Multimedia Coding

Project #2

DPCM system for Audio Coding

Due on January 2021

Prof. Giancarlo Calvagno

Elena Camuffo 1234370

Abstract

DPCM coding is a widely used technique in audio and image processing. It does encode differences instead of coding input symbols and this leads to a great improvement in the efficiency of the system. In this project these differences are encoded using *Golomb codes*, which suites well sequences distributed according to a Geometric pdf. The performances of several different audio sources (Mono, 16bits) are investigated and conclusions are inferred.

Contents

1 Introduction

In signal processing, the process of encoding information using fewer bits than the original representation is called *data compression* or *source coding.* Any particular compression can be either *lossy* or *lossless.*

- **Lossless** data compression techniques involve a reversible coding procedure, so that the original content can be perfectly recovered from the compressed data. This compression procedure is possible only for digital sources, because any transformation applied to analog data automatically implies some loss. The leading principle of this class of coding techniques is to exploit the statistical redundancy present in the data and reduce it to the bone, typically mapping most frequent words into shorter codewords and least frequent ones into longer codewords.
  The main application of lossless coding is in text compression, where a lot of redundancy can be found. Many techniques for lossless coding exist nowadays, such as Huffman coding, Arithmetic coding and Dictionary-based techniques like LZ77, LZ78, LZW. Also Run-Length coding techniques are classified as lossless, and among them it is worth to mention **Golomb coding**, deeply explained in section 2.2. However those techniques achieve not too high levels of compression; this is pertinence of the lossy techniques.

- **Lossy** data compression techniques do involve some loss of information and the coding procedure is generally an irreversible transformation that approximate the original content. This compression procedure introduces some distortion in the reconstruction, and it is needed in the case of analog sources. The compression ratio achievable by lossy coding is very high, especially comparing it to the maximum possible ratio achieved with lossless procedures. In particular these techniques have two different targets: the *compressed size* and the *quality*: for the former the purpose is to find the minimum possible rate for a given distortion (maximum compression criteria), while for the latter the purpose is to minimize the distortion given the rate (maximum fidelity criteria). Examples of lossy coding can be found in the most diffused formats for video, images and audios, i.e. JPEG, MPEG, 3GPP and many others.

The main principle under lossy coding is the *quantization* operation, i.e. a process that maps symbols belonging to a larger alphabet into symbols belonging to a smaller one. The quantization can be either of single symbols (scalar quantization) or of many symbols at a time (vector quantization), which is convenient to exploit again the correlation between subsequent symbols and the structures present in the data.

However vector quantization is a complex procedure, requiring a significant amount of computational resources. Other techniques, which exploit the correlation between samples in a slightly different manner, result in a significantly less complex system. This is the case of *differential encoding techniques*, that transmit information by encoding differences, resulting in a very lightweight load. Among them we can identify the **Differential Pulse Code Modulation** (DPCM) coding technique explained in detail in section 2.1.

The purpose of this project is to present a system based on the DPCM coding scheme which exploits **Golomb codes** to compress the data, starting from many different 16-bits audio tracks. The approach followed is dealt in section 2, the results are presented in section 3 and finally the conclusions are drawn in section 4.

2 Technical Approach

In this section the main concepts and techniques exploited for the analyses are explained and a brief overview of the code implementation is presented.

2.1 DPCM coding scheme

The Differential encoding schemes exploit the correlation present among symbols of a sequence, and make a *prediction* on each sample based on its context (previous samples). What is encoded and transmitted are only the differences between the prediction and the sample value. The basic differential encoding system is known as the **Differential Pulse Code Modulation (DPCM)** system and it was developed at Bell Laboratories a few years after World War II. DPCM coding scheme is one of the most popular speech-encoding systems and is widely used in telephone communications.

Open Loop DPCM

Considering the sample output x_n, it is highly correlated with the context samples $x_{n-1}, x_{n-2}, ..., x_1$, therefore it is possible to make a prediction $p_n = f(x_{n-1}, x_{n-2}, ..., x_1)$ of x_n, which depends on the context. This way what is effectively coded is the difference $d_n = x_n - p_n$ between the sample and its prediction, which demands very little in terms of bits if compared with the original sample.

However, this coding technique is principally exploit in a lossy manner, i.e., starting from analog sources and performing a quantization operation. Therefore the problem of recovering an acceptable reproduction of the original sequence from the quantized difference value, arises. In fact supposing $d_n = x_n - p_n$, after the quantization the result is $\hat{d}_n = Q(d_n) = d_n + q_n$, where q_n is the quantization error introduced quantizing the n^{th} difference d_n. When the symbol is reconstructed at the receiver side, it reads $\hat{x}_n = x_n + \sum_{i=1}^{n} q_i$. This means that as much the sequence is long, as the error increases and the sample at the receiver will be different from the original one.

Closed Loop DPCM

Consequently this basic scheme (*Open Loop DPCM*) is replaced by a more efficient one (*Closed Loop DPCM*), which includes a *feedback loop* at the transmitter side. This allows to exploit the quantized samples $\hat{x}_{n-1}, \hat{x}_{n-2}, ..., \hat{x}_1$ to make the prediction both at the transmitter and at the receiver, avoiding this way the accumulation of the quantization error in the process.

