannot be higher than a 110% of the compression time.
4. At least 10% of the source code lines must be comments.
5. McCabe complexity [McC76] shall be below 50.

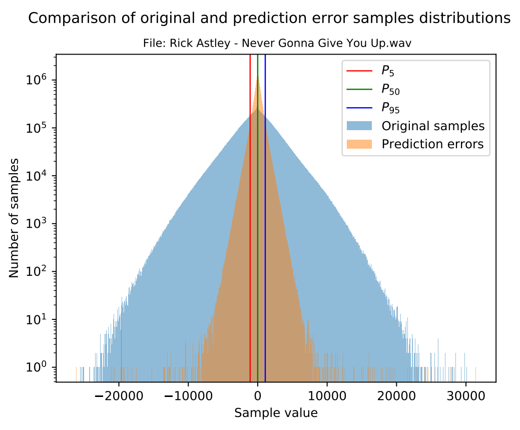
4.2 Evaluation of stages

The aim of this section is to present the metrics used in the following chapters to evaluate the developed preprocessing stages. The proposed indexes may be classified in two groups: information metrics and performance metrics.

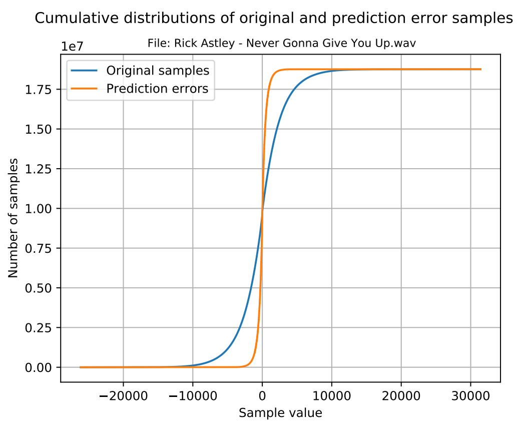
4.2.1 Information metrics

This subset of metrics is purely focused on evaluating how better compressible data are after the decorrelation stage. Our first approach is to plot a **histogram** of the original samples together with a histogram of the prediction errors (i.e. the values that will be sent to the FAPEC entropy coder). For a better visualization, the bin width is calculated from the number of samples and it is the same for both histograms. Besides, the prediction errors histogram also includes three vertical lines at percentiles 5, 50 and 95 in order to clearly show the range where 90% of data is contained.

Making use of the histograms, the **cumulative histograms** of the original samples and the prediction errors are calculated and plotted together. Although these graphics are just the integral of the histograms described above, they provide a clear way to see samples concentration in intervals. Figure 4.1 shows an example.



(a) Histogram of an audio file.



(b) Cumulative histogram of an audio file.

Figure 4.1: Histograms of a file compressed with Wave.

At the beginning of this thesis no more information metrics were planned, but we realized that these two are subject to data scaling. That is, with a simple division the metric can be adulterated. In order to deal with it, we also propose to use **negentropy** (see section 2.1.3) to measure how compressible the prediction errors are compared with the original samples. There are two main reasons behind choosing negentropy: the first is that it is invariant to linear maps, and the second is that it is a measure of non-gaussianity. The first property solves the problem of data scaling, and the second one offers a way of knowing if the stage has improved the data compression potential (i.e. negentropy is higher).

Although negentropy is a highly suitable metric for our purposes, it has an important drawback: its calculation is computationally very complex. Moreover, estimating it from data requires estimating the probability density function of the samples to later plug it into the definition of differential entropy. However, negentropy can be approximated with the following expression [HO00]:

$$J(X) \propto [E[G(X)] - E[G(V)]]^2 \tag{4.1}$$

where X is a random variable with $\mu = 0$ and $\sigma = 1$, $V \sim \mathcal{N}(0, 1)$ and $G(u)$ is a non-quadratic function. In our case, X and V will be data arrays and

$$G(u) = \ln \cosh u. \tag{4.2}$$

In order to obtain a more robust estimator, $G(u)$ must not grow too fast. Besides, Equation 4.2 has been proven to be very useful [HO00]. Further details on the choice of $G(u)$ can be found in [Hyv98].

4.2.2 Performance metrics

We are interested in evaluating the whole compression process: from opening the original file to writing the encoded one. In other words, we want to evaluate the performance of the decorrelation stage plus the entropy coder compared to other compressors.

In this thesis we propose a new metric based on Euclidean distance that takes into account the process time and the compression ratio. Specifically, given an input file, it consists in computing a dot in $\mathbb{R}^2$ where the first coordinate is the normalized process time and the second is the inverse of the compression ratio. Then, the Euclidean distance to the origin is computed for every algorithm and the one with the smallest distance is assumed to be the best one. Formally, the coordinates for each compressor are given by

$$\mathbf{a} = \left(t, \frac{s_c}{s_o}\right) \tag{4.3}$$

and the metric is

$$\gamma = \|\mathbf{a}\|^2 = t^2 + \left(\frac{s_c}{s_o}\right)^2 \tag{4.4}$$

where t is the process time, s_c the compressed file size in bytes and s_o the original file size also in bytes. Note that both metrics are in a range which allows us to compare them, hence time is assumed to be dimensionless.

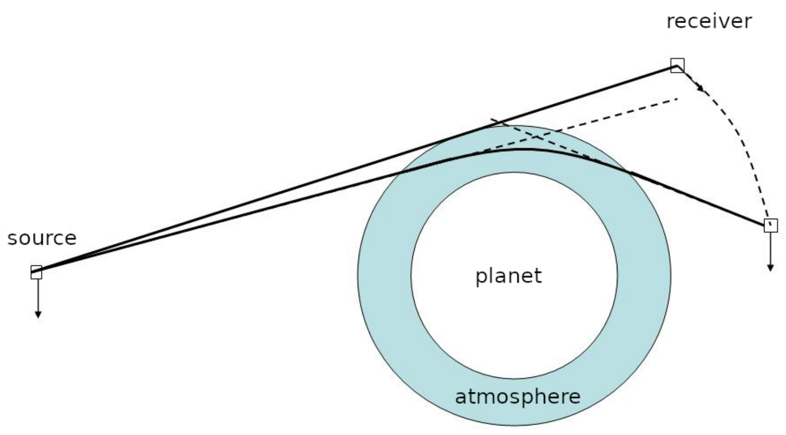
In order to reliably calculate the compression time, each compressor is run as a child-process on a Python script. Then, time is taken before spawning the child-process and again after it exits. Subtracting the start time from the end time gives the process time which is used in equations 4.3 and 4.4.

Finally, we want to point out that in the following chapters the metric γ will be given together with a plot including the dot $\mathbf{a}$ for several input files and compressors.

Chapter 5
Wave stage

Earth Observation techniques have experienced a growth both in terms of quality and quantity, which means that the amount of data produced have increased drastically. In addition, remote sensing is usually carried out by satellites with low storage capacity and limited bandwidth [San10], so clearly data compression is key in this situation.

Some satellites collect RF data, either from ground or air sources (e.g. vessels, airplanes or industrial activity in general), as well as from other satellites. Between the vast amount of remote sensing techniques we find Radio Occultation [Kur+97], which uses low orbit satellites to detect the changes produced by the atmosphere in a radio signal emitted by a GNSS satellite, as illustrated in Figure 5.1. RO data is mainly used as a weather forecasting tool.



Source: Wikipedia user MPRennie, CC BY-SA 3.0, via Wikimedia Commons.

Figure 5.1: Illustration of radio occultation (RO).

This chapter is part of a DAPCOM Data Services strategic project focused on developing a new FAPEC stage intended to compress data collected by satellites. The algorithm will be tested over IQ sample files and other formats structured in two channels, for instance, stereo audio. The proposed solution will be compared with the well-known FLAC coding technique.

5.1 Introduction to IQ data

In the introduction of this chapter we stated that the present stage will be focused on IQ data, that is, the discrete time samples of a RF IQ signal.

From communication theory we know that any pass-band signal $s(t)$ can be expressed as:

$$s(t) = i_s(t) \cdot \cos(2\pi f_0 t) - q_s(t) \cdot \sin(2\pi f_0 t) \tag{5.1}$$

where:

$i_s(t) \equiv$ In-phase component of the RF signal.

$q_s(t) \equiv$ Quadrature component of the RF signal.

$f_0 \equiv$ Carrier frequency of the RF signal.

For simplicity, we may work with the equivalent base-band signal $b_s(t)$:

$$b_s(t) = i_s(t) + jq_s(t). \tag{5.2}$$

Hence, the data to be compressed are the discrete time samples of the signals $i_s(t)$ and $q_s(t)$. In other words, our aim is to compress time series data separated in two channels.

For further information on IQ signals the reader may check references [Kir99], [Car10] and [Rib20a].

5.2 Motivation and applications of the Wave stage

We shall provide a justification on why we can use both IQ and audio files to test Wave and why we can compare it with the FLAC standard.

