COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE
PAYS DE LA LOIRE – IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 648
*Sciences pour l'Ingénieur et le Numérique*
Spécialité : *Télécommunication*

Par
**Thibault PIANA**

## Étude et développement de techniques de compression pour les signaux de télécommunications en bande de base

Analysis and design of baseband signal compression methods for telecommunications

**Thèse présentée et soutenue à IMT Atlantique, Brest, le 04/10/2024**
**Unité de recherche : Lab-STICC & Space and Communication Lab**
**Thèse N° : 2024IMTA0424**

### Rapporteurs avant soutenance :

Bertrand LE GAL Maître de conférences HDR - ENSSAT
Damien ROQUE Professeur - ISAE Supaero

### Composition du Jury :

| | | |
|---|---|---|
| Président : | Matthieu CRUSSIERE | Professeur des Universités - INSA Rennes, IETR |
| Examinateurs : | Camille LEROUX | Maître de conférences HDR - INP Bordeaux, IMS Bordeaux |
| | Bertrand LE GAL | Maître de conférences HDR - ENSSAT |
| | Damien ROQUE | Professeur - ISAE Supaero |
| Dir. de thèse : | Abdeldjalil AÏSSA-EL-BEY | Professeur - IMT Atlantique, Lab-STICC |
| Co-dir. de thèse : | Matthieu ARZEL | Professeur - IMT Atlantique, Lab-STICC |
| Encadrant industriel : | Alain THOMAS | Ingénieur expert communications spatiales - Safran Data Systems |

### Invité(s) :

Abdel-Ouahab BOUDRAA Professeur des Universités - École Navale, Brest

*À tous ceux qui s'efforcent de repousser les frontières du savoir.*
*À ceux qui inspirent par leur quête inlassable de vérité.*
*À ceux qui cherchent, trouvent et partagent.*













Cette thèse et les travaux liés n'auraient jamais vu le jour sans la collaboration entre Safran Data Systems, le laboratoire Lab-STICC et l'IMT Atlantique, avec le soutien du dispositif CIFRE.

Ce cadre unique a permis de réunir les expertises industrielles et académiques nécessaires à la réalisation de ce projet.

# Remerciements

Je tiens tout d'abord à exprimer ma profonde gratitude à mes directeurs de thèse, Abdeljalil Aissa El Bey et Matthieu Arzel, dont l'encadrement, le soutien sans faille et la rigueur m'ont permis d'accomplir ce travail. Votre accompagnement et vos conseils ont été essentiels à chaque étape, et je ne pourrais jamais assez vous remercier pour cela. Un remerciement tout particulier va également à Alain Thomas, mon encadrant technique chez Safran Data Systems pour son expertise technique et sa capacité à toujours orienter mes efforts dans la bonne direction.

Je tiens également à remercier sincèrement les rapporteurs de cette thèse, Bertrand Le Gal et Damien Roque, pour leur lecture attentive du manuscrit et leurs remarques constructives, qui ont permis d'enrichir et d'affiner ce travail. Je joins à ces remerciements les deux autres membres de mon jury, Matthieu Crussière et Camille Leroux, pour leur intérêt et leur bienveillance. Je suis également reconnaissant envers Camille Leroux et Abdel Boudraa, membres de mon comité de suivi individuel (CSI), pour leurs conseils avisés et leur suivi attentif, qui m'ont permis de garder le cap tout au long de ces trois années.

Je remercie chaleureusement Vincent Gouldieff, pour sa bienveillance et son soutien constant ; sa rare perspicacité et son dévouement ont grandement contribué à la qualité de ces trois années. Ma reconnaissance va également à Emmanuelle Peter, directrice de l'innovation chez Safran, pour sa confiance et son implication à chaque moment nécessaire, et à l'ensemble des équipes de Safran Data Systems pour leur accueil et leur collaboration au cours de ces années. Vincent Desormeaux, mon manager chez Safran, pour son soutien et sa compréhension pendant ces trois années et également à Kerryan Mace, dont le travail sur l'intégration des réseaux de neurones a été d'une aide précieuse sur le chapitre consacré.

Un clin d'œil à Oussama Ait Sidi Ali, collègue doctorant chez Safran et ami, pour sa camaraderie et ses précieux conseils, et à Jean-Noël Bazin, pour son aide logistique et ses discussions passionnées.

Un grand merci aux membres du secrétariat MEE de l'IMT Atlantique, Aurélie Marcet et Cynthia Nougé, pour leur assistance efficace et leur aide précieuse dans toutes mes démarches administratives. Merci également à Yasmina Laurent et Delphine Lucas, de la direction de la recherche et de l'innovation, pour leur soutien logistique et leur efficacité.

Je souhaite aussi exprimer ma gratitude à mes parents et à ma soeur, pour avoir cru en moi depuis toujours, ainsi qu'à Aurore Cochard, ma conjointe, dont le soutien et la patience m'ont permis d'aller jusqu'au bout de cette aventure. Un merci tout particulier à Alan Gardin, fidèle ami depuis des années, et à Alexandre Grangier et Louis Chauvet, toujours présents dans les moments décisifs.

Je tiens enfin à remercier tous ceux qui ont marqué mon parcours en amont de cette thèse : Alain Terrier, professeur de mathématiques au Lycée des Pupilles de l'air et de l'espace, David Mentré, de Mitsubishi Electric, qui m'a encouragé à franchir le pas de la recherche, Damien Roque, qui m'a transmis sa passion pour les télécommunications, et Pierre Langlois, professeur à Polytechnique Montréal, qui m'a ouvert les portes de l'électronique numérique. Merci aux équipes de Widenorth, et en particulier à Helge, Hallvard et Christophe, pour leur confiance et leur motivation sans lesquelles je ne serais pas là aujourd'hui.

À tous ceux que je n'ai pas cités, amis, collègues, mentors et proches qui m'ont soutenu, conseillé, aidé ou simplement fait sourire durant ces années : Illyas, Hugo, Nil, Albane, Hector, Julien, Édouard, Alexandre, Mathieu, Albéric... et tant d'autres, sachez que vous avez contribué, chacun à votre manière, à la réussite de ce projet.

# ACKNOWLEDGMENTS

First and foremost, I would like to express my deep gratitude to my thesis supervisors, Abdeljalil Aissa El Bey and Matthieu Arzel, whose guidance, unwavering support, and rigor enabled me to complete this work. Your mentoring and advice were essential at every stage, and I can never thank you enough for that. A special thank you also goes to Alain Thomas, my technical supervisor at Safran Data Systems, for his technical expertise and his ability to consistently steer my efforts in the right direction.

I would also like to sincerely thank the reviewers of this thesis, Bertrand Le Gal and Damien Roque, for their careful reading of the manuscript and their constructive feedback, which helped to enrich and refine this work. I extend these thanks to the other members of my jury, Matthieu Crussière and Camille Leroux, for their interest and kindness. I am equally grateful to Camille Leroux and Abdel Boudraa, members of my individual monitoring committee (CSI), for their wise counsel and attentive follow-up, which helped me stay on course throughout these three years.

Warm thanks to Vincent Gouldieff for his kindness and constant support; his rare insight and dedication have greatly contributed to the quality of these three years. My gratitude also goes to Emmanuelle Peter, Director of Innovation at Safran, for her trust and involvement at every necessary moment, and to all the teams at Safran Data Systems for their welcome and collaboration over these years. My thanks also to Vincent Desormeaux, my manager at Safran, for his support and understanding throughout these three years, and to Kerryan Mace, whose work on neural network integration was a valuable contribution to the chapter dedicated to it.

A special nod to Oussama Ait Sidi Ali, fellow doctoral student at Safran and friend, for his camaraderie and valuable advice, and to Jean-Noël Bazin, for his logistical assistance and passionate discussions.

Many thanks to the MEE department secretariat at IMT Atlantique, Aurélie Marcet and Cynthia Nougé, for their efficient assistance and invaluable help with all my administrative tasks. I also thank Yasmina Laurent and Delphine Lucas, from the Directorate of Research and Innovation, for their logistical support and efficiency.

I would also like to express my gratitude to my parents and sister, for always believing in me, and to Aurore Cochard, my partner, whose support and patience helped me see this project through to the end. Special thanks to Alan Gardin, a loyal friend of many years, and to Alexandre Grangier and Louis Chauvet, always present in decisive moments.

Finally, I would like to thank all those who have shaped my path leading up to this thesis: Alain Terrier, mathematics professor at the Lycée des Pupilles de l'air et de l'espace, David Mentré from Mitsubishi Electric, who encouraged me to take the leap into research, Damien Roque, who instilled in me a passion for telecommunications, and Pierre Langlois, professor at Polytechnique Montréal, who introduced me to digital electronics. Thank you to the Widenorth teams, especially Helge, Hallvard, and Christophe, for their trust and motivation, without which I would not be here today.

To all those I have not mentioned by name, friends, colleagues, mentors, and loved ones who supported, advised, helped, or simply made me smile during these years—Illyas, Hugo, Nil, Albane, Hector, Julien, Édouard, Alexandre, Mathieu, Albéric... and so many others I may be forgetting—know that each of you contributed, in your own way, to the success of this project.