The fundamental result of this coding procedure is that the maximum error introduced when reconstructing a sample is the error introduced by the quantization step of that specific sample difference, and thus depends *only on the quantizer*. In fact:

$$\left.\begin{aligned} d_n &= x_n - p_n \\ \hat{d}_n &= \hat{x}_n - p_n \end{aligned}\right\} \quad \Rightarrow \quad d_n - \hat{d}_n = x_n - \hat{x}_n \tag{1}$$

A block diagram of the differential encoding system described up to this point is shown in figure 1.

Predictor design

The goal of designing a good predictor is a requirement to minimize the distortion. In fact, how much the variance is reduced depends on how well the predictor can predict the next symbol based on the past reconstructed symbols.

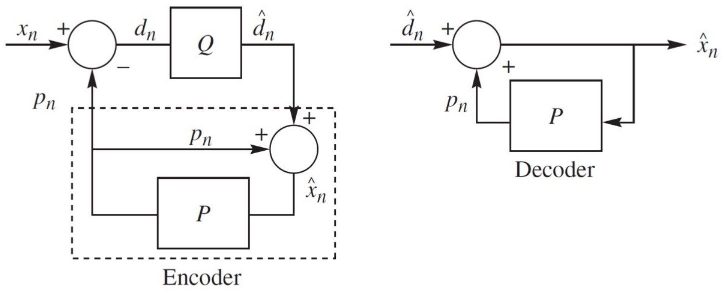


Figure 1: Block diagram of the Closed Loop DPCM scheme.

Considering the distortion with the *mean squared error*, it reads:

$$\sigma_d^2 = \mathbb{E}[d_n^2] = \mathbb{E}[(x_n - p_n)^2] \tag{2}$$

However, since the the predictor $p_n = f(\hat{x}_{n-1}, \hat{x}_{n-2}, ..., \hat{x}_1)$ depends on the reconstructed values $\{\hat{x}_i, i = 1, 2, ..., n-1\}$, it is a function of the quantization error which in turns depends on the variance σ_d^2. Hence minimizing the variance is difficult.

This leads to approximate the reconstructed values with the original ones, over the *fine quantization assumption*, i.e. assuming a small quantization step. Resulting in a predictor $p_n = f(x_{n-1}, x_{n-2}, ..., x_1)$, which in general can be assumed, even for infinite sequences, to depend on the previous N samples. It takes the name of **Nth-order predictor** and can be written as:

$$p_n = f(x_{n-1}, x_{n-2}, ..., x_{n-N}) \tag{3}$$

With this result, supposing the output of the source to be a stationary process, is possible to estimate this predictor, as it is given by the conditional expectation of the sample x_n, given the context, i.e.:

$$p_n = \mathbb{E}[x_n | x_{n-1}, x_{n-2}, ..., x_{n-N}] \tag{4}$$

Unfortunately, the assumption of stationarity is generally not true, and even if it were, finding this conditional expectation requires the knowledge of nth-order conditional probabilities, which would generally not be available. Therefore in many applications this problem is simplified a lot considering a **Nth-order linear predictor**, i.e., it is obtained as a linear combination of the context samples:

$$p_n = \sum_{i=1}^{N} a_i x_{n-i} \tag{5}$$

This predictor will be employed also for this project.

As a consequence, the purpose of minimizing the distortion σ_d^2, restricts to find the values of $\{a_i\}$ as follows:

$$\frac{\partial \sigma_d^2}{\partial a_k} = \frac{\partial}{\partial a_k}\mathbb{E}[d_n^2] = \frac{\partial}{\partial a_k}\mathbb{E}[(x_n - \sum_{i=1}^{N} a_i x_{n-i})^2] = -2\mathbb{E}[(x_n - \sum_{i=1}^{N} a_i x_{n-i})x_{n-k}] = 0, \quad k = 0, ..., N \tag{6}$$

The last equation in particular, states the *Orthogonality principle*, for which $\mathbb{E}[d_n x_{n-k}] = 0 \Rightarrow d_n \perp x_{n-k}$. Further developing equation 6, it is possible to derive the matrix form exploiting the *autocorrelation function* $R_{xx}(k) = \mathbb{E}[x_n x_{x-k}]$, since the source sequence is assumed as a realization of a real-valued wide-sense stationary process:

$$\begin{aligned} \mathbb{E}[(x_n - \sum_{i=1}^{N} a_i x_{n-i})x_{n-k}] = 0 \quad &\Rightarrow \quad \mathbb{E}[x_n x_{x-k}] = \sum_{i=1}^{N} a_i \cdot \mathbb{E}[x_{n-i} x_{x-k}] \\ &\Rightarrow \quad R_{xx}(k) = \sum_{i=1}^{N} a_i \cdot R_{xx}(k-i), \quad k = 0, ..., N \end{aligned} \tag{7}$$

That can be rewritten in matrix form as:

$$\underbrace{\begin{bmatrix} R_{xx}(0) & R_{xx}(1) & \dots & R_{xx}(N-1) \\ R_{xx}(1) & R_{xx}(0) & \dots & R_{xx}(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ R_{xx}(N-1) & R_{xx}(N-2) & \dots & R_{xx}(0) \end{bmatrix}}_{\boldsymbol{R}} \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix}}_{\boldsymbol{a}} = \underbrace{\begin{bmatrix} R_{xx}(1) \\ R_{xx}(2) \\ \vdots \\ R_{xx}(N) \end{bmatrix}}_{\boldsymbol{r}} \tag{8}$$