On the first hand, as we will see shortly, both FLAC and Wave implement, basically, the same preprocessing stage: a classical linear predictor. This decorrelation stage is followed by a Rice-Golomb entropy coder (see section 2.2.1) in FLAC and by the FAPEC entropy coder in Wave.

On the other hand, IQ data and stereo audio have the same structure: time series separated in two channels (notice that the samples distribution may not be the same). In fact, most SDR records are stored as WAV files, a well known audio format.

Additionally, in literature we may find papers about IQ data compression that make use of the FLAC algorithm, for instance [NA13].

To conclude, there are clear similarities between the two proposed algorithms and also between the proposed data formats. Hence, from a qualitative point of view, comparing Wave and FLAC with a dataset of stereo audio files is a logical choice.

5.3 Design

In this section we will design the algorithm intended to decorrelate audio and IQ data. The structure of the section will be the following: first we will state the specific requirements for the stage. Then, according to these and to the data structure described above, an algorithm will be proposed.

5.3.1 Requirements and specifications

Besides the requirements and specifications from section 4.1, Wave must also fulfill:

1. Compression ratio must be at least 80% of that from FLAC.
2. Compression speed must be better than that of FLAC.
3. It shall work for an arbitrary number of channels, at least up to a thousand.

5.3.2 Algorithm design

The first step in the design process is to determine the system input and output. Clearly, the input are the samples from a file. The output, as we have seen in section 3.2, are the prediction errors for our samples plus some flags.

In section 5.2 we justified why FLAC should have a good performance with IQ data. Taking this into account, the proposed algorithm should have a structure similar to that of FLAC, but using FAPEC as the entropy coder. This should translate into a compressor faster than FLAC and with a similar compression ratio.

From the FLAC format specifications we know that its preprocessing stage is based on a linear predictor [Coa01]. Therefore, our goal is to implement a predictor which, given an input sequence $x(n)$, predicts future samples following the model

$$\hat{x}(n) = \sum_{i=1}^{Q} h_i x(n-i) \tag{5.3}$$

where $\hat{x}(n)$ is the predicted sequence, $x(n-i)$ the previous observed samples, h_i the filter coefficients and Q the filter order.

From filter theory [Hay86] [Rib20b] we know that the coefficients h_i are given by

$$\mathbf{R}_x \mathbf{h} = \mathbf{r}_x \tag{5.4}$$

where $\mathbf{R}_x$ is a $Q \times Q$ Toeplitz matrix with elements $r_{ij} = r_x(|i-j|),\ 0 \leq i,j < Q$, $\mathbf{r}_x$ the correlation vector with $r_j = r_x(j),\ 1 \leq j \leq Q$ and $\mathbf{h}$ the filter coefficients.

Equation 5.4 can also be expressed with scalar notation as:

$$r_x(j) = \sum_{i=1}^{Q} h_i r_x(|i-j|),\ 1 \leq j \leq Q \tag{5.5}$$

where $r_x(j)$ is the lag j of the autocorrelation of $x(n)$.

In order to obtain the filter coefficients h_i we need to solve the previous system. In this case, using Gauss-Jordan elimination is not desirable as it has a complexity of $O(Q^3)$. However, there are algorithms such as the Levinson-Durbin recursion [IMS08] which, taking advantage of the Toeplitz structure of $\mathbf{R}_x$, reduce the computational complexity to $O(Q^2)$.

Equation 5.4 is obtained from a statistical approach to the problem, assuming that the sequence $x(n)$ is WSS. However, in our case $x(n)$ is a deterministic finite sequence from an unknown origin. For this reason, we need to split $x(n)$ in subsequences $x_N(n)$ of length N where assuming stationarity is reasonable. For each of these subsequences $x_N(n)$, the filter coefficients h_i are assumed to be independent (i.e. they are reset at the beginning of each subsequence). In this scenario, the autocorrelation lags for each subsequence would be given by the short-term autocorrelation sequence:

$$r_x(i) = \sum_{m=0}^{N-1-i} x_N(m)x_N(m+i), \quad i \geq 0. \tag{5.6}$$

It is important to remark that given the unknown character of the input data, N is a custom parameter that the user may choose when setting up the compressor.

However, for computational purposes, we will not use all the N samples of $x_N(n)$. Instead, we will set a training length $T \leq N$ and, as we are assuming stationarity, coefficients should be very similar. The autocorrelation will be calculated as follows:

$$r_x(i) = \sum_{m=0}^{T-1-i} x_N(m)x_N(m+i), \quad i \geq 0. \tag{5.7}$$

If lossy is enabled, samples will be divided and then floored just before this step. It is important to notice that lossy must be applied to samples instead of prediction errors to avoid accumulating quantization errors.

To sum up, the proposed algorithm consists in splitting $x(n)$ in sequences of size N for which the autocorrelation will be computed using T samples. Then, the filter coefficients h_i will be computed using the Levinson-Durbin algorithm and they will be sent to the entropy coder together with the prediction errors given by:

$$e(n) = x(n) - \hat{x}(n). \tag{5.8}$$

We should point out that recovering samples in decompression is as easy as calculating $\hat{x}(n)$ with Equation 5.3 and then adding it to the prediction error for that sample. Note that we coded the prediction errors and the coefficients. Taking this into account, it is obvious that decompression will be faster than compression.

In general, the previous algorithm will be applied independently in each channel, but the user may enable channel coupling and then the coefficients from the first channel will be reused in the others.

In Figure 5.2 we include a flowchart for the algorithm that we have just described, where *Settings* are the number of bits per sample, a boolean to enable or disable channel coupling, the level of losses, the number of channels, the period length N, the training length T and the filter order Q.

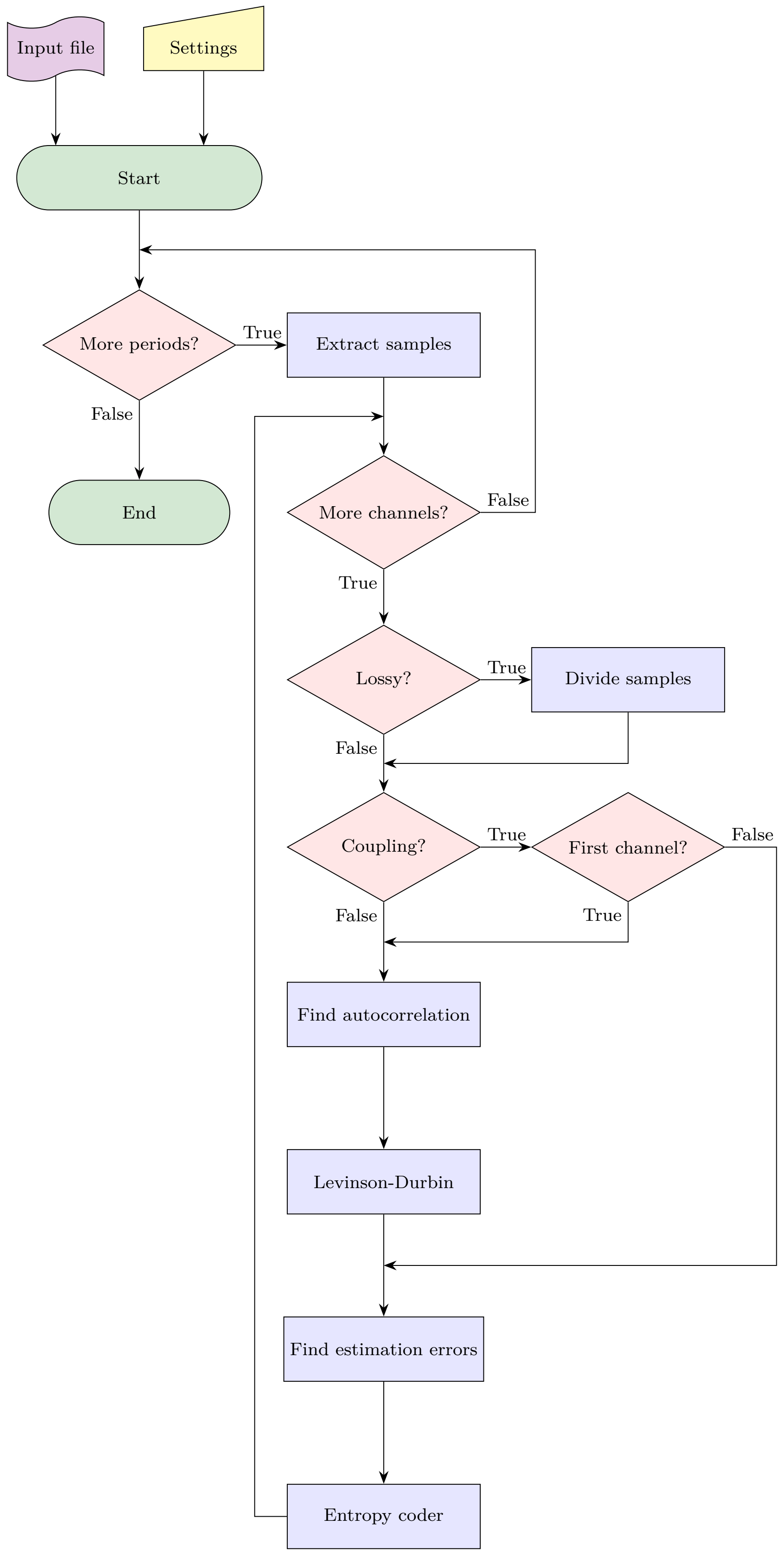


Figure 5.2: Flowchart of the Wave preprocessing stage.