# Table of content

**Glossary** **17**

**Acronyms** **24**

**1 Introduction** **29**
1.1 Context and motivation . . . 29
1.1.1 Ground Station as a Service . . . 29
1.1.2 Emergence of Software-Defined Radio (SDR) . . . 30
1.1.3 RF over IP . . . 31
1.2 Problem statement . . . 33
1.3 Hypotheses & challenges . . . 35
1.3.1 Origins . . . 35
1.3.2 Challenges of lossless compression . . . 36
1.3.3 Challenges of lossy compression . . . 36
1.4 Achievements & innovations . . . 37
1.5 Methodology . . . 38
1.5.1 Notations and definitions . . . 38
1.5.1.1 Signal to noise ratio . . . 38
1.5.1.2 Bit error rate (BER) . . . 38
1.5.1.3 Degradation . . . 38
1.5.1.4 Compression ratio . . . 39
1.5.1.5 Expansion factor . . . 39
1.5.1.6 Time factor . . . 39
1.5.2 Data sources . . . 40
1.5.2.1 Synthesized signal data . . . 40
1.5.2.2 Real-life signal data . . . 40
1.5.3 The importance of shaping when compressing signals . . . 40
1.5.4 Techniques and tools . . . 42
1.6 Limitations and scope . . . 43
1.7 Structure of the document . . . 44

**2 State-of-the-art** **45**
2.1 Entropy coding . . . 46
2.1.1 General algorithmic principles . . . 46
2.1.2 Universal codes . . . 47
2.2 Dictionary compression . . . 49
2.3 Quantization based compression . . . 50
2.3.1 Uniform quantization . . . 50
2.3.2 Non-uniform & vector quantization . . . 51
2.3.3 Adaptive quantization . . . 51
2.3.4 Quantization of complex-valued data . . . 51
2.3.5 Companding techniques: $\mu$-law & A-law . . . 52
2.4 Audio compression methods . . . 53
2.4.1 Lossless compression techniques . . . 53
2.4.2 Lossy compression techniques . . . 55
2.5 Predictive coding . . . 56
2.5.1 Background . . . 56
2.5.2 Formal approach . . . 56
2.5.3 Linear predictive coding (LPC) . . . 57
2.6 Emerging techniques . . . 58
2.6.1 Sparse coding . . . 58

2.6.2 Compressed sensing . . . 59
2.6.2.1 Presentation . . . 59
2.6.2.2 Measurement requirements in compressed sensing . . . 59
2.7 Neural-network based compression for digital communication signals . . . 61
2.8 Upcoming challenges . . . 61

**I Compression based on signal processing & analysis 63**

**3 Quantization 64**
3.1 Essential insights and familiar concepts . . . 64
3.2 Uniform quantization . . . 65
3.2.1 Basic principles and implementation . . . 65
3.2.1.1 Quantization . . . 65
3.2.1.2 Requantization . . . 65
3.2.2 Bit depth and dynamic range: the AGC dilemma . . . 65
3.2.3 Effectiveness with different signal types . . . 67
3.2.4 Comparative summary . . . 68
3.3 Non-uniform & vector quantization . . . 69
3.3.1 Codebooks . . . 69
3.3.2 Algorithmic codebook construction . . . 70
3.3.2.1 The LLoyd-max algorithm . . . 70
3.3.2.2 Linde-Buzo-Gray algorithm . . . 73
3.3.3 Practical implementation . . . 73
3.3.3.1 Non-uniform quantization of noisy signals . . . 73
3.3.3.2 Residual entropy & effective compression . . . 75
3.4 Conclusion . . . 76

**4 Sampling methods 77**
4.1 From one sample to another . . . 77
4.1.1 Up-sampling & downsampling . . . 77
4.1.2 Baseband signal sampling . . . 78
4.1.2.1 Methods & limitations . . . 79
4.1.3 Application to digital communications . . . 80
4.1.3.1 Minimum sampling rate . . . 80
4.1.3.2 Joint coding . . . 82
4.2 Discussion . . . 84
4.2.1 About uniform sampling . . . 84
4.2.2 About non-uniform sampling . . . 84
4.3 Conclusion & actual usage . . . 85

**II Compression by pattern recognition & dictionary construction 87**

**5 Dictionary-based lossless compression 88**
5.1 Introduction & overview . . . 88
5.1.1 LZ algorithms & their derivatives . . . 88
5.1.2 Scope and significance in telecommunications . . . 89
5.2 Methodology . . . 89
5.2.1 Simulation environment . . . 89
5.2.1.1 Signal generation and noise addition . . . 89
5.2.1.2 Algorithms implementation . . . 89
5.2.1.3 Data representation . . . 90
5.2.2 Influence of signal parameters . . . 91
5.2.2.1 Quantization depth $N_q$ . . . 91
5.2.2.2 Number of samples per symbol $\mathcal{F}$ . . . 91
5.2.3 A word about binary data representation . . . 93
5.3 Results . . . 94
5.3.1 Performance report . . . 94
5.3.1.1 Compression ratio . . . 94
5.3.1.2 Total processing time . . . 95
5.3.1.3 Time factor . . . 96
5.3.2 Discussions . . . 97

5.3.2.1 NRZ/RZ shaped signals . . . 97
5.3.2.2 RRC shaped signals . . . 97
5.3.2.3 Gaussian shaped signals (GMSK like) . . . 97
5.4 Summary of key findings & conclusion . . . 98

**6 Sparse coding & sparse dictionary learning 99**
6.1 Presentation . . . 99
6.1.1 Introduction to sparse coding . . . 99
6.2 Dictionary creation . . . 100
6.2.1 Exposition & pertinence . . . 100
6.2.2 Data preprocessing . . . 101
6.2.3 Choice of learning algorithm . . . 102
6.3 Sparse coefficients: analysis & compression . . . 103
6.3.1 One problem for three equations . . . 103
6.3.2 Managing loss with sparse coding . . . 104
6.3.3 Sending sparse data over the network . . . 105
6.3.3.1 Conventional versus novel approaches . . . 105
6.3.3.2 An optimized way to send nonzero values positions . . . 105
6.4 Compression chain . . . 107
6.5 Simulations . . . 109
6.6 Results & discussions . . . 110
6.6.1 Algorithms based on $\ell_0$ norm . . . 110
6.6.2 Alternatives & exotic algorithms . . . 112
6.6.2.1 The ridge regression . . . 112
6.6.2.2 About compressed sensing . . . 113
6.6.3 A word about hardware implementation . . . 115
6.7 Conclusion . . . 116

**III Compression by partial matching & predictive coding 117**

**7 Predictive coding & PPM 118**
7.1 Related work . . . 118
7.2 Linear prediction . . . 119
7.2.1 Mathematical model . . . 119
7.2.2 LPC based predictors . . . 120
7.2.2.1 Presentation & mathematical background . . . 120
7.2.2.2 Coefficients determination . . . 120
7.2.2.3 Effective lossless compression using LPC . . . 123
7.3 Tests & usage . . . 124
7.3.1 Usage of FLAC codec . . . 124
7.3.2 LPC vs FLAC . . . 126
7.4 Limitations . . . 127
7.5 Conclusion . . . 129

**8 Non-linear PPM compression 131**
8.1 Motivations . . . 131
8.2 Choice of the model . . . 132
8.2.1 Taylor series model . . . 132
8.2.2 Volterra model . . . 132
8.2.2.1 Presentation . . . 132
8.2.2.2 Volterra decomposition as ideal predictor . . . 133
8.2.2.3 Limitations . . . 133
8.2.3 Deep neural models to the rescue . . . 135
8.3 Neural Network-based predictors . . . 136
8.3.1 General approach . . . 136
8.3.2 Simulations setup & training methodology . . . 138
8.3.3 A word about resources usage . . . 139
8.4 Compression on noiseless signals: TX channel . . . 140
8.4.1 Standalone modulations . . . 140
8.4.2 DVB-S2 and multi-modulation Standards . . . 142
8.4.3 Discussion . . . 144
8.4.4 Selection of hyper-parameters . . . 145
8.4.4.1 Size and number of layers . . . 145
8.4.4.2 Data complexity . . . 145

8.4.4.3 Window width W . . . 145
8.4.4.4 Cost function . . . 145
8.5 Compression in presence of noise: RX channel . . . 147
8.5.1 Experimental setup . . . 147
8.5.2 Results and discussion . . . 147
8.6 Implementation & optimization . . . 150
8.6.1 Deploying neural networks for high throughput applications . . . 150
8.6.2 Pruning & quantization . . . 151
8.6.2.1 Understanding pruning . . . 151
8.6.2.2 Quantization overview . . . 152
8.6.2.3 Importance in real implementations . . . 152
8.6.3 Implementing neural networks on FPGA . . . 153
8.6.4 Results for an FPGA implementation of the shallow network . . . 153
8.7 Conclusion and future perspectives . . . 156

**9 Conclusion** **157**
9.1 Restatement of research objectives . . . 157
9.1.1 Advancements in lossless compression . . . 157
9.1.2 Advancements in lossy compression . . . 158
9.1.3 Results on practical case . . . 159
9.2 Study limitations . . . 160
9.3 Recommendations for future research . . . 161
9.3.1 Broader signal types . . . 161
9.3.2 Hardware implementations . . . 161
9.3.3 Algorithm optimization . . . 161
9.3.4 Cross-layer optimization . . . 161
9.3.5 Adaptive and intelligent compression techniques . . . 162
9.3.6 Extending techniques beyond compression . . . 162
9.4 Going further: new and emerging methods . . . 163
9.4.1 Compression based on autoencoder networks . . . 163
9.4.2 Generative model-based compression . . . 164
9.4.3 Vector quantization via Neural Networks . . . 164
9.4.3.1 Overview & emerging methods . . . 164
9.4.3.2 Specific compression methods using neural networks . . . 166
9.5 Overall conclusion & final thoughts . . . 169

**Bibliography** **171**

**IV Appendices** **183**

**A Average compression of Rice coding & ideal value of Rice parameter** **184**
A.1 Introduction . . . 184
A.2 Normal & half-normal distribution . . . 184
A.3 Investigation of the residual distribution . . . 185
A.4 Rice coding . . . 186
A.4.1 Normal distribution case . . . 186
A.4.1.1 $p_1$ estimation . . . 186
A.4.1.2 $P_k$ estimation . . . 186
A.4.1.3 Estimation of the expectation of the reduced random variable . . . 188
A.4.1.4 Optimal Rice parameter . . . 188
A.4.2 Conclusion on Rice coding efficiency for Gaussian distribution . . . 189