Leading to the final result, also known as the *Weiner-Hopf equations* in discrete time. Here $\boldsymbol{R}$ is a symmetric, non-singular semi-definite positive matrix; thus it is invertible, and the coefficients $\{a_i, i = 1, ..., N\}$ can be drawn as:

$$\boldsymbol{R} \cdot \boldsymbol{a} = \boldsymbol{r} \quad \Rightarrow \quad \boldsymbol{a} = \boldsymbol{R}^{-1} \cdot \boldsymbol{r} \tag{9}$$

Nevertheless, the data available in practice (hence also in this project) are real data distributed according to an unknown distribution function. Consequently the autocorrelation function can only be estimated. For M data points available, it is drawn from the following equation:

$$\tilde{R}_{xx}(k) = \frac{1}{M-k} \sum_{i=1}^{M-k} x_i x_{i+k}, \quad k = 0, ..., N \tag{10}$$

Quantizer design

As already stated, in a closed loop DPCM scheme, the error introduced on the compressed sequence depends only on the quantizer (equation 1). Consequently a good choice for the quantizer is extremely important to optimize the compression and make a trade-off between the compression ratio and the quantization error introduced.

To design a well-fitting system, the distribution of the coded data must be kept into account. As in the considered system the differences $\{d_i, i = 1, ..., N\}$ are coded instead of the original samples, their distribution results to be almost *Laplacian*, often highly peaked around the origin, due to the fact that the samples are correlated. The concentration of the distribution around the origin suggests a **variable length** code to be more efficient, coding lower-valued samples with longer codewords and viceversa.
This leads to design the quantizer keeping into account that, dealing with variable length codewords, the selection of the decision boundaries will affect the rate and the performances of the system. The simplest solution to adopt is to use the **Lloyd-Max quantizer**, which optimizes the thresholds and the reconstruction levels, and simply *entropy-code* the quantizer output.

On the other hand, a more complex and sophisticated approach that leads to even better results is to entropy-constraint the quantizer, in such a way that the minimization of the variance is constrained with the imposition that $H(Q) \approx R_0$, where R_0 is the rate and $H(Q)$ the entropy.
Fortunately, at higher rates it is possible to prove that the optimal quantizer is a *uniform quantizer*, simplifying the problem. Furthermore, it has been shown that the results also hold for lower rates.

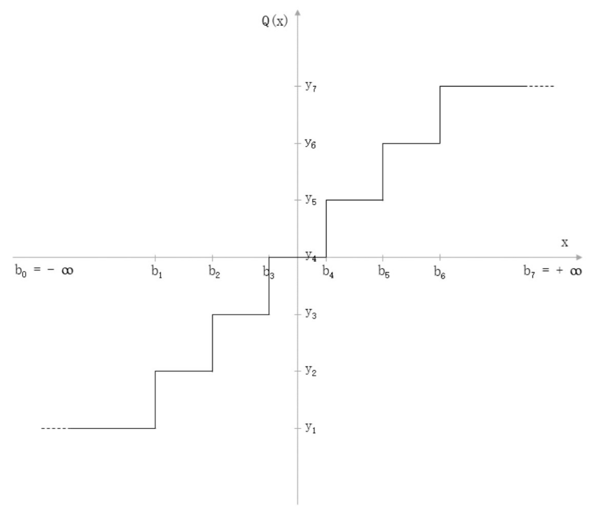


Figure 2: An example of Midtread Uniform Scalar Quantizer with $M = 7$ levels.

As a result, the quantizer chosen for this project is a *Midtread Uniform Scalar Quantizer* (figure 2). The choice is due to the fact that the Midtread Quantizer has an *odd number of levels*, allowing to code also the zero valued samples correspondent to silent periods in the audio tracks. The entropy coding in then achieved with Golomb codes, explained in detail in section 2.2.
As anticipated, the quantizer is uniform, thus both the intervals $\{b_i\}$ and the reconstruction levels $\{y_i\}$ with $i = 1, ..., M$ are equally spaced, and they are defined as:

$$b_i = \begin{cases} -\infty & i = 0 \\ b_1 & i = 1 \\ b_{i-1} + \Delta & i = 2, ..., M-1 \\ +\infty & i = M \end{cases} \qquad\qquad y_i = \begin{cases} b_1 - \frac{\Delta}{2} & i = 1 \\ b_{i-1} + \frac{\Delta}{2} & i = 2, ..., M \end{cases} \tag{11}$$

where Δ is the *quantization step*, defined for a source with output uniformly distributed as constant and belonging to the interval $[-X_{max}, X_{max}]$:

$$\Delta = \frac{2X_{max}}{M} \tag{12}$$

Since the quantizer is also Midtread, the number of levels is odd and given by $M = 2^b - 1$. Assuming the output of the source bounded in the interval $[-X_{max}, X_{max}]$, the *quantization rule* is the following:

$$Q(x) = \Delta \cdot \left\lfloor \frac{|x|}{\Delta} + \frac{1}{2} \right\rfloor \cdot sign(x) \tag{13}$$

where the notation $\lfloor\ \rfloor$ denotes the *floor function* and only the *granular region* is considered. The output of this quantizer is $\hat{d_n}$ the quantized prediction error.