5.4 Results

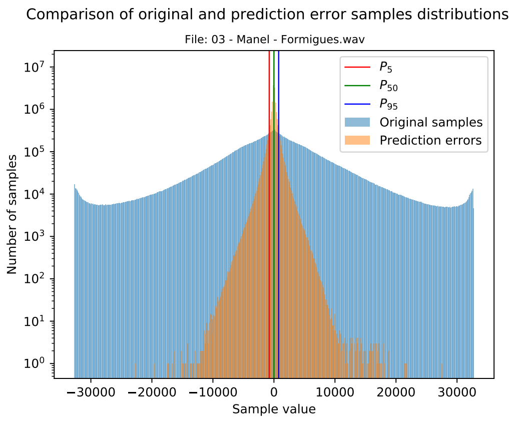
The final step in the development of a new stage is evaluating it. To do that, we will use the metrics proposed in section 4.2.

The results presented in this section have been obtained with the following settings:

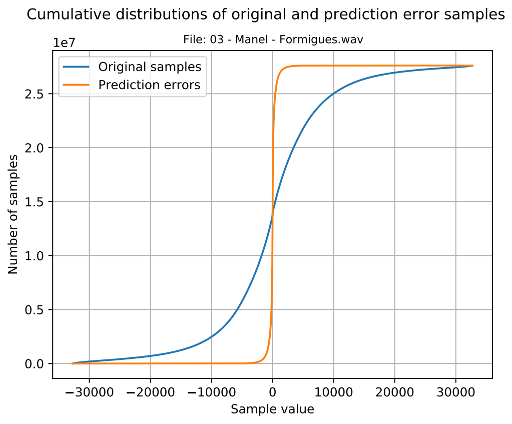
- Signed 16-bit samples.
- Filter order: $Q = 10$
- Period length: $N = 65536$
- Training length: $T = 65536$
- Number of channels: 2.
- No coupling.
- No losses.

Here we include negentropies, ratios, execution times and Euclidean distances for all our dataset, but we just show the histogram and the cumulative histogram for one file. The remaining ones can be found in appendix B.

We start our evaluation by showing the histogram (and cumulative histogram) of the song *Formigues*, by Manel, in Figure 5.3.



(a) Histogram.



(b) Cumulative histogram.

Figure 5.3: Histograms of *Formigues* by Manel, showing the range reduction.

We clearly see that histograms provide us a qualitative and visual way to identify the reduction of the dynamic range, but they are sensitive to scale. Besides, we also use negentropy to quantify if the preprocessing stage is suitable for our data (see section 4.2.1). For this reason, we have estimated negentropy before and after the decorrelation process. Note that an improvement of α times in negentropy does not imply that the compression ratio will be α times better.

As we can see in Table 5.1, negentropy of all files improves after applying the stage we have developed.

Filename	$J(X)$	$J(E)$	$J(E)/J(X)$
03 - Manel - Formigues.wav	0.0018	0.0144	8.0
04 - Pink Floyd - Time.wav	0.0004	0.0238	59.5
06 - Manel - Els entusiasmats.wav	0.0012	0.0098	8.2
06 - Pink Floyd - Money.wav	0.0009	0.0090	10

Table 5.1: Negentropies of audio files. It shows the performance of the Wave stage.

Finally, we compare the compression ratio and time of FAPEC, FLAC and GZIP.

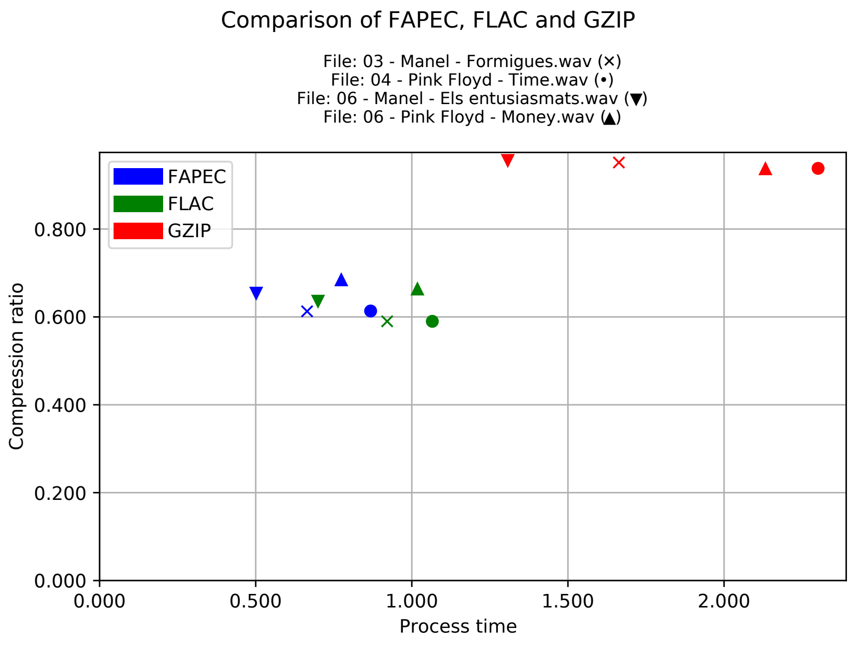


Figure 5.4: Comparison of compressors which shows that FAPEC is the fastest.

As we can see in Figure 5.4, FLAC and FAPEC have a similar performance, with the former having a slightly better ratio and the latter being faster. Applying our Euclidean distance criteria, we determine the FAPEC behaves slightly better than FLAC thanks to its speed. In Table 5.2 we include the numerical values of the Euclidean distances.

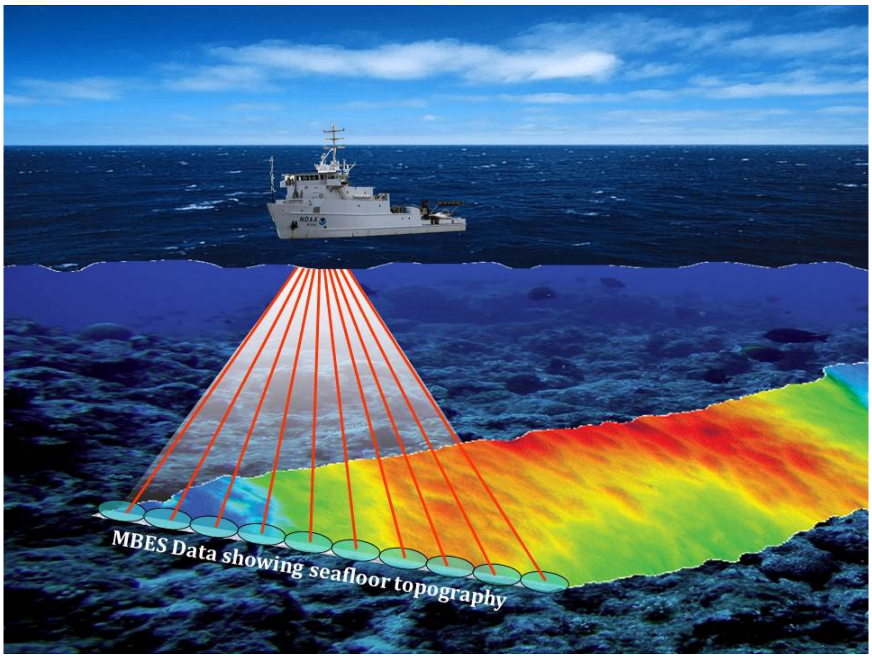
Filename	FAPEC dist.	FLAC dist.	GZIP dist.
03 - Manel - Formigues.wav	0.9040	1.0938	1.9161
04 - Pink Floyd - Time.wav	1.0630	1.2181	2.4851
06 - Manel - Els entusiasmats.wav	0.8242	0.9453	1.6197
06 - Pink Floyd - Money.wav	1.0333	1.2152	2.3294

Table 5.2: Euclidean distances from Figure 5.4.

Finally, we want to present a brief code evaluation. The McCabe complexity of the stage is 31, smaller than the maximum value set in the specifications. Besides, in the worst scenario, the maximum occupied memory is 18.3 MB. Thus, all the requirements and specifications are fulfilled.