**B Algorithm decision tree** **190**

**C Compressed sensing architecture** **191**

**D Sparsity compression chain** **192**

**E Predictive compression chain** **193**

**Résumé en Français** **195**

# List of Figures

1.1 Space communication systems.
**Source:** *Generated by DALL-E* . . . 29
1.2 Overview of remote/centralized modem processing architectures used in satellite communications systems. Compressing the digitized raw signals is the key to limit the congestion with an affordable network infrastructure. . . . 34
1.3 Evolution of related projects at Safran Data Systems . . . 35
1.4 Degradation measurement (in $dB$) . . . 38
1.5 MATLAB®, created by MATHWORKS®, is a commercial programming language with multiple paradigms and a numerical computing environment. It enables manipulation of matrices, plotting of functions and data, algorithm implementation, development of user interfaces, and interaction with programs in different languages. . . . 40
1.6 A Nuron Engine architecture.
**Source:** *Safran Data Systems* . . . 40
1.7 Visualization of various shaping for a dataset represented on a Binary Phase Shift Keying (BPSK) constellation. . . . 41

2.1 Logo of gZIP. gZIP is based on the deflate algorithm, which is a combination of the LZ77 and Huffman algorithms . . . 49
2.2 Illustrations of 4-bit uniform quantization of a signal. . . . 50
2.3 Illustrations of 3-bit nonuniform vector quantization of a signal. A codebook has been precomputed using the LBG algorithm [47]. . . . 51
2.4 Logo of FLAC audio codec. The FLAC codec is a lossless audio codec that uses predictive LPC coding [48] combined with a Rice entropy coder [38] to compress the audio stream.
**Source:** *xiph.org. Open-source image* . . . 53
2.5 Spectrogram comparison of the same music track encoded in FLAC (top) and MP3 at 48 kbps (bottom). The MP3 spectrogram reveals a significant reduction in frequency range and detail, particularly above 5 kHz, illustrating the effects of lossy compression at a lower bitrate. . . . 55
2.6 Estimated entropy of the original signal and its residual, computed with an LPC-based predictor. In this simulation, each signal sample is encoded using 16 bits, and all signals are transmitted using $\mathcal{F} = 2$ samples per symbol. The analysis is performed using MATLAB®. 57
2.7 Evolution of the estimator 2.13 depending on the number of nonzeros coefficients. $K = 256$. 60

3.1 Quantization of a 8-PSK | $\text{RRC}_{0.25}$ signal with a signal-to-noise ratio of $\mathtt{E_s}/\mathtt{N}_0 = 10$ dB using three different Automatic gain control (AGC) techniques on an 8-bit system. The first AGC approach causes signal saturation, leading to significant signal loss (EF $> 1$). In contrast, the second AGC method accurately matches the dynamic range of the signal (EF $= 1$). The final one does not use the complete range of available values, resulting in the waste of certain quantization levels that are never utilized (EF $< 1$). . . . 66
3.2 Evolution of Effective Range of a signal depending on AGC level and $\mathtt{E_s}/\mathtt{N}_0$. –Saturated at 1–. . . . 66
3.3 Quantization depth and its effect on degradation for **DVB-S2 modcods**. This evaluation is carried out on a pure Gaussian channel without any synchronization processes, employing a ($\text{RRC}_{0.25}$) filter. . . . 67
3.4 Convergence of Lloyd-max algorithm according to waveform distribution after shaping. . . . 72
3.5 Distinct codebook based on the LBG algorithm. The resulting centroids are the centroids of a Voronoi diagram that align with the edges of the codebook. . . . 74
3.6 Evolution of a BPSK waveform histogram, shapped with $\text{RRC}_{0.25}$, depending on signal-to-noise ratio value. . . . 75

4.1 Effect of upsampling and downsampling on a signal (only real part is shown). The signal is a 16-APSK ($\text{RRC}_{0.25}$) modulated signal with an arbitrary initial sampling frequency. Resampling is performed using a FFT resampler. . . . 78
4.2 Evolution of signal histogram depending on the number of sample per symbol. Resampling has been performed using a FFT resampler. . . . 79
4.3 Up & Down sampling process using a FIR filter. In MATLAB®, this kind of filter can be applied using *upfirdn* function . . . 79
4.4 Operation of double resampling of a 8-PSK | $\text{RRC}_{0.25}$ signal from $\mathcal{F} = 16$ to $\mathcal{F} = \beta$, then back to $\mathcal{F} = 16$ using a polyphase filter with $N$ coefficients. The lower $\beta$, the more coefficients are needed. Note that the filter is designed using the *firls* function from MATLAB®. 81
4.5 Sampling rate and its effect on degradation for **DVB-S2 modcods**. This evaluation is carried out on a pure Gaussian channel without any synchronization processes, employing a ($\text{RRC}_{0.25}$) filter. . . . 83

5.1 Illustration of the Lempel-Ziv compression process. The input data are scanned, and recurring patterns are replaced with references to a dictionary. This process results in a compressed output that can be efficiently decompressed using the same dictionary. . . . 88
5.2 Compression ratio evolution for different waveforms with different numbers of quantization bits (from 6 to 12 bits per sample). . . . 91
5.3 Compression efficiency for different waveforms with different numbers of samples per symbol. Compression efficiency, within a parameter range, is determined by the ratio of the actual compression ratio to the maximum achievable compression ratio within the specified parameter range. . . . 92
5.4 Different representations of complex data. Symbol < means that samples are stored in *little endian* format, and symbol > means that samples are stored in *big-endian* format . 93
5.5 Compression ratio $\mathcal{C}$ depending on the chosen representation & SNR . . . 93
5.6 Evolution of compression ratio $\mathcal{C}$ –as defined in Section 1.5.1.4–for different waveforms and value of $\mathtt{E_b/N_0}$ in dB. All signals are sampled on 12 bits and 2 samples per symbol. . . . 94
5.7 Total processing time for different waveforms and value of $\mathtt{E_b/N_0}$ in dB. All signals are sampled on 12 bits and 2 samples per symbol. . . . 95
5.8 Evolution of Time factor $\mathcal{T}_F$ –as defined in Section 1.5.1.6–for different waveforms and value of $\mathtt{E_b/N_0}$ in dB. All signals are sampled on 12 bits and 2 samples per symbol. . . . 96

6.1 The simple case of a sinusoidal signal $\boldsymbol{y}$ represented as a sparse combination of atoms in a dictionary $\boldsymbol{\mathcal{D}}$ where $\boldsymbol{\mathcal{D}}$ is the Fourier basis. . . . 100
6.2 Categorization of recovery algorithms.
**Source:** *Adapted from [114]* . . . 100
6.3 Example of signal approximation using ODL with OMP and a parameter $\tau_0 = 12$ on a 8-PSK signal sampled at 120 samples per symbols. The atoms of the dictionary and the associated coefficients are represented. . . . 101
6.4 Convergence and calculation time of learning algorithms depending on problem conditions & parameters. Dictionaries are trained on a noiseless 8-PSK | $\text{RRC}_{0.25}$ signal sampled at 2 samples per symbol. . . . 102
6.5 Changes in sparsity parameters based on the selected formulation and parameter. Dictionaries are trained on a noiseless 8-PSK | $\text{RRC}_{0.25}$ signal sampled at 2 samples per symbol. 103
6.6 Two ways to send nonzeros positions when the number of nonzero values is known. . . . 105
6.7 Evolution of the number of bits to send for each channel in order to transmit the position of nonzeros values to the decompressor ($K = 16$). zSTD could actually be replaced by another compression mechanism (gZIP, LZMA,...). . . . 106
6.8 Proposed comprehensive compression and decompression chain for signal compression using sparse coding . . . 107
6.9 Expected compression ratio for each degradation. Simulations were performed for an 8-PSK signal with a 5/6 LDPC FEC. . . . 110
6.10 Effect of parameters on compression ratio & degradation for a 8-PSK signal using **SOC** optimization method . . . 111
6.11 Determination of optimal subspace of parameters on compression ratio & degradation for a 8-PSK signal using **O** optimization method . . . 111

7.1 Classical architecture for compression/decompression schema from a remote modulation unit to an antenna using predictors. Entropy coding is usually performed by a Golomb-Rice encoder. . . . 119

7.2 Linear 1-layer neural network without activation function . . . 121
7.3 Predictor (compressor) using LPC coefficients. The implementation of this kind of predictor actually has the architecture of a FIR filter . . . 123
7.4 Evolution of compression ratio $\mathcal{C}$ for various algorithms, waveforms, and SNR conditions . 124
7.5 Evolution of time factor $\mathcal{T}_F$ for various algorithms, waveforms, and SNR conditions . . . . 125
7.6 Evolution of compression ratio $\mathcal{C}$ for various algorithms, waveforms, and SNR conditions. –If a curve is missing, it means that the algorithm compression ratio is too low to be displayed.– . . . 126
7.7 Histograms of noiseless QPSK signals and their prediction using LPC predictors. $\mathcal{F}$ indicate the number of samples per symbol (also called the oversampling factor). The signals are shaped with a 0.25 roll-off resampled from a unique source using polyphase resampling. . 127
7.8 Histograms of noiseless 8-PSK signals and their prediction using LPC predictors. . . . . 127
7.9 Evolution of global compression ratio when a re-sampling followed by a compression using LPC is performed. The original signal oversampling is $\mathcal{F} = 2$, it is a 8-PSK signal generated by a *Safran NuRon Engine* modulator, shaped using a 0.25 roll-off RRC that allows a theoretical minimum for $\mathcal{F}$ of 1.25 samples per symbol. We note that the compression efficacy of LPC decreases as the number of samples per symbol is reduced. . . . 128