2.2 Golomb codes

Golomb coding is a lossless data compression method using a family of data compression codes invented by Solomon W. Golomb in the 1960s. This family of codes is designed to encode integers with the assumption that the larger is an integer, the lower is its probability of occurrence, and this fits well variable length coding of distributions highly peaked around zero, like the Laplacian.

In addition, Golomb Codes are proven to be optimal prefix codes for alphabets following a *geometric distribution.* Considering this project's system, the quantized prediction error has to be entropy coded. If a map is used to make non-negative-valued the quantizer's output, the result is distributed almost as a Geometric Random Variable and this means that a Golomb Code is very well suited to accomplish the task of entropy coding.

The general procedure to obtain a Golomb Code is the described in the following. Let's consider a symbol $n \in \mathbb{N}_0 = \{0, 1, 2, 3, ...\}$ drawn from a Geometric Probability Mass function (pmf):

$$p_x(n) = p^n(1-p), \quad 0 < p < 1 \tag{14}$$

It can be written as $n = q \cdot m + r$, where:

- $q = \lfloor \frac{n}{m} \rfloor$ is the *quotient.* In Golomb codes it is coded with a **unary code**, i.e. a code where the symbol q is transmitted as q ones and the zero is used to end the sequence (e.g. if $q = 3$ the unary code is 1110).
- $r = n \mod m$, $0 \leq r < 1$ is the *remainder.* In Golomb codes it is coded with a **binary prefix code**, generally with variable-length codewords, that depends on the value of the parameter m. The procedure is the following:
  - if $m = 2^k$, it is the simplest case of **Golomb-Rice Codes**. Here r is coded using fixed-length code with $k = \log_2 m$ bits.
  - if $m \neq 2^k$, then the first r symbols, i.e. $0, 1, ..., 2^{\lfloor \log_2 m \rfloor} - m - 1$, are represented using the $\lfloor \log_2 m \rfloor$ binary representation. The other symbols, i.e. $2^{\lceil \log_2 m \rceil} - m, ..., m - 1$, are coded using $\lceil \log_2 m \rceil$ bits binary representation of $r + 2^{\lceil \log_2 m \rceil} - m$.

If we assume that $p^m = \frac{1}{2}$, then the pmf results:

$$p_x(k+m) = p^{m+k}(1-p) = \frac{1}{2} \cdot p^k(1-p) = \frac{1}{2} \cdot p_x(k) \tag{15}$$

Taking the $\log_2(\cdot)$ on both sides, the optimal length of $p_x(k+m)$ is obtained as the optimal length of $p_x(k) + 1$bit, so the code is optimal. Therefore what is to find is the parameter $m > 0$ for which p^m is close enough to $\frac{1}{2}$ optimizing the code lengths:

$$p^m = \frac{1}{2} \quad \Rightarrow \quad \log_2 p^m = -1 \quad \Rightarrow \quad m = \frac{-1}{\log_2 p} \tag{16}$$

Since in the most of the cases $p^m \neq \frac{1}{2}$, the parameter m is found in such a way that optimizes the code, drawing $p^m \approx \frac{1}{2}$. Therefore:

$$m = \left\lfloor \frac{-1}{\log_2 p} \right\rfloor \tag{17}$$

Once the parameter m and the pmf of the data are known at the receiver, the Golomb Code is easily decoded in a lossless way from r and q.

On the other hand, as the dealing is made with empirical data, also the parameter p requires to be estimated. The estimation can be made using the **Maximum Likelihood Estimation** (MLE) criterion, a method for estimating the parameters of a probability distribution by maximizing the likelihood function, i.e.:

$$\hat{p} = \arg\max_{p \in \Theta} \hat{L}_n(p; \boldsymbol{n}), \qquad \hat{L}_n(p; \boldsymbol{n}) = \prod_{i=1}^{N} p_x(n_i, p) \tag{18}$$

where $\boldsymbol{n} = \{0, ..., N\}$ and $\hat{L}_n(p; \boldsymbol{n})$ is the likelihood function, obtained as the product of the distribution $p_x(\cdot)$ computed on a single sample at a time.
The result of the estimation gives a value of p expressed by:

$$\hat{p} = \frac{\sum_{i=1}^{N} n_i}{N + \sum_{i=1}^{N} n_i} \tag{19}$$

As a consequence the optimal value for m is drawn and this leads to an optimal Golomb Code.

2.3 Implementation

The whole system is implemented in **Matlab**, and the code holds the following structure:

- `main.m` is the main executable file where the whole system is built. The system is composed of a 16-bit audio converter, a DPCM encoder, a quantizer, a Golomb encoder, a Golomb decoder and a DPCM decoder. First the audio is loaded and it is standardized to 16-bit CD quality. Then, a quantizer with `L` levels is built to be used later in the DPCM encoding system. The *autocorrelation matrix* $\boldsymbol{R}$ is estimated as well, and the coefficients of the linear predictor of order `n` are drawn. The input signal is fed to the `DPCM_encoder.m` which outputs the differences vector. This vector is mapped through the function `map.m` into positive numbers, to be further fed to `Golomb_encoder.m`. The result is a sequence of binary digits, ready to be transmitted and decoded at the receiver first using `demap.m`, then `Golomb_decoder.m` and finally extracting the result with `DPCM_decoder.m`. At the end the performances of the coding system are computed, in particular:
  - **the Rate (average codelength)** to measure the performances of Golomb code.
  - **the SNR**[1] to measure the performances of the Quantization.
  - **the SPER**[2] and the **prediction gain** to measure the performances of the predictor.