Chapter 6
KMALL stage

Recent advances in sonar technology and computing power have led to an improvement in ocean monitoring. For instance, Multibeam Echosounders (MBES) are now capable of collecting data from the water column, besides some usual metrics such as seafloor reflectivity. This new information allows geoscientists to identify sunken structures, schools of fish, gas seeps, etc.



Source: NOAA Photo Library, CC BY 2.0, via Wikimedia Commons.

Figure 6.1: Conception of multibeam sonar on NOAA Ship NANCY FOSTER.

We clearly see that water column sensing has a big interest. However, their storage requirements are quite demanding (in the order of some GB/h). Hence, although it is possible to continuously record water column data, in practice it is only used in some specific explorations such as gas-seeps zones. In this scenario, efficient compression algorithms will be essential.

At the present moment, one of the biggest echosounder manufacturers is Kongsberg Maritime, a company with more than 7000 employees over 25 countries. In this chapter we will focus on the development of a preprocessing stage specially designed for the KMALL files from Kongsberg [Mar20]. First, a general overview of the data structures will be given. Then, with this format in mind, a design criterion will be proposed and implemented. Finally, we will analyze its performance and compare it with other compression algorithms.

6.1 The KMALL data format

In this section we describe the KMALL data structure and why there is a need to develop new compression algorithms for this format, but first, some concepts related to echosounders must be defined:

- **Ping:** A ping is defined as a number of pulses transmitted at approximately the same time.
- **Beam:** Each ping is formed by some pulses in different angles, which we call beams.

Notice that the number of beams per ping depends on each sonar model, and the numbers of samples per beam depends on the angle and on the depth. This dependence can be seen in Figure 6.3.

The `.kmall` files use a new format from Kongsberg which is just being implemented in new echosounder models. It is the successor of the Kongsberg `.all` format. Analyzing the latter is out of the scope of this project, yet we can state some of the improvements that KMALL brings.

On the first hand, KMALL is a generic format with high resolution data and with a datagram structure designed to avoid breaking existing decoders when updating the data structure.

On the other hand, `.all` format used to have a datagram size constraint of 64 kB due to the maximum size of UDP packets, but `.kmall` files are designed to be stored ”as is” and then fragmented if needed. Its main advantage is that pings are not splitted between datagrams.

The size of a `.kmall` file is not fixed. Actually, the echosounder operator decides when the file being recorded at that moment should end. In our dataset, kindly provided by Kongsberg Maritime, most of the files are between 100 and 400 MB. Inside these files there are several datagrams, which will be described later.

The mixture of data types in the same file makes it difficult for standard compressors like GZIP to identify data statistics. Therefore, they do not perform as good as one could expect. In order to improve the compression ratio, we are interested in knowing how are those datagrams placed inside the file and how to identify them. From Kongsberg KMALL documentation [Mar20], we know that all datagrams start with a generic header that contains the datagram size in bytes and a 4-character identifier. In the current version (410224 Revision H), there are the datagram types listed in tables 6.1 to 6.5.

Type code	Struct name	Description
#CPO	EMdgmCPO_def	Compatibility (C) data for position (PO).
#CHE	EMdgmCHE_def	Compatibility (C) data for heave (HE).

Table 6.1: Compatibility datagrams of the KMALL format.

Type code	Struct name	Description
#IIP	EMdgmIIP_def	Installation and sensor parameters.
#IOP	EMdgmIOP_def	Runtime operator parameters.
#IBE	EMdgmIB_def	Built in test (BIST) error report.
#IBR	EMdgmIB_def	Built in test (BIST) reply.
#IBS	EMdgmIB_def	Built in test (BIST) short reply.

Table 6.2: Installation and runtime datagrams of the KMALL format.

Type code	Struct name	Description
#SPO	EMdgmSPO_def	Sensor (S) data for position (PO).
#SKM	EMdgmSKM_def	Sensor (S) KM binary sensor format.
#SVP	EMdgmSVP_def	Sensor (S) data from sound velocity (V) profile (P) or CTD.
#SVT	EMdgmSVT_def	Sensor (S) data for sound velocity (V) at transducer (T).
#SCL	EMdgmSCL_def	Sensor (S) data from clock (CL).
#SDE	EMdgmSDE_def	Sensor (S) data from depth (DE) sensor.
#SHI	EMdgmSHI_def	Sensor (S) data for height (HI).

Table 6.3: External sensor output datagrams of the KMALL format.

Type code	Struct name	Description
#FCF	EMdgmFCF_def	Backscatter calibration (C) file (F) datagram.

Table 6.4: File datagrams of the KMALL format.

Type code	Struct name	Description
#MRZ	EMdgmMRZ_def	Multibeam (M) raw range (R) and depth (Z) datagram.
#MWC	EMdgmMWC_def	Multibeam (M) water (W) column (C) datagram.

Table 6.5: Multibeam datagrams of the KMALL format.

It is important to notice that a given `.kmall` file can contain all the above datagrams. However, we may usually find that water column data is logged in a separate file with extension `.kmwcd`. In other words, MWC datagrams will be stored in `.kmwcd` files instead of `.kmall` files, but the decoding process is the same for both extensions as the data format is exactly the same.

As we will see in the next section, our main interest in this chapter will lie in MWC datagrams. Therefore, before continuing, an appropriate description of them shall be given.

MWC datagrams are structured as follows:

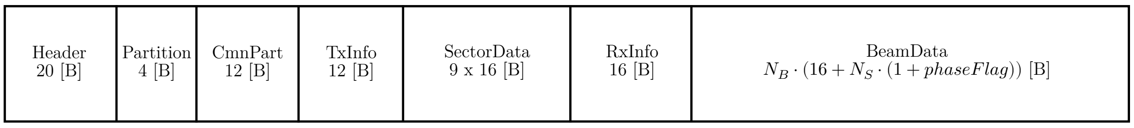


Figure 6.2: MWC datagram as described by Kongsberg (410224 Revision H).

Where:

$N_B \equiv$ Number of beams in ping. It is stored as a 16-bit integer in *RxInfo*.

$N_S \equiv$ Number of samples in beam. It is stored as a 16-bit integer in *BeamData*.

$phaseFlag \equiv$ Flag that indicates if there is phase information after amplitude samples. It may be 0 (no phase information), 1 (8-bit resolution) or 2 (16-bit resolution). It is stored as an 8-bit integer in *RxInfo*.

In the present work we will only use information contained in *Header* such as the datagram size and the datagram type, in *RxInfo* and in *BeamData*. For further details, the interested reader is referred to [Mar20].

6.2 Design

We have already described the KMALL format and we have also justified why this stage is needed. Now it is time to start designing an algorithm to decorrelate the data in these files. We will follow a two-step approach: first we will analyze the KMALL files in our dataset to determine which of the datagrams listed in section 6.1 are worth to be compressed. Then, we will propose an algorithm that suits the requirements stated in section 4.1. There are no specific requirements for this stage.

6.2.1 Data analysis

In order to decide what to compress, we have first analyzed some `.kmall` and `.kmwcd` files to determine the percentage of occupancy by datagram type. The procedure we have followed is the following:

1. Implement the existing C++ KMALL format in Python.
2. Implement a Python class which acts as top level API for the KMALL files.
3. Write a Python script that takes a directory and an extension and returns the percentages of every datagram type.

The result is that for `.kmall` files, MRZ datagrams are a 92% of the total size; and for `.kmwcd`, MWC datagrams are a 99%. From this evaluation it is clear that these two data types are our objective; however, they are too different to be preprocessed with the same stage, therefore in this thesis we will ignore MRZ datagrams and we will just focus on MWC data.

If we look closer at a MWC datagram (see Figure 6.2) we see that all the headers size is insignificant compared with the samples volume. Hence, we will only compress the payload, which includes amplitude and phase information.

In conclusion, we will design a stage which will only give some special attention to the information contained in the *BeamData* subdatagram from MWC datagrams.

6.2.2 Algorithm design

In 2019, DAPCOM developed a FAPEC stage to compress MWC data contained in `.wcd` files [Por+19], the predecessor of `.kmwcd`. For this reason, the stage we are presenting here is not strictly new, but an adaption of an already existing one.

Our first concern is FAPEC data chunking. This feature may be dangerous because splitting a datagram into different chunks is not desirable as it complicates file reading. For this reason, assuming a degradation in performance, we will allow chunks with a true size smaller than the chunk size set by the user. We should remark that even with this constraint, the chunk size will be typically between 1 MB and 2 MB, therefore fulfilling specification 2.

The second step is to identify MWC datagrams. To do that, we just need to read the datagram type field in the header. If it is a MWC datagram, a custom algorithm is applied. Otherwise, samples are directly sent to the entropy encoder.