8.1 Simple 3-Layer neural network with a tanh activation function (NAR-tanh). This network is equivalent to an approximated infinite Volterra decomposition according to Wray & Green. 135
8.2 Shallow network architecture: A simple and robust network that outperforms LPC [165], which can be easily trained on a conventional CPU and has a very high inference rate. This allows compression at low latency and high throughput. . . . 136
8.3 Schematic representation of the neural network training pipeline, from dataset preparation to model evaluation using a Rice coder. . . . 138
8.4 Histograms of QPSK signal and its prediction using Neural predictors. $\mathcal{F} = 1.25$ for both figures. Predictions still follow the signal histogram for low values of oversampling. . . . 140
8.5 Compression performance of the combination of resampling with lossless techniques applied on QPSK and 8-PSK (RRC 0.25) signals. . . . 141
8.6 Structure of a DVB-S2 frame [168]–[170]. . . . 142
8.7 Evolution of global compression ratio for each DVB-S2 modcod. We observe that the compression ratio for lossless compression does not depend on the FEC used but only on the modulation index. Note that 37.5% of the compression ratio is due to the decimation / resampling step, the rest comes from our compressor. . . . 142
8.8 Each signal $m_i(t)$ of each modcod $i$ has been normalized to the same power ($\sigma^2 = 1$). But their amplitude range is different. When transmitting a DVB-S2 signal, a lower modulation index leads to a large non-used amplitude range that is easily compressed by the entropy coder . . . 143
8.9 Evolution of global compression ratio depending on neurons on hidden layer for QPSK and 8-PSK modulated signals. Studied for Shallow network. . . . 145
8.10 Evolution of global compression ratio depending on window size W for Shallow network. . 145
8.11 Impact of the cost function on the compression ratio. . . . 146
8.12 Relative efficiency over $\ell_1$ . . . 146
8.13 Compression ratio obtained for various signals and SNRs. All signals have an original sampling rate of 2 samples per symbol. Simulations showed that performing a resampling before applying lossless compression increases the global compression ratio. NNs were specifically trained for successive SNR ranges. The parameter mentioned for each network is the tuple $(\mathrm{W}, N)$. . . . 148
8.14 Evolution of compression ratio depending on *Pruning factor* for two networks trained on QPSK signals with $\mathrm{RRC}_{0.25}$ shaping on **noiseless data**. **Compression ratio includes resampling step from $\mathcal{F} = 2.00$ to $\mathcal{F} = 1.25$** . . . 154

9.1 A streamlined decision tree employed to select the appropriate algorithm family for signal compression. Larger version of this schematic is available in Appendix B. . . . 158
9.2 Structure of an autoencoder network. . . . 163
9.3 Architecture of SoundStream
**Source:** *SoundStream Github Project [190]* . . . 166
9.4 Architecture of Encodec
**Source:** *Encodec Github Project [191], MIT licensed* . . . 167

A.1 Value of $A_k^{\mathcal{R}_\mathrm{P}}$ depending on $\sigma_\mathcal{R}$ and $k$. . . . 187

A.2 Difference between our estimated value of $\mathbb{E}\left(\frac{\mathcal{R}_{\mathrm{P}}}{2^k}\right)$ and its actual value. In this context, $\sigma_{\mathcal{R}} = 10^6$ . . . . . 188
A.3 Computed vs optimal value of Rice parameter $k$ for a Gaussian distribution of standard deviation $\sigma$. . . . . 188

B.1 A streamlined decision tree employed to select the appropriate algorithm family for signal compression. . . . . 190

C.1 Proposed compression chain for signal compression using *compressed sensing* . . . . . 191

D.1 Proposed comprehensive compression chain for signal compression using sparse coding . . 192

E.1 Classical architecture for compression/decompression schema from a remote modulation unit to an antenna using predictors. Entropy coding is usually performed by a Golomb-Rice encoder (Zoomed). . . . . 193
E.2 Architecture d'un réseau te type C-RAN pour le segment sol satellitaire: le couple modulateur-démodulateur (modem) est déplacé du pied d'antenne dans une localisation déportée. Il est alors nécessaire de faire transiter les échantillons bruts entre l'antenne et le modem en utilisant un réseau fibré de haute bande passante. . . . . 196

# List of Tables

2.1 Huffman encoding dictionary of Gaussian Data between −5 and 5 . . . . . 46
2.2 Comparison between Huffman Coding and Arithmetic Coding . . . . . 47

5.1 List of some efficient dictionary compression algorithms . . . . . 90

6.1 Simulation parameters . . . . . 109

8.1 Volterra decomposition for different orders. . . . . 133
8.2 Number of parameters for each network, without any optimization step (weight quantization, node pruning or network compression). The training window is W = 24. The given number includes the bias and the weights. . . . . 139
8.3 Resources usage on FPGA depending on pruning ratio parameters. **Quantization of network weights is 8 bits** Pruning and implementation has been comped with *Xilinx FINN*. . . . . 154

9.1 Results of compression for the different compression algorithms presented, implemented and tested in this manuscript. . . . . 159

# GLOSSARY

**(Square)-Root-raised-cosine filter** A type of filter used in digital signal processing to shape the waveform of a digital signal, thereby reducing inter-symbol interference. It is particularly used in modulated data transmission..

**Accelerated K-Singular Value Decomposition** A fast variant of the K-SVD algorithm used for dictionary learning, where the decomposition process is accelerated by leveraging optimizations in the singular value decomposition steps..

**Adaptive Coding and Modulation** An advanced technique that dynamically adjusts coding and modulation parameters in real time, leveraging feedback from the channel to maximize throughput and reliability..

**Additive White Gaussian Noise** A basic noise model used in information theory to mimic the effect of many random processes that occur in nature. The noise is characterized by a constant spectral density across all frequencies (white noise) and a normal distribution of amplitude (Gaussian). AWGN is commonly used in the analysis and simulation of communication systems to represent the impact of noise on signal transmission..

**Akaike Information Criterion** A statistical measure used to compare models by balancing the goodness of fit and model complexity. It is calculated as $\mathrm{AIC} = 2k - 2\ln(L)$, where $k$ is the number of parameters and $L$ is the likelihood of the model..

**Amplitude Phase Shift Keying** A digital modulation technique that combines both amplitude and phase shift keying to improve spectral efficiency and performance in the presence of noise. APSK is used in advanced communication systems, such as satellite communication and high-speed data links, to achieve higher data rates and better signal quality..

**Analog-to-Digital Converter** A device or circuit that converts an analog signal, such as voltage, into a digital representation, enabling processing by digital systems..

**Application-Specific Integrated Circuit** A custom-designed integrated circuit created for a specific application or function, often optimized for performance, power efficiency, and cost in high-volume production..

**Artificial Intelligence** The simulation of human intelligence processes by machines, especially computer systems. This includes learning, reasoning, and self-correction..

**Automatic gain control** A system in electronic devices that automatically adjusts the gain to maintain a constant output level despite varying input signal levels. It is essential in receivers to cope with fluctuating signal strengths..

**Autoregressive model** A type of random process used in statistical analysis where future values are regressed on their own previous values. It is used for time-series analysis and forecasting..

**Autoregressive Moving-Average model** A combination of autoregressive and moving-average models used for analyzing and forecasting time series data. It captures both the autoregressive and the moving average components of the data..

**Basis Pursuit** An optimization technique for decomposing a signal into an optimal superposition of dictionary elements by solving a linear programming problem..

**Bayesian Information Criterion** A criterion for model selection based on the likelihood function and model complexity. It is calculated as $\text{BIC} = k\ln(n) - 2\ln(L)$, where $k$ is the number of parameters, $n$ is the number of observations, and $L$ is the likelihood of the model..

**Belief Propagation** An iterative algorithm used for performing inference on graphical models, especially in decoding LDPC codes and Bayesian networks..

**Bit-error rate** The number of bit errors divided by the total number of bits transmitted, received, or processed over a given time period. It is a key parameter in assessing the performance of a digital communication system..

**Central Processing Unit** The primary processing unit in a computer, responsible for executing instructions and performing arithmetic, logic, control, and input/output operations specified by the instructions..

**Centralized Radio Access Network** A centralized, cloud computing-based architecture for radio access networks that supports advanced features such as centralized processing, real-time virtualization, and cooperative radio technologies. C-RAN decouples the baseband processing units (BBUs) from the remote radio heads (RRHs), allowing BBUs to be centralized and virtualized in data centers. This architecture improves network efficiency, reduces operational costs, and enables more flexible and scalable network management..

**Compressed Sensing** A signal processing technique for efficiently acquiring and reconstructing a signal by finding solutions to underdetermined linear systems. It is used in imaging and data compression..

**Compressive Sampling Matching Pursuit** An iterative algorithm for signal reconstruction that combines elements of compressed sensing and matching pursuit. It is used to efficiently recover sparse signals..

**Constant Coding and Modulation** A technique where coding and modulation parameters remain fixed, offering simplicity and predictability in performance, commonly used in stable communication channels..

**Convolutional Neural Network** A class of deep neural networks, most commonly applied to analyzing visual imagery. CNNs are used in image and video recognition, recommender systems, and natural language processing..

**Convolutional Neural Network** Another term for Convolutional Neural Network, a deep learning algorithm that can take in an input image, assign importance to various aspects and objects in the image, and differentiate one from the other..

**COordinate Rotation DIgital Computer** An algorithm used for efficiently performing trigonometric, hyperbolic, exponential, and other mathematical functions using only addition, subtraction, bitshift operations, and table lookups. CORDIC is particularly useful in hardware implementations such as FPGA and ASIC designs due to its simplicity and efficiency, making it ideal for applications in signal processing, computer graphics, and embedded systems..

**Cumulative Distribution Function** A function that represents the probability that a random variable takes a value less than or equal to a given value..

**Decibel** A logarithmic unit used to express the ratio of two values, commonly used to measure sound intensity, signal power, and other quantities in science and engineering. One decibel is one-tenth of a bel, a seldom-used unit. Decibels provide a convenient way to express large ratios, such as gain and loss in signal strength, on a manageable scale..

**Decibels relative to Full Scale** A unit of measurement that expresses the amplitude of a signal relative to the maximum level achievable (full scale) in a system, often used in digital audio and signal processing..

**DeConvolutional Neural Network** A network that performs the inverse operation of convolution, used to visualize learned features and in applications like image segmentation..

**Demodulation** The process of extracting the original data signal from a modulated carrier wave. Demodulation is the reverse operation of modulation, and it is necessary for interpreting the transmitted information at the receiver end. Techniques for demodulation vary depending on the modulation method used and include methods such as envelope detection for AM signals, frequency discrimination for FM signals, and coherent detection for PSK signals..