  But further details about the results are given in section 3.
- `DPCM_encoder.m` encodes the input signal using the DPCM technique. It relies on the quantizer built with `build_quantizer.m`.
- `DPCM_decoder.m` decodes the input signal using the DPCM technique.
- `build_quantizer.m` builds a uniform Midtread quantizer with `L` (odd) levels.
- `map.m` Performs a mapping of the difference vector obtained with the DPCM encoding, to a vector of positive numbers, in such a way that the positive values are mapped to odd numbers and the negative values to even ones, i.e. $f : \mathbb{Z} \to \mathbb{N}_0$ such that:

$$f : \{..., -2, -1, 0, 1, 2, ...\} = \boldsymbol{z} \mapsto f(\boldsymbol{z}) = \{..., 4, 2, 0, 1, 3, ...\} \tag{20}$$

- `demap.m` is the inverse function of `map.m`, and performs the inverse transformation $f^{-1} : \mathbb{N}_0 \to \mathbb{Z}$ such that:

$$f^{-1} : \{..., 4, 2, 0, 1, 3, ...\} = f(\boldsymbol{z}) \mapsto \boldsymbol{z} = \{..., -2, -1, 0, 1, 2, ...\} \tag{21}$$

- `Golomb_encoder.m` encodes with the Golomb encoding procedure the vector of integer numbers with the optimal value for m and gives a sequence of bits.

[1]signal to noise ratio
[2]signal to prediction error ratio

- `Golomb_decoder.m` decodes with the Golomb decoding procedure the sequence of bits and gives back the vector of integers.

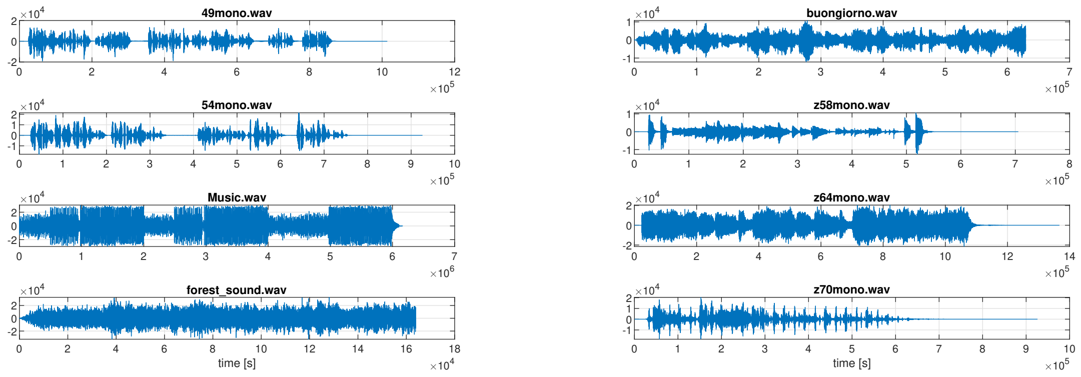


Figure 3: The input audio signals.

3 Results

In this section the results of the computation are presented; different kinds of signals are analyzed and compared. The implementation choices and the proofs are not given here, since they were deeply investigated in section 2.

3.1 Dataset

The dataset employed for the analyses is composed of several audio files (16bit, mono) in *.wav* format. These audios are of different nature, shape and pitch (figure 3). In particular both male and female voices are taken into account, as well as melodic sounds, music of different genre and songs with voice over music. The whole set of audio files is listed in table 1.

	audio signal	original title	audio description	F_s $[KHz]$
1	*female voice*	49mono	english speech of a woman.	44.1
2	*male voice*	54mono	german speech of a man.	44.1
3	*game music*	music	happy music *AshamaluevMusic, kids.*	44.1
4	*animals sound*	forest sound	sounds from different birds in the forest.	16
5	*buongiorno song*	buongiorno	song *Luciano Pavarotti, Buongiorno A Te.*	44.1
6	*guitar sound*	58mono	guitar solo *Zapateado, op. 23 No. 2.*	44.1
7	*opera music*	64mono	Opera *Carl Orff, Carminia Burana: O fortuna.*	44.1
8	*country song*	70mono	Song *Eddie Rabbitt, Early in the Morning*	44.1

Table 1: List of reference audio files and their specifics.