Now it is time to focus on the payload data and how can we decorrelate it. If we represent the amplitude samples as a raw rectangular image (see Figure 6.3) we can see that correlation between beams is higher than between samples (i.e. the image is smoother from top to bottom than left to right). Although this is purely qualitative, it matches with the algorithm designed in [Por+19], and it can be proved to be true by applying a simple differential first by rows and then by columns and finally comparing the results. In the picture we can also see that the number of samples per beam is not constant, but it has a parabolic behavior.

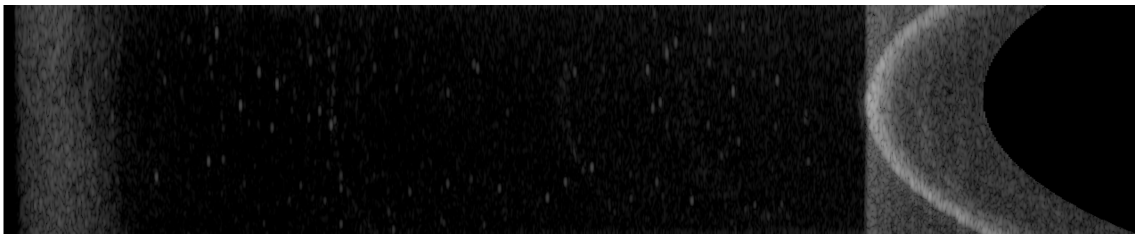


Figure 6.3: Raw representation of MWC amplitude from a EM 2040 echosounder.

In the implementation for `.wcd` files, given a sample $S_{i,j}$ in the position i of the beam j, the vertex coordinates (V_x, V_y) and the number of beams N_B, prediction errors $E_{i,j}$ in the rectangular region are calculated:

$$E_{i,j} = S_{i,j} - S_{i,j-1}, \qquad 0 \leq i < V_x, \quad 1 \leq j < N_B. \tag{6.1}$$

In other words, the samples from the rectangular region are predicted from the samples in the same position as those from the previous beam.

The remaining samples are predicted following a simple differential between them and a correction coefficient. Formally, given N_{S_j} the number of samples in the beam j, then

$$E_{i,j} = S_{i,j} - S_{i-1,j} - C_{i,j}, \qquad Vx \leq i < N_{S_j}, \quad 1 \leq j < N_B \tag{6.2}$$

where $C_{i,j}$ is given by:

$$C_{i,j} = \frac{1}{16}(7E_{i-1,j} + 6E_{i-2,j} + 4E_{i-3,j} + 3E_{i-4,j}). \tag{6.3}$$

As the reader may have noticed, these equations are not defined for $j = 0$ (i.e. the first beam). The reason is that the system must be causal, therefore the first beam is predicted from the previous sample in the same beam:

$$E_{i,0} = S_{i,0} - S_{i-1,0}, \qquad 1 \leq i < N_{S_0}. \tag{6.4}$$

The algorithm we propose in this thesis is very similar to the one we have just described for `.wcd` files, but with some improvements. We also follow the approach of predicting samples from the previous beam, but we do not restrict to the square region. Instead, we do that for all the samples which have a sample above. Formally,

$$E_{i,j} = S_{i,j} - S_{i,j-1}, \qquad 0 \leq i < \min(N_{S_{j-1}}, N_{S_j}), \quad 1 \leq j < N_B. \tag{6.5}$$

The remaining samples are predicted with the previous samples. That is,

$$E_{i,j} = S_{i,j} - S_{i-1,j}, \qquad \min(N_{S_{j-1}}, N_{S_j}) \leq i < N_{S_j}, \quad 1 \leq j < N_B. \tag{6.6}$$

The prediction errors for the first beam are also calculated with Equation 6.4, with the addition that

$$E_{0,0} = S_{0,0} + 128. \tag{6.7}$$

Now we are interested in observing if this algorithm also works for the phase information present in MWC datagrams. Notice that the former stage for `.wcd` was not designed to work with phase samples. In Figure 6.4 we show the phase corresponding to the amplitudes in Figure 6.3. As the reader can see, this information is almost white noise, therefore it is theoretically impossible to compress. Our final decision has been to apply a simple differential between samples (i.e. Equation 6.6).

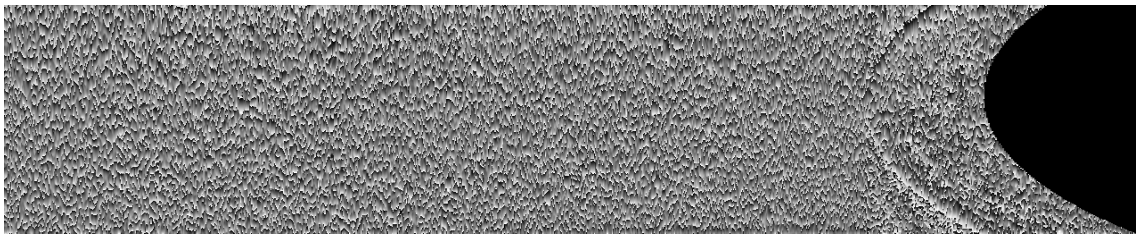


Figure 6.4: Raw representation of MWC phase from a EM 2040 echosounder.

Finally, we should remark that more complex algorithms like nearest-neighbor or a generic linear predictor of order Q have been tested, but they are slower and also have a worse compression ratio.

In Figure 6.5 a flowchart for the KMALL stage is included, where *Settings* are two parameters: amplitude and phase losses levels.

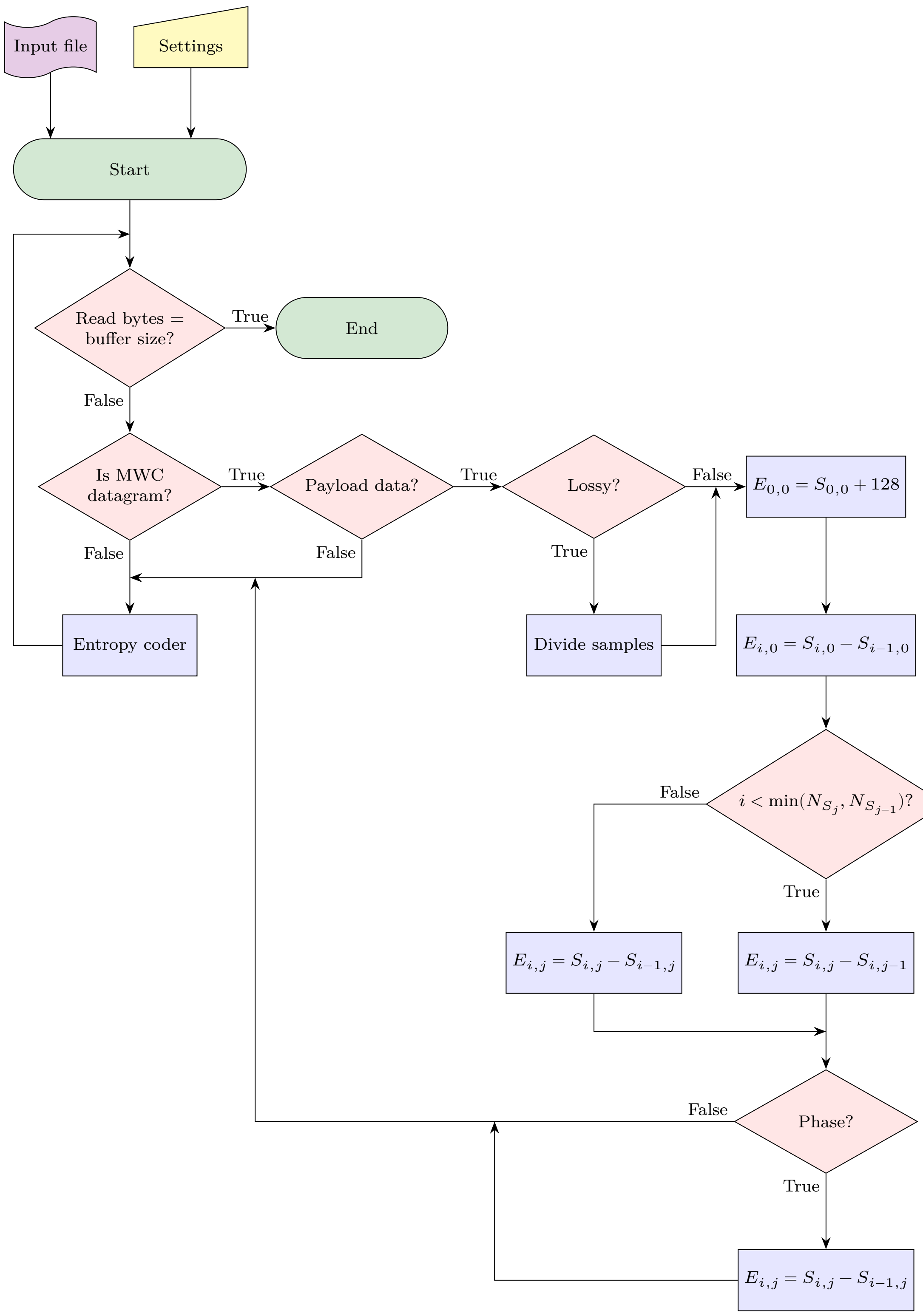
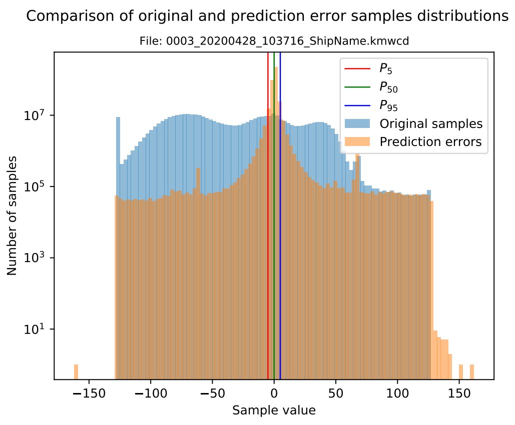