**Dictionary Learning** A machine learning method for building dictionaries upon which sparse representations of data can be constructed. It is used in image processing and signal processing..

**Digital Signal Processing** The use of digital processing, such as by computers or more specialized digital signal processors, to perform a wide variety of signal processing operations. It is used in telecommunications, audio signal processing, and other fields..

**Digital Video Broadcasting** A suite of internationally accepted open standards for digital television. DVB standards are used for satellite, cable, and terrestrial television broadcasts..

**Digital Video Broadcasting - Satellite** A satellite communication standard. It is widely used for broadcasting television and data services via satellite..

**Digital Video Broadcasting - Satellite - Second Generation** An enhanced version of the DVB-S standard for satellite broadcasting, offering improved performance and efficiency. It is widely used for broadcasting television and data services via satellite..

**Digital Video Broadcasting - Satellite - Second Generation - Extension** An extension of the DVB-S2 standard that provides additional features and improved performance, particularly for professional applications..

**Digital-to-Analog Converter** A device or circuit that converts digital data into an analog signal, often used to drive analog outputs such as speakers or actuators..

**Fast Fourier Transform** An algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse, efficiently. The FFT reduces the complexity of computing the DFT from $O(N^2)$ to $O(N \log N)$, where $N$ is the number of points in the sequence. This efficiency makes the FFT a fundamental tool in digital signal processing, used for spectral analysis, filtering, and fast convolution..

**Field-Programmable Gate Array** An integrated circuit designed to be configured by a customer or a designer after manufacturing—hence "field-programmable". FPGAs are used in a variety of applications including digital signal processing, telecommunications, and aerospace..

**Finite Impulse Response filter** A type of digital filter that responds to a finite number of input samples, producing an output that eventually settles to zero. FIR filters are inherently stable and have a linear phase response, making them suitable for applications in signal processing where phase linearity is important. They are widely used in audio processing, communications, and control systems..

**Forward Error Correction** A method of obtaining error control in data transmission by having the sender add redundant data and the receiver use this data to detect and correct errors without needing a retransmission..

**Fully Connected** A type of neural network layer where each neuron is connected to every neuron in the previous layer. It is often used in the final layers of neural networks..

**Gated Recurrent Unit** A type of recurrent neural network (RNN) architecture that is designed to handle sequence data and address the vanishing gradient problem. GRUs use gating units to modulate the flow of information, making them simpler and computationally more efficient than LSTMs while still retaining similar performance. They are used in various applications, including natural language processing, time-series forecasting, and speech recognition..

**Gaussian Minimum Shift Keying** A type of continuous-phase frequency shift keying modulation that applies a Gaussian filter to the input data to smooth the phase transitions. GMSK is known for its spectral efficiency and resistance to interference, making it suitable for mobile communication systems like GSM (Global System for Mobile Communications)..

**Generalized Orthogonal Matching Pursuit** An extension of the OMP algorithm used in sparse signal recovery, where multiple atoms are selected at each iteration, enhancing recovery in noisy environments..

**Generative Adversarial Network** A class of machine learning frameworks designed by Ian Goodfellow and his colleagues. GANs consist of two neural networks, the generator and the discriminator, that contest with each other in a zero-sum game framework..

**Global Navigation Satellite System** A satellite-based system that provides positioning, navigation, and timing (PNT) services worldwide, encompassing systems such as GPS, GLONASS, Galileo, and BeiDou..

**Graphics Processing Unit** A specialized processing unit designed to accelerate rendering of images and parallel processing tasks, widely used in graphics applications and increasingly in machine learning and scientific computations..

**Ground Station as a Service** A cloud-based service model that provides satellite operators with access to ground station networks on a pay-as-you-go basis. GSaaS allows operators to communicate with their satellites without the need to invest in and maintain their own ground station infrastructure. This service model offers flexibility, scalability, and cost-efficiency, making it easier to manage satellite operations and data transmission..

**Infinite Impulse Response filter** A type of digital filter that responds to an infinite number of input samples, due to feedback elements in its design. IIR filters are efficient in terms of computational resources because they can achieve a desired filter response with fewer coefficients than FIR filters. However, they can be unstable and have a non-linear phase response. They are commonly used in applications where computational efficiency is critical, such as in real-time signal processing..

**Institute of Electrical and Electronics Engineers** A professional association dedicated to advancing innovation and technological excellence for the benefit of humanity. IEEE is the world's largest technical professional organization with over 400,000 members worldwide, and it is known for its extensive range of standards, publications, conferences, and professional development programs in various fields of electrical, electronics, and computing engineering..

**Inter-Range Instrumentation Group** A set of time code standards, including IRIG-106, used for precise time and frequency synchronization in applications such as aerospace and defense..

**Inter-Symbol Interference** A phenomenon in digital communication where one symbol interferes with subsequent symbols, often caused by multipath propagation or bandwidth limitations, which can degrade the quality of the received signal..

**Interleaver** A device or algorithm used in digital communication systems and data storage to perform interleaving. The interleaver rearranges data in a non-contiguous way to protect against burst errors and improve error correction performance. Interleavers are commonly used in conjunction with error correction codes such as Turbo codes and LDPC codes to enhance their effectiveness..

**Interleaving** A technique used in digital communications and data storage to arrange data in a non-contiguous way. This method helps to protect against burst errors by distributing errors over a wider range of data, thereby improving error correction and overall system performance. Interleaving is commonly used in error correction schemes such as Turbo codes and LDPC codes, as well as in data storage formats like CDs and DVDs..

**Internet Protocol** A set of rules governing the format of data sent over the Internet or other networks. IP is responsible for addressing and routing packets of data so that they can travel across networks and arrive at the correct destination. It is the primary protocol in the Internet protocol suite, forming the basis of modern internet communication..

**Iterative Hard Thresholding** An optimization algorithm used in compressed sensing for recovering sparse signals by iteratively applying thresholding to update the estimated signal..

**Iterative Shrinkage-Thresholding Algorithm** An optimization algorithm used in sparse signal recovery and compressed sensing. ISTA iteratively applies thresholding to update the solution, balancing signal sparsity and data fidelity..

**K-Singular Value Decomposition** An algorithm for dictionary learning that iteratively optimizes both the dictionary and the sparse representations of the data. It is used in signal processing and image compression..

**Learning Rate Scheduler** In machine learning, a method to adjust the learning rate during training, helping to improve convergence and prevent overshooting in optimization..

**Least Absolute Shrinkage and Selection Operator** A regression analysis method that performs variable selection and regularization to enhance prediction accuracy and interpretability of the statistical model..

**Linear Prediction Coding** A tool used in signal processing and speech synthesis that represents the spectral envelope of a digital signal of speech in compressed form, using the information of a linear predictive model..

**Log-Likelihood Ratio** A measure in signal processing and statistical analysis that represents the ratio of the probabilities of two events, often used in decoding algorithms for error-correcting codes..

**Long Short-Term Memory** A type of RNN architecture that is capable of learning long-term dependencies. LSTM networks are well-suited to classifying, processing, and predicting time series given time lags of unknown duration..

**Low Earth Orbit** An orbit around Earth with an altitude between 160 km and 2,000 km. Satellites in LEO are used for various applications including communications, earth observation, and scientific missions..

**Low-Density Parity-Check codes** A class of error-correcting codes used in data transmission that allows for reliable communication over noisy channels by using sparse parity-check matrices..

**Mean Absolute Error** A measure used to evaluate the accuracy of a model by calculating the average of the absolute differences between predicted and actual values. MAE provides a straightforward interpretation of prediction errors, making it useful for regression analysis and model validation..

**Mean Squared Error** A measure of the average of the squares of the errors, that is, the average squared difference between estimated values and actual values. It is used in regression analysis to measure the quality of an estimator..

**Minimum Description Length** A principle in information theory and statistics that selects the model that minimizes the total description length of the data and the model itself. It balances data fitting and complexity to achieve an optimal representation..

**modcod** In DVB-S2 (Digital Video Broadcasting - Satellite, Second Generation), ModCod refers to the combination of modulation and coding schemes designed to optimize bandwidth efficiency and reliability in satellite communication systems. ModCod parameters vary to adapt to channel conditions..

**Modulation** The process of varying one or more properties of a carrier signal, such as its amplitude, frequency, or phase, in accordance with a data signal. Modulation is used in communication systems to transmit information over long distances by adapting the data signal to the characteristics of the transmission medium. Common modulation techniques include AM (Amplitude Modulation), FM (Frequency Modulation), and PSK (Phase Shift Keying)..

**Moving-Average model** A time series model that expresses the output variable as a linear function of past error terms. It is used for smoothing and forecasting time series data..

**Multiple Input Multiple Output** A technology used in wireless communication systems that employs multiple antennas at both the transmitter and receiver to improve communication performance. MIMO leverages spatial diversity to increase data throughput, enhance signal quality, and reduce the impact of interference and fading. It is widely used in modern wireless standards such as Wi-Fi, LTE, and 5G for its ability to significantly boost network capacity and reliability..

**Neural Network** A series of algorithms that attempt to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates..

**Non return to zero** A binary code in which 1s are represented by one significant condition (usually a positive voltage) and 0s by another significant condition (usually zero voltage), with no return to a neutral (zero) condition between bits..

**Non-deterministic Polynomial-time hard** A classification in computational complexity theory for problems that are at least as hard as the hardest problems in NP (Non-deterministic Polynomial time), often requiring non-polynomial time to solve..

**Non-linear Autoregressive model** A type of autoregressive model where the relationship between the variables is non-linear. It is used for modeling and forecasting non-linear time series data..

**Numba** An open-source JIT (Just-In-Time) compiler that translates a subset of Python and NumPy code into fast machine code, leveraging LLVM to optimize code execution. Commonly used to speed up numerical and scientific computations..

**Offset Quadrature Phase-Shift Keying** A digital modulation scheme that minimizes phase discontinuities, reducing amplitude fluctuations and improving signal robustness..

**Online Dictionary Learning** A method in machine learning and signal processing for constructing a dictionary of basis functions that can represent data sparsely, learned incrementally to handle large datasets..