3.2 Signals and Features

Under a brief overview, the considered input signals present different features, that can be heard, and visualized in the temporal plot of figure 3. The vector of the differences $\hat{d}$, obtained through the DPCM encoding, is instead shown in figure 4 together with the original input for a reference audio; the residuals' samples are

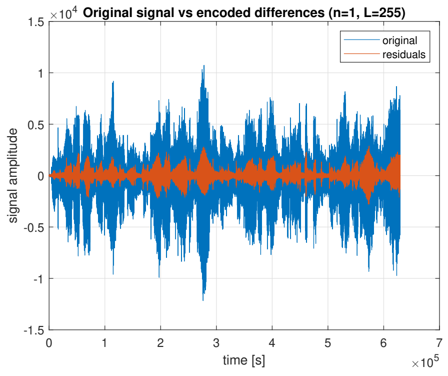
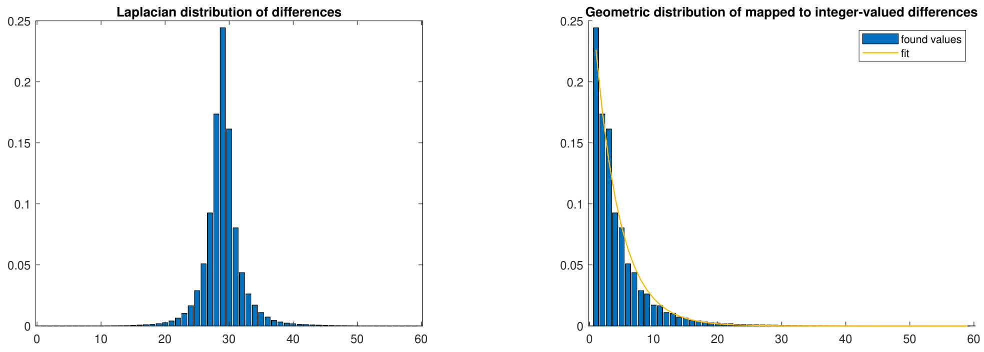


Figure 4: Original Signal *buongiorno* vs Residuals obtained after DPCM encoding.

considerably smaller-valued with respect to the original input vector. These samples are distributed with a Laplacian-like distribution, as shown in figure 5a. As far as they are mapped to positive values, the samples result being distributed according to a Geometric distribution. Figure 5b shows the mapped values together with the shape of a Geometric random variable of parameter $\hat{p}$, where $\hat{p}$ is estimated from the data with the maximum likelihood criterion (equation 18). It is possible to see that the analytical curve fits well the data and this will lead to an optimal estimation of m and thus to an optimal Golomb code.



Output of DPCM encoder for $L = 255$ and $n = 1$.

Output of mapping, before Golomb coding and fit of the curve.

Figure 5: The differences are distributed similarly to a Laplacian, while if mapped seem to fit well a Geometric distribution. The distributions reported here refer to *buongiorno* audio signal.

3.3 Performances

The performances of the system are described in the following. As anticipated, the metrics taken into consideration to measure the performances of the coding system, in relation with the parameters chosen, are presented below.

- **the Rate** is the average codelength of the transmitted bit sequence over the channel (output of Golomb code). It is defined as:

$$R = \frac{1}{M}\sum_{i=1}^{M} l_i \tag{22}$$

  where l_i is the codelength of the $i-th$ codeword in a sequence of M codewords.

- **the SNR** is the *signal to noise ratio* measurement, and it is needed to measure the performances of the system. It considers foremost the error that is introduced in the quantization step, in the DPCM encoder, and it is defined as:

$$SNR_{dB} = 10\log_{10}\left(\frac{\sum_{i=1}^{M} x_i^2}{\sum_{i=1}^{M}(x_i - \hat{x}_i)^2}\right) \tag{23}$$

- **the SPER** is the *signal to prediction error ratio* and it is generally used to measure the performances of the predictor. It is defined as:

$$SPER_{dB} = 10\log_{10}\left(\frac{\sum_{i=1}^{M} x_i^2}{\sum_{i=1}^{M}(x_i - p_i)^2}\right) \tag{24}$$

- **the prediction gain**, defined as:

$$G_{DPCM\_dB} = 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_d^2}\right) \tag{25}$$

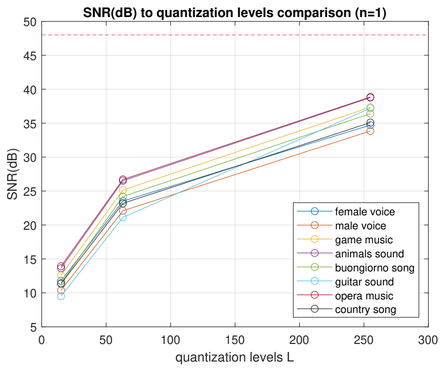
  where σ_x^2 is the input signal variance and σ_d^2 is the theoretic variance of the residuals.

In the following, the analysis is split to take into account separately each block of the system.

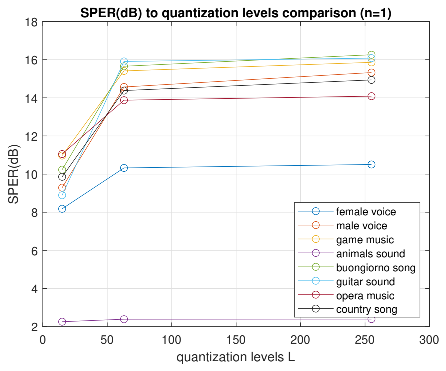
Quantization

The error that can be measured at the output of the system depends on the error introduced in the quantization (see equation 1) inside the DPCM encoder, and depends on the number of quantization values chosen. The distance between the output samples and the input ones grows as the number of quantization levels is reduced. A good measure that can be exploited in order to test the goodness of the quantizer is the SNR, as figure 6a shows. The red dotted line represents the upper bound for the SNR reachable with a uniformly distributed input and a uniform quantizer with $L = 255$ bits (that is 48.16dB). The last point for all the signals' curves, which correspond to the SNR value for $L = 255$, is sufficiently close to the boundary, hence the performances are to be considered acceptable.