Input file
Settings
Start
Read bytes = buffer size?
True
End
False
Is MWC datagram?
True
Payload data?
True
Lossy?
False
$E_{0,0} = S_{0,0} + 128$
False
False
True
Entropy coder
Divide samples
$E_{i,0} = S_{i,0} - S_{i-1,0}$
False
$i < \min(N_{S_j}, N_{S_{j-1}})$?
True
$E_{i,j} = S_{i,j} - S_{i-1,j}$
$E_{i,j} = S_{i,j} - S_{i,j-1}$
False
Phase?
True
$E_{i,j} = S_{i,j} - S_{i-1,j}$

Figure 6.5: Flowchart of the KMALL preprocessing stage.

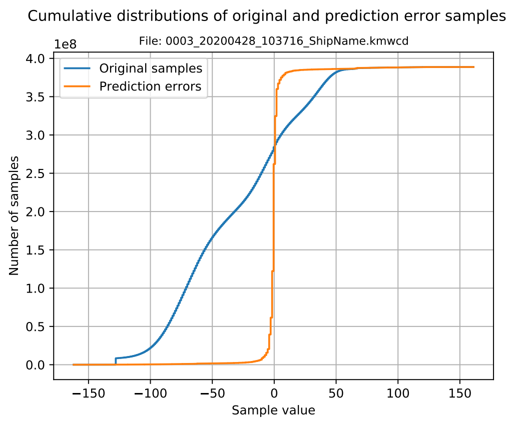
6.3 Results

To evaluate the KMALL stage we also use the metrics from section 4.2. All the results have been obtained with lossless compression.

As we did in Wave, here we include the histogram and cumulative histogram of one file (Figure 6.6), but further examples may be found in appendix C.



(a) Histogram.



(b) Cumulative histogram.

Figure 6.6: Histograms showing the range reduction of MWC data from a EM 304.

From the histograms we see that original data is distributed among the whole dynamic range, which would result in a poor compression ratio. After the preprocessing stage we propose, 90% of the samples fall between -7 and +7.

If we look at the negentropies in Table 6.6. we see that for all the files the negentropy is higher after preprocessing, thus improving the posterior compression ratio. It is important to notice that we have estimated negentropies up to the 4th decimal, thus files with a very low negentropy are estimated with 0. We should also point out that the improvement for the last file is worse because it is the only one with phase information, which is almost incompressible (see Figure 6.4).

Filename	$J(X)$	$J(E)$	$J(E)/J(X)$
0003_20200428_103716_ShipName.kmwcd	0.0004	0.0530	132.5
0004_20200428_093723_ShipName.kmwcd	0.0000	0.0252	-
0039_20200428_115408_ShipName.kmwcd	0.0001	0.0146	146
0010_20210409_092708.kmwcd	0.0001	0.0089	89

Table 6.6: Negentropies before and after preprocessing of files holding MWC data.

We see that the KMALL stage has good results reducing the dynamic range of samples and increasing negentropy, but it would be almost useless if a general purpose compressor such as GZIP had a better compression ratio or was faster. For this reason, now we will compare the performance of FAPEC with the KMALL stage and GZIP. We will proceed as in Wave and use the Euclidean distance metric proposed in section 4.2.2.

In Figure 6.7 we plot the compression time and ratio for different files using FAPEC and GZIP. We see that FAPEC behaves much better than GZIP both in speed and ratio. For the particular case of a file with phase, both algorithms perform poorly in terms of ratio but FAPEC is much faster. Besides, FAPEC allows applying lossy compression to phase and lossless to amplitude.

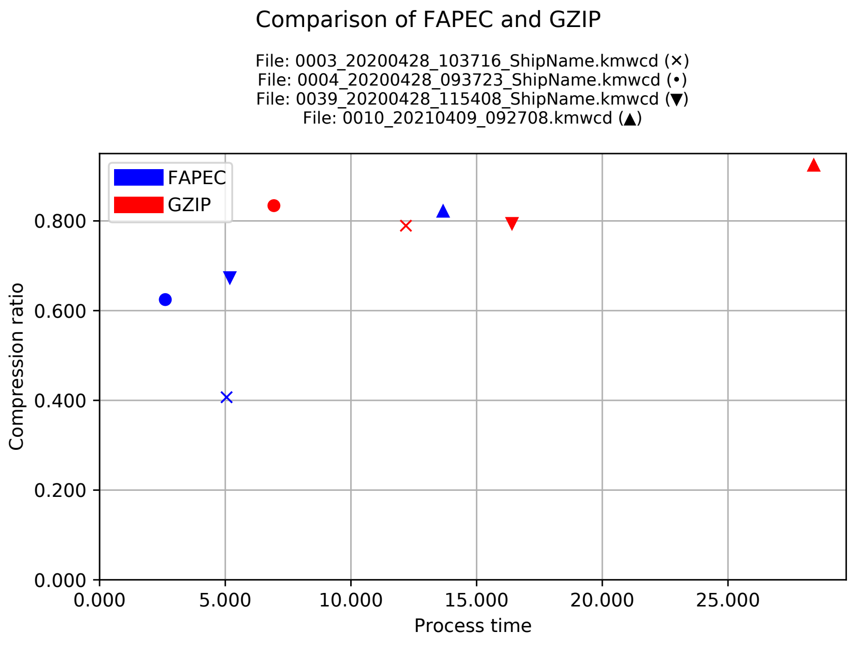


Figure 6.7: Comparison of compressors showing that FAPEC performs better both in time and ratio than GZIP.

In Table 6.7 we present the numerical values of the distances in Figure 6.7.

Filename	FAPEC dist.	GZIP dist.
0003_20200428_103716_ShipName.kmwcd	5.0696	12.2123
0004_20200428_093723_ShipName.kmwcd	2.6879	6.9850
0039_20200428_115408_ShipName.kmwcd	5.2258	16.4260
0010_20210409_092708.kmwcd	13.6912	28.4271

Table 6.7: Euclidean distances from Figure 6.7.

Our final considerations are that the stage has a McCabe complexity of 14, much smaller than 50 (see section 4.1); and a memory occupation of 32 MB in the worst case. In conclusion, the proposed algorithm satisfies all requirements and specifications.

Chapter 7
Conclusions and future work

This project had a very clear objective since the beginning: develop two preprocessing stages for Radio Frequency (RF) and KMALL MWC data. The process to reach this goal was not straightforward, because in order to propose an algorithm one needs to know well the data structure. However, understanding the data structure is not enough, as one must also understand the FAPEC framework, and it has truly been one of the hardest parts of this project.

In addition to the two stages, we also wanted to propose a universal metric to evaluate them. We finally decided to use negentropy, but arriving there required several meetings and many hours to develop the basic theoretical framework, which we decided to include in chapters 2 and 4. Apart from the formal definition of negentropy (see section 2.1.3), we also present Equation 4.1 to approximate it from data, which resulted to be very useful thanks to its consistency and its low computational complexity.

In sections 5.4 and 6.3 we have evaluated the Wave and the KMALL stages, respectively. In the former, we have used a classical linear predictor to estimate the input samples and then the prediction errors have been calculated. Remember from section 3.2 that the prediction errors are the values sent to the FAPEC entropy coder, hence we have compared the negentropy of the input file samples and the prediction errors. In Table 5.1 we can see that negentropy increases for all the files in our dataset.

On the other hand, the achieved compression ratios are a bit worse than those of FLAC. However, we are using a filter order of 10 which is quite high, and even in this situation FAPEC is much faster and its speed compensates the ratios, as can be seen in Table 5.2.

In the KMALL stage we have developed a tailored algorithm for the `.kmwcd` files from Kongsberg Maritime. The algorithm takes advantage of the high correlation between samples in the same column (see Figure 6.3) to estimate a sample from the sample in the previous row but the same column, when possible (see section 6.2.2 for a more concise explanation). Following this approach, negentropy increases by about two orders of magnitude, as can be seen in Table 6.7. At this point it is worth to point out that trying to run FAPEC on `.kmwcd` files without any kind of preprocessing results in no compression at all, result that matches with the zero negentropy in the second column of the previous table.