**Open Neural Network Exchange** An open-source format for machine learning models that allows interoperability across different deep learning frameworks, enabling easy model sharing and deployment..

**Orthogonal Matching Pursuit** An iterative algorithm for finding the best matching projections of multidimensional data onto an over-complete dictionary. It is used in compressed sensing and sparse approximation..

**Phase Error** A type of error in communication systems where the phase of the received signal deviates from the phase of the transmitted signal. Phase errors can be caused by various factors, including noise, multipath propagation, and oscillator instability. These errors can significantly affect the performance of modulation schemes that rely on precise phase information, such as PSK (Phase Shift Keying) and QAM (Quadrature Amplitude Modulation). Mitigating phase error often involves using phase-locked loops (PLLs) and other synchronization techniques..

**Phase Shift Keying** A digital modulation technique that conveys data by changing, or modulating, the phase of a reference signal (the carrier wave). PSK is widely used in wireless communication systems, including Wi-Fi, satellite communication, and RFID. It offers robustness against noise and is efficient in terms of bandwidth usage..

**Phase-Locked Loop** An electronic circuit that synchronizes an output oscillator signal with a reference signal in phase and frequency. It is widely used in telecommunications, computers, and other electronic applications..

**Prediction by Partial Matching** A data compression technique based on adaptive statistical modeling and partial context matching. PPM predicts the next symbol in a sequence based on previous symbols, commonly used in text compression and machine learning..

**Probability Density Function** In probability and statistics, a function that describes the relative likelihood for a random variable to take on a given value..

**Quadrature Amplitude Modulation** A modulation technique that conveys data by changing the amplitude of two carrier waves, using amplitude-shift keying (ASK) on both carriers. These two waves, usually sinusoids, are out of phase by 90 degrees (hence the name quadrature). QAM is used in many digital telecommunication systems, including cable television and internet, wireless communication, and digital television broadcasting, due to its high spectral efficiency..

**Radio Frequency** The range of electromagnetic frequencies used in wireless communication systems, typically from 3 kHz to 300 GHz. RF technology is the backbone of modern wireless communication,

enabling the transmission of data over the airwaves. It is used in various applications such as broadcasting, cellular networks, satellite communication, and radar..

**Receiver** A device or component that receives incoming signals or data transmitted over electromagnetic waves, such as radio waves, and converts them back into a usable form. Receivers are essential in communication systems for decoding and interpreting the transmitted information. "RX" is also commonly used to refer to the reception channel in communication systems..

**Rectified Linear Unit** An activation function used in neural networks that allows a model to account for non-linearities. It is defined as the positive part of its argument..

**Recurrent Neural Network** A class of neural networks where connections between nodes can create cycles, allowing the network to maintain a memory of previous inputs. They are used for tasks involving sequential data, such as language modeling and time-series prediction..

**Register Transfer Level** An abstraction in digital design that describes the flow of signals and data between registers and the logical operations performed on those signals, often used in hardware design and verification..

**Restricted Isometry Property** A property of certain matrices that are used in compressed sensing and sparse signal recovery. The RIP ensures that the geometry of the sparse signal is preserved during the measurement process, allowing for accurate reconstruction of the signal from a limited number of measurements. Matrices satisfying the RIP condition are crucial for the effectiveness of compressed sensing algorithms..

**Return to zero** A line code in which the signal drops to zero between each pulse. This coding scheme can be used to ensure synchronization between the sender and the receiver..

**Shaped Offset Quadrature Phase-Shift Keying** A variant of OQPSK that uses continuous phase modulation to achieve spectral efficiency and minimize out-of-band emissions..

**Signal-to-Noise Ratio** A measure used in science and engineering to quantify the level of a desired signal to the level of background noise. It is defined as the ratio of signal power to the noise power, often expressed in decibels (dB). Higher SNR values indicate a cleaner (less noisy) signal, which is crucial for accurate data transmission and reception in communication systems..

**Simultaneous Orthogonal Matching Pursuit** An algorithm for jointly sparse signal recovery across multiple measurement vectors, commonly used in applications such as array signal processing and multi-channel compressed sensing..

**Single Carrier with Cyclic Coding** A modulation and coding scheme used in telecommunications to improve the robustness and efficiency of data transmission over single-carrier systems..

**Singular Value Decomposition** A factorization of a matrix into three matrices, used in many applications in signal processing and statistics for dimensionality reduction and data compression..

**Smoothed L0** An algorithm in compressed sensing for sparse signal recovery, which approximates the L0 norm with a smooth function to make the optimization problem more tractable..

**Software Defined Radio** A radio communication system where components that have traditionally been implemented in hardware are instead implemented by means of software on a personal computer or embedded system. SDR allows for flexible and adaptable radio systems that can support multiple communication standards and can be updated with new functionalities through software updates. It is widely used in research, prototyping, and commercial applications such as telecommunications, broadcasting, and military communications..

**Stochastic Gradient Descent** An iterative method for optimizing an objective function with suitable smoothness properties. It is commonly used to train machine learning models..

**Stochastic Weight Averaging** An optimization technique in deep learning that averages weights from different training epochs to improve generalization and robustness..

**Transmitter** A device or component that sends out signals or data in the form of electromagnetic waves, such as radio waves, to be received by a receiver. Transmitters are used in a wide variety of

communication systems, including radio, television, and cellular networks. "TX" is also commonly used to refer to the transmission channel in communication systems..

**Variable Coding and Modulation** A technique allowing the adaptation of coding and modulation parameters based on channel conditions, optimizing spectral efficiency and robustness..

**Variational AutoEncoder** A type of autoencoder that learns a probabilistic mapping from the input data to a latent space, allowing for generation of new data samples..

**Vector Quantization** A quantization technique used in signal processing and data compression where vectors of data are mapped to a finite set of vectors in a codebook. VQ reduces the dimensionality and redundancy of data, making it useful for applications like image and speech compression, pattern recognition, and machine learning. The process involves dividing the data into blocks and then replacing each block with the closest codebook vector..

**VMEbus International Trade Association** An organization that develops and promotes open technology standards for the embedded computing industry. One of its well-known standards is VITA 49, which defines a transport layer protocol for digitized radio communication signals, facilitating interoperability between different systems and components in software-defined radio (SDR) and other applications..

**waveform** In telecommunications, a waveform represents the shape and form of a signal that carries data over a medium. It is characterized by parameters such as amplitude, frequency, and phase..

# Acronyms

**ACM** Adaptive Coding and Modulation. 149, *Glossary:* Adaptive Coding and Modulation

**ADC** Analog-to-Digital Converter. 31, 65, 84, 98, 109, *Glossary:* Analog-to-Digital Converter

**AGC** Automatic gain control. 13, 65, 66, 68, 109, *Glossary:* Automatic gain control

**AI** Artificial Intelligence. 153, *Glossary:* Artificial Intelligence

**AIC** Akaike Information Criterion. 119, *Glossary:* Akaike Information Criterion

**AK-SVD** Accelerated K-Singular Value Decomposition. 102, *Glossary:* Accelerated K-Singular Value Decomposition

**APSK** Amplitude Phase Shift Keying. 14, 32, 67, 68, 72, 74, 78, 83, 92, 94–97, 142–144, *Glossary:* Amplitude Phase Shift Keying

**AR** Autoregressive model. 120, 121, 127, 133, 185, *Glossary:* Autoregressive model

**ARMA** Autoregressive Moving-Average model. 120, 128, 185, *Glossary:* Autoregressive Moving-Average model

**ASIC** Application-Specific Integrated Circuit. 65, 150, 161, *Glossary:* Application-Specific Integrated Circuit

**AWGN** Additive White Gaussian Noise. 38, 40, 89, 160, *Glossary:* Additive White Gaussian Noise

**BER** Bit-error rate. 36, 38, 76, 82, 85, 104, 108–110, 114, 158, 159, *Glossary:* Bit-error rate

**BIC** Bayesian Information Criterion. 119, *Glossary:* Bayesian Information Criterion

**BP** Belief Propagation. 67, *Glossary:* Belief Propagation

**BP** Basis Pursuit. 58, *Glossary:* Basis Pursuit

**C-RAN** Centralized Radio Access Network. 16, 30–32, 118, 195, 196, *Glossary:* Centralized Radio Access Network

**CCM** Constant Coding and Modulation. 149, *Glossary:* Constant Coding and Modulation

**CDF** Cumulative Distribution Function. 70, 71, 186, *Glossary:* Cumulative Distribution Function

**CNN** Convolutional Neural Network. *Glossary:* Convolutional Neural Network

**ConvNet** Convolutional Neural Network. *Glossary:* Convolutional Neural Network

**CORDIC** COordinate Rotation DIgital Computer. 115, *Glossary:* COordinate Rotation DIgital Computer

**CoSaMP** Compressive Sampling Matching Pursuit. 100, 114, *Glossary:* Compressive Sampling Matching Pursuit

**CPU** Central Processing Unit. 15, 65, 76, 102, 136, 138, 139, 150, 160, 197, *Glossary:* Central Processing Unit

**CS** Compressed Sensing. 59, 60, 84, *Glossary:* Compressed Sensing

**DAC** Digital-to-Analog Converter. 65, *Glossary:* Digital-to-Analog Converter

**dB** Decibel. 38, *Glossary:* Decibel

**dBFs** Decibels relative to Full Scale. 66, 68, 109, *Glossary:* Decibels relative to Full Scale

**DeconvNet** DeConvolutional Neural Network. *Glossary:* DeConvolutional Neural Network

**DL** Dictionary Learning. 100, *Glossary:* Dictionary Learning

**DSP** Digital Signal Processing. *Glossary:* Digital Signal Processing

**DVB** Digital Video Broadcasting. *Glossary:* Digital Video Broadcasting

**DVB-S** Digital Video Broadcasting - Satellite. *Glossary:* Digital Video Broadcasting - Satellite

**DVB-S2** Digital Video Broadcasting - Satellite - Second Generation. 13–15, 37, 66–68, 82, 83, 109, 113, 142–144, 149, 159, *Glossary:* Digital Video Broadcasting - Satellite - Second Generation