Also the predictor is influenced by the number of quantization levels, and this can be seen in the SPER behaviour in figure 6b. The predictor is better with many quantization levels, but it tends to saturate after a sufficiently large value of L. The obtained results' behaviour agree with the reference example of [1].



SNR



SPER

Figure 6: SNR and SPER values to quantization levels for different audio files with $n = 1$.

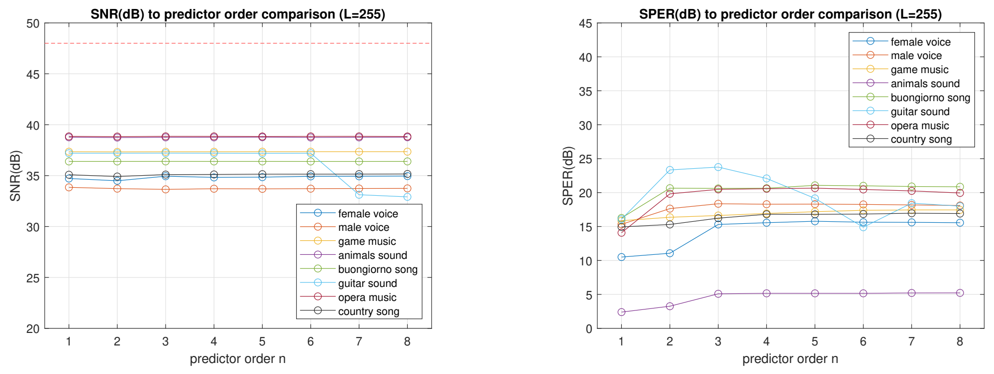
Prediction

The predictor's order n instead influences the system in a different way. Since the prediction is made considering the past n samples, the performances of the predictor result to be even better, as the considered n samples are correlated, and the SPER provides the most relevant measurement of its goodness.
Figure 7b shows the behaviour of the SPER for different signals. It is to notice that an increase in the order leads to better performances up to a certain value. In fact, the SPER improves until the past n considered samples are correlated with the currently considered one. When the threshold is overcome, the SPER tends to saturate. If the order is increased more, the samples are no longer correlated, and the performances worsen. Asymptotically, for $n \to \infty$, the process tends to be whitened. The results otained are comparable with the reference example of [1] and the SPER behaviours are similar to [4].

The SNR is not much affected, since the predictor affects first the values of d in the DPCM encoder, and then the value of $\hat{d}$ changes accordingly with the quantization. This way the SNR doesn't depend much on the prediction and remains almost stable for different predictor's orders.

Table 2 reports some additional results. The prediction goodness is confirmed by the variance ratio, which is, for every order of the predictor, very close to the theoretical value. A little decrease occurs if the predictor's order is increased, since the considered processes are theoretically stationary, but the real data available are not. In addition, also the prediction Gain is considered. It is possible to notice that it is in agreement with the SPER and rate behaviours for the most of the signals.

A curious exception is made by the *guitar sound* which holds a particular behaviour in both SNR and SPER diagrams. This is due to the accentuated nonstationarity of the signal, which can be clearly seen in worse values of variance ratio, as well as in the divergence between the theoretical gain and the empirical SPER value.



SNR SPER

Figure 7: SNR and SPER values to prediction orders for different audio files with $L = 255$.

Coding

To measure the performances of Golomb coding the rate can be employed, that is the average length of its output codewords. However, in this project the only parameter m of the Golomb coding system is fixed by the estimated optimal value of $\hat{p}$. Therefore the rate is not affected by Golomb coding, but only on the Golomb input $\hat{d}$ which depends on the DPCM encoder output, thus on the parameters L and n. This dependence is shown in figure 8.

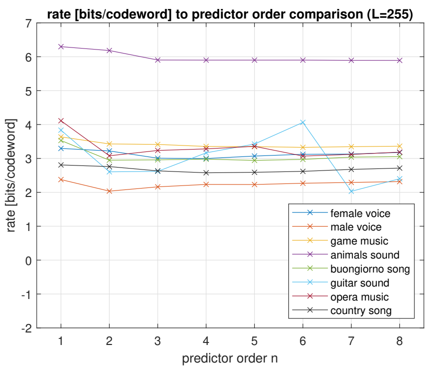
signal	L	n	$\sigma^2_{d\_model}/\sigma^2_d$	G_{DPCM} $[dB]$	$SPER_{dB}$	R [bits/codeword]
female voice	255	1	0.9963	10.5209	10.5045	3.2948
		2	0.9925	11.0988	11.0656	3.2206
		4	0.9932	16.0061	15.5632	2.9973
animals sound	255	1	0.9999	2.3966	2.3964	6.2974
		2	0.9996	3.2626	3.2609	6.1832
		4	0.9992	5.1647	5.1611	5.9006
buongiorno song	255	1	0.9908	16.2994	16.2594	3.5289
		2	0.8946	21.1323	20.6484	2.9484
		4	0.8637	21.2802	20.6439	2.9784
guitar sound	255	1	0.9742	17.4379	16.0807	3.8338
		2	0.9235	18.5043	23.3378	2.6049
		4	0.8152	19.4628	22.0825	3.1647

Table 2: performances of some sample signals with different behaviours.