The last important result is that FAPEC with the KMALL stage performs better both in term of process time and compression ratio than GZIP (see figure 6.7). In average, FAPEC has a compression ratio 1.4 times better and is 2.5 times faster. Finally, note that the KMALL stage is only compared with GZIP because we were not able to find any specific algorithms for `.kmwcd` files, if they exist at all.

Our final words are that the results are satisfactory and meet the specifications, although some stages are open for future work and improvements. In the following, we list different ideas that can be explored beyond this work:

- In the KMALL stage, analyze the entropy of the original samples per beam. This procedure is similar to the one in [Mos+13] and could help to detect where the entropy is lower and apply different techniques on different zones.
- Explore other audio encoding algorithms such as TAK in order to take some ideas and improve the Wave stage.
- Improve the KMALL stage with a new algorithm to compress MRZ datagrams, which take around a 90% of `.kmall` files and contain the most commonly used information.

Besides improving the stages here proposed, another interesting line of work is to find a relation between the negentropy and the compression ratio achieved by FAPEC. As already observed in section 5.4, the negentropy of `04 - Pink Floyd - Time.wav` is 2.42 times better than that of `06 - Manel - Els entusiasmats.wav`, but the compression ratios are 0.64 and 0.68, which clearly do not have a relation of 2.42. This research will hopefully conclude with the publication of a paper.

Since the beginning this project was a way to open the door to different formats which FAPEC did not support. After this first proposal, the proposed stages will be implemented and used in real situations with real data, revealing the high applicability of this work.

Bibliography

[Car10] A.B. Carlson. *Communication System.* Tata McGraw-Hill Education, 2010. ISBN: 9780071321174.

[Coa01] Josh Coalson. *FLAC - format.* 2001. URL: `https://xiph.org/flac/format.html` (visited on 2021-03-10).

[Com94] Pierre Comon. “Independent component analysis, A new concept?” In: *Signal Processing* 36.3 (1994), pp. 287–314.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* USA: Wiley-Interscience, 2006. ISBN: 0471241954.

[Dud+15] Jarek Duda et al. “The use of asymmetric numeral systems as an accurate replacement for Huffman coding”. In: (2015), pp. 65–69.

[GV75] R. Gallager and D. van Voorhis. “Optimal source codes for geometrically distributed integer alphabets (Corresp.)” In: *IEEE Transactions on Information Theory* 21.2 (1975), pp. 228–230.

[Gol66] S. Golomb. “Run-length encodings”. In: *IEEE Transactions on Information Theory* 12 (3 1966), pp. 399–401.

[Hay86] S. Haykin. *Adaptive Filter Theory.* USA: Prentice-Hall, Inc., 1986. ISBN: 0130040525.

[Hyv98] A. Hyvärinen. “New Approximations of Differential Entropy for Independent Component Analysis and Projection Pursuit”. In: 10 (1998). Ed. by M. Jordan, M. Kearns, and S. Solla.

[HO00] A. Hyvärinen and E. Oja. “Independent component analysis: algorithms and applications”. In: *Neural Networks* 13.4 (2000), pp. 411–430.

[IMS08] B. Iser, W. Minker, and G. Schmidt. *Bandwidth extension of speech signals.* New York: Springer, 2008. Chap. Levinson-Durbin Recursion. ISBN: 978-0-387-68899-2.

[Kir99] Johan Kirkhorn. “Introduction to IQ-demodulation of RF-data”. In: *IFBT, NTNU* 15 (1999).

[Kur+97] E. R. Kursinski et al. “Observing Earth’s atmosphere with radio occultation measurements using the Global Positioning System”. In: *Journal of Geophysical Research: Atmospheres* 102.D19 (1997), pp. 23429–23465.

[Mac02] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms.* USA: Cambridge University Press, 2002. ISBN: 0521642981.

[Mar20] Kongsberg Maritime. *EM datagrams on *.kmall format.* 2020. URL: `https://www.kongsberg.com/globalassets/maritime/software/410224ah_em_dgm_format.zip` (visited on 2021-03-17).

[McC76] T.J. McCabe. “A Complexity Measure”. In: *IEEE Transactions on Software Engineering* SE-2.4 (1976), pp. 308–320.

[Mos+13] Marek Moszynski et al. “A novel method for archiving multibeam sonar data with emphasis on efficient record size reduction and storage”. In: *Polish Maritime Research* 20.1 (2013), pp. 77–86.

[NA13] S. Nanba and A. Agata. “A new IQ data compression scheme for fronthaul link in Centralized RAN”. In: *2013 IEEE 24th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC Workshops)* (2013), pp. 210–214.

[Por+18] J. Portell et al. “FAPEC, a versatile and efficient data compressor for space missions”. In: *International Journal of Remote Sensing* 39.7 (2018), pp. 2022–2042.

[Por+19] Jordi Portell et al. “High-Performance Compression of Multibeam Echosounders Water Column Data”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 12.6 (2019), pp. 1771–1783.

[Rib20a] J. Riba. *Introducción a las Comunicaciones*. Barcelona: ETSETB (UPC), 2020.

[Rib20b] J. Riba. *Procesado de Señal Audiovisual y de Comunicaciones*. Barcelona: ETSETB (UPC), 2020.

[RP71] R. Rice and J. Plaunt. “Adaptive Variable-Length Coding for Efficient Compression of Spacecraft Television Data”. In: *IEEE Transactions on Communication Technology* 19.6 (1971), pp. 889–897.

[San10] Rainer Sandau. “Status and trends of small satellite missions for Earth observation”. In: *Acta Astronautica* 66.1 (2010), pp. 1–12.

[Sha48] C. Shannon. “A Mathematical Theory of Communication”. In: *tech. J* 27.379 (1948), p. 623.

[Spa20] Consultative Committee for Space Data Systems (CCSDS) 121.0-B-3. *Lossless Data Compression*. 2020-08. URL: `https://public.ccsds.org/Pubs/121x0b3.pdf`.

[Spa17] Consultative Committee for Space Data Systems (CCSDS) 122.0-B-2. *Image Data Compression*. 2017-09. URL: `https://public.ccsds.org/Pubs/122x0b2.pdf`.

[Vas76] Oldrich Vasicek. “A Test for Normality Based on Sample Entropy”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 38.1 (1976), pp. 54–59.

[YRM91] Pen-Shu Yeh, Robert Rice, and Warner Miller. “On the optimality of a universal noiseless coder”. In: *JPL Technical Report* 91-2 (1991).

Appendix A
Proof of Equation 2.5

In some papers such as [Vas76] the authors use Equation 2.5 instead of Equation 2.4, but no formal proof is included. In this appendix we provide a clear proof which shows that equations 2.4 and 2.5 are equivalent.

Let X be a continuous random variable with cumulative distribution function $F(x)$. Then, its quantile function is defined as:

$$Q(p) = F^{-1}(p) = \inf\{x \in \mathbb{R} : p \leq F(x)\} \tag{A.1}$$

for a probability $0 \leq p \leq 1$.

From the previous definition we obtain

$$F(x) = F(Q(p)) = p \tag{A.2}$$

$$Q(p) = Q(F(x)) = x. \tag{A.3}$$

Applying the chain rule to Equation A.2:

$$\frac{dF(Q(p))}{dp} = \frac{dF(Q)}{dQ} \cdot \frac{dQ(p)}{dp} = f(Q(p)) \cdot Q'(p) = 1. \tag{A.4}$$

Now, take the differential entropy as defined in 2.4 and, for simplicity, rewrite it as:

$$h(X) = -\int_{-\infty}^{+\infty} f(x) \ln f(x) dx = -\int_{-\infty}^{+\infty} u(x) dx. \tag{A.5}$$

Applying the change of variables from A.3 to A.5 and remembering that $F(x)$ is monotonically increasing, we obtain the integration limits

$$a = \lim_{x \to -\infty} F(x) = 0 \tag{A.6}$$

$$b = \lim_{x \to +\infty} F(x) = 1 \tag{A.7}$$

and the integrand

$$h(X) = -\int_0^1 u(Q(p)) \cdot Q'(p) dp = -\int_0^1 f(Q(p)) \cdot \ln(f(Q(p))) \cdot Q'(p) dp \tag{A.8}$$

which, making use of Equation A.4, can be simplified to Equation 2.5:

$$h(X) = -\int_0^1 \ln\left(f(Q(p))\right) dp = -\int_0^1 \ln\left(\frac{1}{Q'(p)}\right) dp = \int_0^1 \ln\left(Q'(p)\right) dp. \tag{A.9}$$

■

Appendix B
Additional Wave histograms

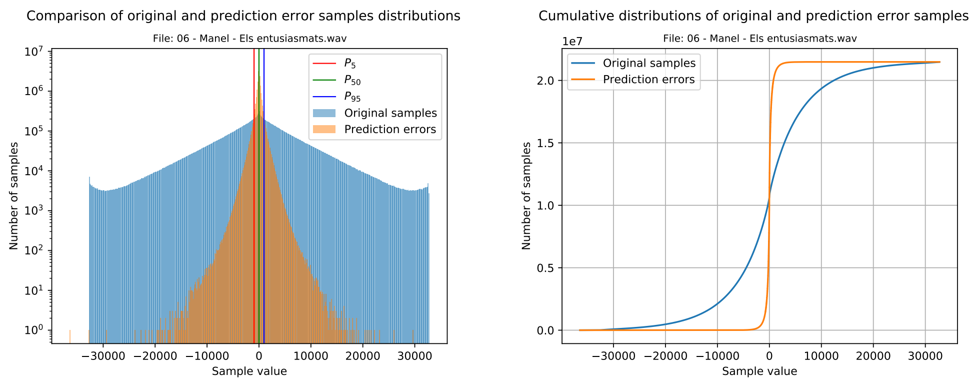


Figure B.1: Histograms of *Els entusiasmats* by Manel.