**DVB-S2X** Digital Video Broadcasting - Satellite - Second Generation - Extension. 142, 144, *Glossary:* Digital Video Broadcasting - Satellite - Second Generation - Extension

**FC** Fully Connected. 136, *Glossary:* Fully Connected

**FEC** Forward Error Correction. 14, 15, 32, 39, 67, 80, 82, 108–110, 142, 158, 160, *Glossary:* Forward Error Correction

**FFT** Fast Fourier Transform. 14, 77–80, 138, *Glossary:* Fast Fourier Transform

**FIR** Finite Impulse Response filter. 14, 15, 79, 123, 127, *Glossary:* Finite Impulse Response filter

**FPGA** Field-Programmable Gate Array. 16, 45, 65, 76, 90, 115, 116, 129, 136, 139, 150–155, 160, 161, 197, *Glossary:* Field-Programmable Gate Array

**GAN** Generative Adversarial Network. 61, *Glossary:* Generative Adversarial Network

**GMSK** Gaussian Minimum Shift Keying. 41, 72, 74, 89, 92, 94–97, 121, 131, 148, *Glossary:* Gaussian Minimum Shift Keying

**GNSS** Global Navigation Satellite System. 59, *Glossary:* Global Navigation Satellite System

**gOMP** Generalized Orthogonal Matching Pursuit. 37, 100, 102, *Glossary:* Generalized Orthogonal Matching Pursuit

**GPU** Graphics Processing Unit. 136, 139, 150, 160, *Glossary:* Graphics Processing Unit

**GRU** Gated Recurrent Unit. 136, 137, 139, 140, 143, *Glossary:* Gated Recurrent Unit

**GSaaS** Ground Station as a Service. 29, 30, 195, *Glossary:* Ground Station as a Service

**IEEE** Institute of Electrical and Electronics Engineers. 31, *Glossary:* Institute of Electrical and Electronics Engineers

**IHT** Iterative Hard Thresholding. 37, 58, 100, *Glossary:* Iterative Hard Thresholding

**IIR** Infinite Impulse Response filter. *Glossary:* Infinite Impulse Response filter

**IP** Internet Protocol. 31, 32, 45, 50, 131, 169, *Glossary:* Internet Protocol

**IRIG** Inter-Range Instrumentation Group. 80, *Glossary:* Inter-Range Instrumentation Group

**ISI** Inter-Symbol Interference. 41, 125, *Glossary:* Inter-Symbol Interference

**ISTA** Iterative Shrinkage-Thresholding Algorithm. 60, *Glossary:* Iterative Shrinkage-Thresholding Algorithm

**K-SVD** K-Singular Value Decomposition. 100, 102, 115, *Glossary:* K-Singular Value Decomposition

**LASSO** Least Absolute Shrinkage and Selection Operator. 58, 100, 102, 103, *Glossary:* Least Absolute Shrinkage and Selection Operator

**LDPC** Low-Density Parity-Check codes. 67, *Glossary:* Low-Density Parity-Check codes

**LEO** Low Earth Orbit. *Glossary:* Low Earth Orbit

**LLR** Log-Likelihood Ratio. 67, *Glossary:* Log-Likelihood Ratio

**LPC** Linear Prediction Coding. 13, 15, 37, 53, 54, 57, 120–129, 131, 133, 134, 136, 137, 139, 140, 142–144, 146, 147, 156–158, 185, *Glossary:* Linear Prediction Coding

**LRS** Learning Rate Scheduler. 138, *Glossary:* Learning Rate Scheduler

**LSTM** Long Short-Term Memory. 135, 140, *Glossary:* Long Short-Term Memory

**MA** Moving-Average model. 120, 185, *Glossary:* Moving-Average model
**MAE** Mean Absolute Error. 104, 146, *Glossary:* Mean Absolute Error
**MDL** Minimum Description Length. 119, *Glossary:* Minimum Description Length
**MIMO** Multiple Input Multiple Output. 31, *Glossary:* Multiple Input Multiple Output
**MSE** Mean Squared Error. 36, 72, 75, 76, 104, 164, *Glossary:* Mean Squared Error

**NAR** Non-linear Autoregressive model. 15, 135, 137, *Glossary:* Non-linear Autoregressive model
**NN** Neural Network. 61, 147, 148, 163, *Glossary:* Neural Network
**NP-hard** Non-deterministic Polynomial-time hard. 58, 100, 116, *Glossary:* Non-deterministic Polynomial-time hard
**NRZ** Non return to zero. 41, 42, 66, 78, 91, 92, 94–98, 124, 125, *Glossary:* Non return to zero

**ODL** Online Dictionary Learning. 14, 100–102, 109, *Glossary:* Online Dictionary Learning
**OMP** Orthogonal Matching Pursuit. 14, 37, 58, 60, 100–102, 109, 110, 114–116, *Glossary:* Orthogonal Matching Pursuit
**ONNX** Open Neural Network Exchange. 138, *Glossary:* Open Neural Network Exchange
**OQPSK** Offset Quadrature Phase-Shift Keying. 35, 67, *Glossary:* Offset Quadrature Phase-Shift Keying

**PDF** Probability Density Function. 70, 71, *Glossary:* Probability Density Function
**PLL** Phase-Locked Loop. *Glossary:* Phase-Locked Loop
**PPM** Prediction by Partial Matching. 56, 118, *Glossary:* Prediction by Partial Matching
**PSK** Phase Shift Keying. 13–15, 35, 66–68, 72, 74, 81, 83, 91, 92, 94–96, 101–103, 109–111, 113, 124–128, 140, 141, 143, 145, 148, *Glossary:* Phase Shift Keying

**QAM** Quadrature Amplitude Modulation. 74, 92, 94–97, 148, *Glossary:* Quadrature Amplitude Modulation

**ReLU** Rectified Linear Unit. 136, 151, *Glossary:* Rectified Linear Unit
**RF** Radio Frequency. 31, 32, 45, 50, 89, 131, 169, 195, 196, *Glossary:* Radio Frequency
**RIP** Restricted Isometry Property. 59, 60, *Glossary:* Restricted Isometry Property
**RNN** Recurrent Neural Network. 61, 143, *Glossary:* Recurrent Neural Network
**RRC** (Square)-Root-raised-cosine filter. 13–15, 42, 66, 67, 72, 74, 75, 78, 81, 83, 85, 91, 92, 94–98, 102, 103, 113, 124–126, 140, 148, 153, 154, *Glossary:* (Square)-Root-raised-cosine filter
**RTL** Register Transfer Level. 153, *Glossary:* Register Transfer Level
**RX** Receiver. 30, 36, 98, 142, 156, 197, *Glossary:* Receiver
**RZ** Return to zero. 41, 42, 78, 97, 98, *Glossary:* Return to zero

**SCCC** Single Carrier with Cyclic Coding. 66, *Glossary:* Single Carrier with Cyclic Coding
**SDR** Software Defined Radio. 30, *Glossary:* Software Defined Radio
**SGD** Stochastic Gradient Descent. 138, *Glossary:* Stochastic Gradient Descent
**SL0** Smoothed L0. 100, 114, *Glossary:* Smoothed L0
**SNR** Signal-to-Noise Ratio. 15, 32, 38, 50, 52, 65, 66, 97, 98, 103, 109, 125, 126, 147–149, 156, *Glossary:* Signal-to-Noise Ratio
**SOMP** Simultaneous Orthogonal Matching Pursuit. 37, 102, *Glossary:* Simultaneous Orthogonal Matching Pursuit
**SOQPSK** Shaped Offset Quadrature Phase-Shift Keying. 131, *Glossary:* Shaped Offset Quadrature Phase-Shift Keying
**SVD** Singular Value Decomposition. 115, *Glossary:* Singular Value Decomposition

**SWA** Stochastic Weight Averaging. 138, *Glossary:* Stochastic Weight Averaging

**TX** Transmitter. 30, 36, 97, 98, 131, 138, 142, 196, *Glossary:* Transmitter

**VAE** Variational AutoEncoder. 164, *Glossary:* Variational AutoEncoder

**VCM** Variable Coding and Modulation. 149, *Glossary:* Variable Coding and Modulation

**VITA** VMEbus International Trade Association. 31, 32, 195, 196, *Glossary:* VMEbus International Trade Association

**VQ** Vector Quantization. 51, 164, 165, *Glossary:* Vector Quantization

Chapter 1

# Introduction

*At the Journey's Start*


1.1 Context and motivation . . . . . **29**
1.1.1 Ground Station as a Service . . . . . 29
1.1.2 Emergence of Software-Defined Radio (SDR) . . . . . 30
1.1.3 RF over IP . . . . . 31
1.2 Problem statement . . . . . **33**
1.3 Hypotheses & challenges . . . . . **35**
1.3.1 Origins . . . . . 35
1.3.2 Challenges of lossless compression . . . . . 36
1.3.3 Challenges of lossy compression . . . . . 36
1.4 Achievements & innovations . . . . . **37**
1.5 Methodology . . . . . **38**
1.5.1 Notations and definitions . . . . . 38
1.5.2 Data sources . . . . . 40
1.5.3 The importance of shaping when compressing signals . . . . . 40
1.5.4 Techniques and tools . . . . . 42
1.6 Limitations and scope . . . . . **43**
1.7 Structure of the document . . . . . **44**


## 1.1 Context and motivation

### 1.1.1 Ground Station as a Service

Digital communications are, in essence, the process of transmitting information through digital signals in noisy environments with the least possible errors [1]. This field encompasses a wide range of technologies and techniques, each designed to facilitate the efficient and reliable exchange of data across various communication channels [2]–[4]. Among the modern applications of digital communications, space communications stand out due to their complexity and evolving requirements.