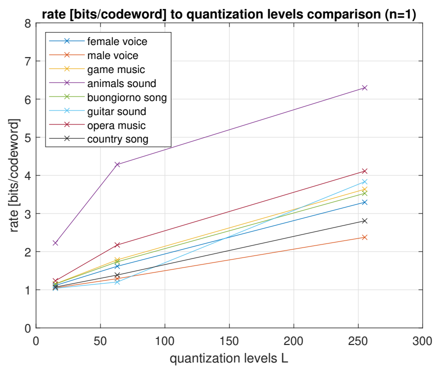
On the other hand, it is possible to test the optimality for the value of m found, making it varying and proving that the m found is the one that gives minimum rate, keeping the parameters L and n fixed. The result is shown in figure 9. In some cases with too low levels of quantization, the estimated probability $\hat{p}$ resulted too low to guarantee $m > 0$ and m was set to 1.

Figure 8b instead shows the relationship between the quantization levels and the rate. It is possible to see that if the number of quantization levels increases, the rate increases as well. This can be explained taking into account the fact that increasing L the symbols to be represented assume higher values. If there was not Golomb coding the result would be intuitive. On the other hand, also in this case the average codelength increases, since the distribution of the differences does not change in an extreme way varying L, and the small variations of the optimal value of m are increasing with L. This leads to a longer representation of the values of r and q on average.

Figure 8a shows the rate as a function of the predictor's order. It is possible to notice that the behaviour is in agreement with the SPER diagram of figure 7b. In fact in general an increase of the rate corresponds to a decrease in the SPER value (observe for example the behaviour of *opera music*, or the behaviour of *animals sound*). Like the SPER, the rate has a minimum and after a certain value, it does not improve anymore. This because if too high values of the predictor are considered, the samples are no longer correlated and the model is overfitted. The best predictor order is to be found in the minimum rate, and in the maximum SPER before saturation occurs.



rate-order



rate-levels

Figure 8: rate versus predictor order and quantization levels.

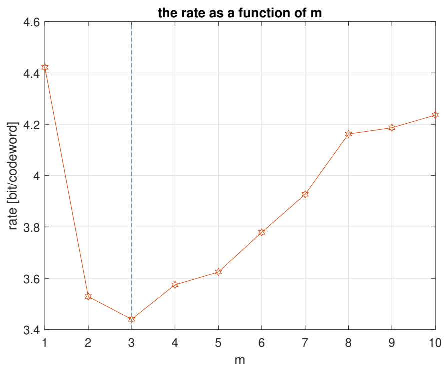


Figure 9: rate as a function of parameter m in the Golomb code. In this example (*buongiorno* audio signal with $n = 1, L = 255$) the optimal $m = 3$, drawn from $\hat{p} = 0.7738$ (see derivation in section 2), is the minimum.

Observations

From the results obtained it is possible to make some observations about the different behaviours of the signals. In general their behaviours are very similar one another, but an exception is made for a couple of them.

The behaviour of *animals sound* is peculiar, because it has very worst performances, despite a behaviour similar to the others. This is probably due to the fact that the signal is noisy, so it is almost stationary but not much correlated. This way the samples are hard to be predicted from the previous.

Some tests were led also with audio files representing a *bomb explosion* and a speech disturbed by *vuvuzela noise*, but the results obtained are not reported; from these analyses very bad results were obtained. This can be explained in the fact that also those kinds of signals are very noisy and unpredictable because of uncorrelation between samples. In addition, they are also nonstationary due to the oscillating behaviours of SNR and SPER.

4 Conclusions

In this paper a DPCM system, exploiting lossless Golomb coding, is explored. The system is analyzed from a theoretical point of view and the results of the Matlab implementation are investigated.

The relationships between the SNR, SPER and rate measurements suggest that higher level predictors, and quantizers with many quantization levels, lead to achieve the best performances. On the other hand an increase in the quantization levels means a decrease in the rate and longer length codewords to be transmitted over the channel. This way, a tradeoff between the SNR and rate guarantee the best performances for the system. In general also the prediction order is better to be kept low, in order to make easier the computation and achieve the same, or even better performances and avoid overfitting the data.

The issue that arose with the nonstationarity is an open problem. Probably it can be mitigated by applying some regularization to the autocorrelation matrix of the signals, or completely change the system structure and, in particular for speech coding, use an adaptive predictor.

References

[1] Kalid Sayhood, *Introduction to Data Compression.* 4th Edition, 2012.

[2] Giancarlo Calvagno, *Multimedia Coding, course notes*, 2020.

[3] Golomb Run-Length Encodings, 1966.

[4] Arun Kumar, Sandeep Kumar and Indra Narayan Kar, *Voiced speech synthesis using pitch asynchronous code excited linear filters for the glottal source.*