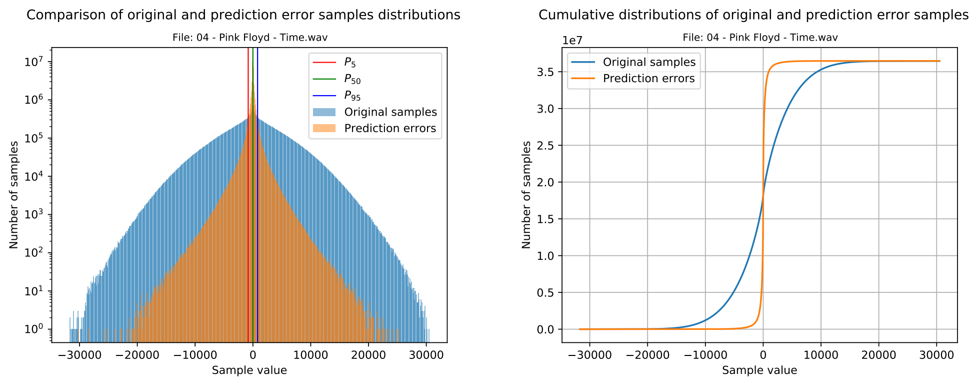


Figure B.2: Histograms of *Time* by Pink Floyd.

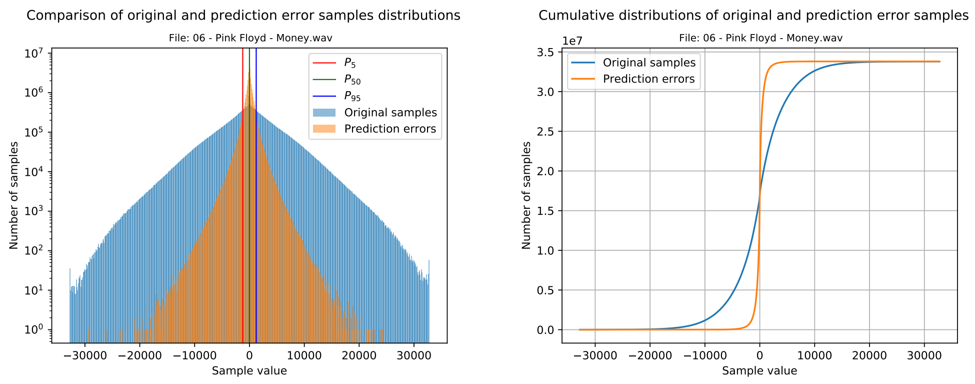


Figure B.3: Histograms *Money* by Pink Floyd.

Appendix C
Additional KMALL histograms

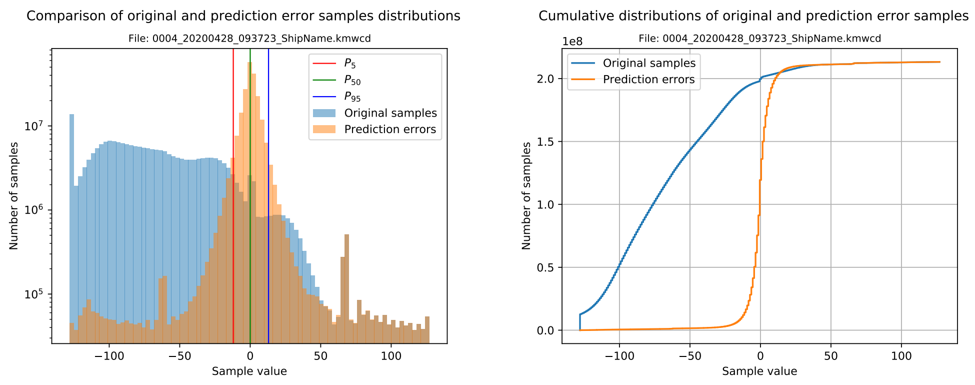


Figure C.1: Histograms of EM 712 retrieved MWC data.

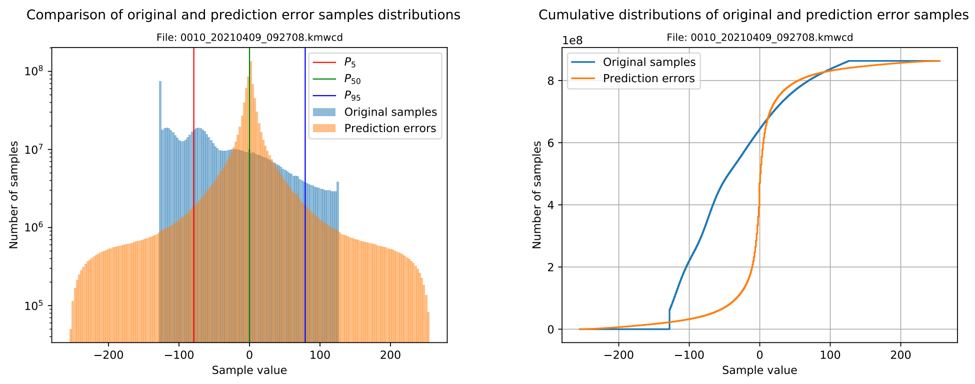


Figure C.2: Histograms of EM 2040 retrieved MWC data (with phase).

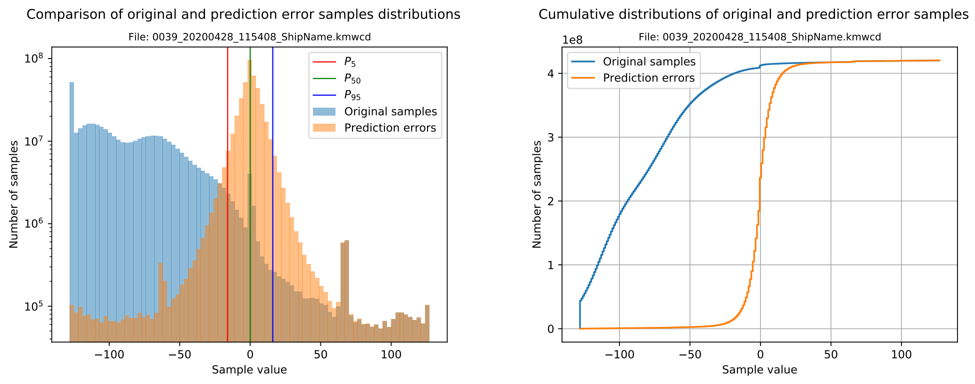


Figure C.3: Histograms of EM 2040 retrieved MWC data.

Acronyms

AES Advanced Encryption Standard. 8

ANSI American National Standards Institute. 8

API Application Programming Interface. 8, 22

CCSDS Consultative Committee for Space Data Systems. 4, 6, 7

CPU Central Processing Unit. 9

DWT Discrete wavelet transform. 8

FAPEC Fully Adaptive Prediction Error Coder. ii, iii, v, vii, 1, 2, 4, 6–10, 12–14, 18, 23, 26–29

FLAC Free Lossless Audio Codec. ii, iii, 1, 2, 4, 12–14, 18, 28

GNSS Global Navigation Satellite System. 12

IQ In-phase & Quadrature. 2, 8, 12–14

LZW Lempel-Ziv-Welch. 8

MWC Multibeam Water Column. vii, viii, 2, 22–24, 26, 28, 34

RF Radio Frequency. ii, iii, 1, 2, 12, 13, 28

RO Radio Occultation. 12

SDR Software Defined Radio. 1, 13

TAK Tom's lossless Audio Kompressor. 29

UB University of Barcelona. 1

UDP User Datagram Protocol. 20

UPC Polytechnic University of Catalonia. 1, 2

WAV Waveform Audio Format. 13

WSS Wide-Sense Stationary. 15