**Figure 1.1:** Space communication systems.
**Source:** *Generated by DALL-E*

The arena of space communications has witnessed significant transformation in recent decades. As we move further into the New Space Age of satellite constellations and interplanetary missions, the complexities and demands of space communications continue to escalate. Innovations in this field are numerous, driven by both technological advances and burgeoning use cases. In this context, some companies own a distributed and flexible ground station infrastructure whose access is provided as a service to any customer company to exploit and communicate with its satellites. This emerging model is known as Ground Station as a Service (GSaaS).

Led by companies from the Cloud ecosystem, like *Amazon AWS* or *Microsoft Azure*, Ground Station as a Service has revolutionized the provision of services, particularly in the realm of Earth

observation. This innovation has democratized access to **what was once prohibitively expensive infrastructure for many organizations**, allowing them to leverage ground stations and extensive terrestrial network systems without huge investments: these companies can focus on their core business around satellite design or data processing, for instance. Drawing inspiration from the cost effectiveness and operational agility seen in mobile networks, these services have adopted the principles of **Centralized Radio Access Network (C-RAN)**. Centralization, a hallmark of C-RAN, offers notable advantages that include economical maintenance, optimized resource distribution, and improved efficiency in a widely deployed network. **This method requires that most of the digital processing is done within data centers, reducing the function of base or ground stations to simply transmitting and receiving raw digitized signal samples to and from their analog front ends using high-speed analog-to-digital and digital-to-analog converters.** In such an architecture, the **raw digitized samples are transferred as packets over a network**, which requires substantial bandwidth to connect ground stations with data centers to handle both the transmission (TX) and reception (RX) paths. **The *TX path* is characterized by its noiseless transmission of signals to the antenna, whereas the *RX path* involves the reception of noisy signals from the antenna.**

> To support the efficient and reliable transport of these raw signal samples over complex terrestrial networks, new technological and infrastructural solutions are required, capable of supporting such extensive data transfer requirements, bridging the gap between traditional ground station infrastructure and advanced networked environments.

### 1.1.2 Emergence of Software-Defined Radio (SDR)

Software Defined Radio (SDR) technology represents a paradigm shift from traditional hardware-centric radio systems to more versatile software-driven solutions. Initially pioneered by the United States military, SDR was developed to enhance local interconnectivity through the use of multiple, interchangeable components. **The fundamental idea was to create a radio system that could be easily reconfigured and adapted to various communication standards and protocols through software adjustments**, thus eliminating the need for extensive hardware changes. In this way, SDRs provides unmatched flexibility in handling different modulation schemes, frequency bands, and encoding techniques. This is particularly crucial in dynamic environments such as space communications, where conditions can be highly variable and unpredictable. Unlike traditional radios, which are limited by their fixed hardware configurations, SDRs can be updated and reprogrammed remotely, making them ideal for long-term missions where physical access to the hardware is limited or impossible.

The integration of SDR technology into space missions has facilitated significant advances in communication capabilities. For instance, NASA's Lunar Reconnaissance Orbiter (LRO) has benefited from SDR by enabling high-data-rate communication links that are essential for transmitting vast amounts of scientific data back to Earth. The flexibility of SDR allows for adjustments to be made in real time to accommodate changes in signal quality and environmental conditions, ensuring reliable and efficient data transmission.

Considering these characteristics, SDR has become a cornerstone technology within Ground Station as a Service (GSaaS) infrastructures, primarily due to its unparalleled flexibility in large-scale and highly parallelized signal processing tasks. SDR allows for an adaptable and efficient handling of sampled signals through software-defined components. This approach aligns perfectly with the GSaaS requirements for scalable and cost-effective signal processing capabilities.

In addition, **integration of SDR on a large scale requires meticulous control over the transmission and reception of signal samples to ensure the integrity of the communication**

**chain**. This is where the development of *RF over IP* becomes critical, as it facilitates the reliable transport of these digitized signals over IP networks.

### 1.1.3 RF over IP

As highlighted previously, digital communications are increasingly decoupled from **traditional hardware and geographical constraints**, moving towards more flexible, scalable network architectures. This shift is exemplified in the concept of "*RF over IP*" (Radio Frequency over Internet Protocol), which broadly encompasses the transport of digitized radio signals over terrestrial networks.

This evolution is driven by the need to connect remote antenna sites, which do not handle modulation / demodulation processing –and were historically located close to the receiving antenna–, to centralized data processing centers that manage these functions. Such connections require robust systems capable of converting radio signals to digital data and vice versa, **facilitating their exchange as packets or frames across network infrastructures using appropriate protocols**. In addition, a high-capacity network is required to handle such a volume of data, typically facilitated by large-bandwidth optical fibers.

Originally conceived for simpler applications, this technique has evolved into a complex framework involving various transmission forms such as "*IF over IP*" (Intermediate Frequency over Internet Protocol) and "*Radio over IP.*". However, it is important to recognize that these terms, while convenient, may obscure the diverse technological and architectural nuances involved in modern networked communication systems, particularly in mobile cellular and satellite systems where dedicated network layers and Ethernet-based solutions (e.g., IEEE 1914.3 Radio over Ethernet [5]) are increasingly prevalent.

The methodologies and protocols used for the transport of *RF over IP* are diverse, each responding to specific technical and operational needs. For example, the VITA 49 protocol standard [6] is crucial in defining how software-defined radios handle digitized –sampled– signals, facilitating a seamless interface between digital signal processing units and various transmission mediums. Alongside VITA 49, protocols such as the *Common Public Radio Interface* (CPRI) [7] and its successors such as eCPRI [8] underscore advances in reducing latency and improving data transfer rates between radio equipment and baseband units in cellular networks. These standards are increasingly complemented by Ethernet-based solutions, such as IEEE 1914.3 Radio over Ethernet [5], which addresses the scalability and flexibility needs of massive MIMO and advanced beamforming technologies by leveraging the widespread Ethernet infrastructure. Collectively, these techniques not only cater to the efficient transmission of vast volumes of data, but also ensure compatibility and interoperability across an evolving ecosystem of network architectures and devices, highlighting a fundamental shift towards more open, flexible, and scalable communication systems, as advocated by the OpenRAN initiative.

Different standards, recommendations and articles published by these organizations [9], [10] are part of the OpenRAN initiative and are instrumental in shaping the future of telecommunications. They provide a comprehensive framework for the development and deployment of open, interoperable, and cost-effective RAN solutions, fostering innovation and competition in the telecommunications industry. In this context, **the virtualization of satellite communications must align with these standards to ensure seamless integration and interoperability with terrestrial networks and also to benefit from the vast market of reduced-cost network components**.

Despite the appeal of the contemporary C-RAN model, it faces a significant challenge: it depends on the ability to transmit all these raw RF samples as packets across a terrestrial network. The Nyquist theorem mandates that to digitize an analog bandwidth, it must be sampled at a minimum rate of twice the original frequency. Using a 12-bit ADC at the Nyquist frequency for a 100MHz band results in a data generation rate of 2.4 Gbps, calculated as $12 \times 2 \times 100$ Mbps. Additional data over-

head comes from the transport protocol (for example, VITA-49.2), Ethernet transport protocols (such as RTP/UDP/IP/ethernet) and signaling overhead. Consequently, to transmit a 100 MHz band via IP over IP, the overall data throughput requirement typically reaches 3 Gbps. With 100 MHz bandwidth, with DVB-S2X standard and high Signal-to-Noise Ratio (SNR), one may modulate the signal with a 256-APSK with a Forward Error Correction (FEC) code rate equal to 3/4 and tight roll-off equal to 1.05. This would provide an information date rate equal to 571 Mbps. With a low SNR, one may also combine a QPSK with FEC rate equal to 1/4 and the same roll-off, achieving an information date rate equal to 48 Mbps. **Therefore, processing the digitized RF samples in a distant data center requires transferring between 5 and 62 times as much digital data over the terrestrial network as processing the samples close to the antenna and sending only digital information data!**

> Can such a cost be minimized to embrace the benefits of C-RAN infrastructures and interoperability with contemporary cellular networks?

## 1.2 Problem statement

With the ongoing advancement in digital communication systems, the demand for **effective data transmission over restricted bandwidth is becoming more and more essential**. This requirement leads to the discussion of **bandwidth minimization**, the techniques to achieve it, and its impact on the signal.
This is why the primary issue of this thesis can be summarized as follows:

**Research question**

*How can we minimize the bandwidth required to transmit a baseband signal while preserving a high level of data integrity?*

This issue primarily spans across two main aspects:

- **Network approach:** This facet explores improvements in transmission protocols to optimize bandwidth usage. It involves strategies to alleviate congestion in protocols such as TCP or UDP and improvements in the efficiency of the physical layer of communication systems.
- **Data-based Approach:** This aspect focuses on decreasing the effective size of the data to be transmitted. It involves the application of compression techniques to reduce the volume of the data, as illustrated in Figure 1.2

With regard to the network approach, our field is not the only one facing challenges in transmission efficiency. Various research groups have long attempted to refine existing protocols with the objective of limiting congestion, improving throughput and latency, and ensuring accurate packet reconstruction upon arrival [11], [12]. Furthermore, it is challenging to see how the unique aspects of satellite communication systems under our study could be utilized and leveraged within this research framework. Integration of satellite-specific characteristics into the development of improved transmission protocols presents an intricate problem that requires a nuanced understanding of both the network and data-based approaches to data transmission.

In this thesis, **the focus is primarily on the latter approach, specifically targeting the compression of baseband signals.** The compression of baseband signals presents a multifaceted challenge, necessitating a comprehensive consideration of the signal's physical attributes, inherent noise, the transmission protocol employed, and the characteristics of the hardware utilized for signal transmission.

A pivotal distinction in the realm of compression methods is between lossless and lossy compression. With lossless compression, even the most adapted algorithm hits a wall and cannot compress beyond a limit. Lossy compression allows to compress beyond this limit but distorts the reconstructed signal to a level whose acceptability depends on the considered application. **Consequently, a crucial task in this context is the extraction and analysis of relevant metrics**. These metrics play a crucial role in assessing the compression's quality and the extent of information loss caused by the compression technique. This evaluation is vital to determine the feasibility and effectiveness of the compression methodology in practical scenarios.

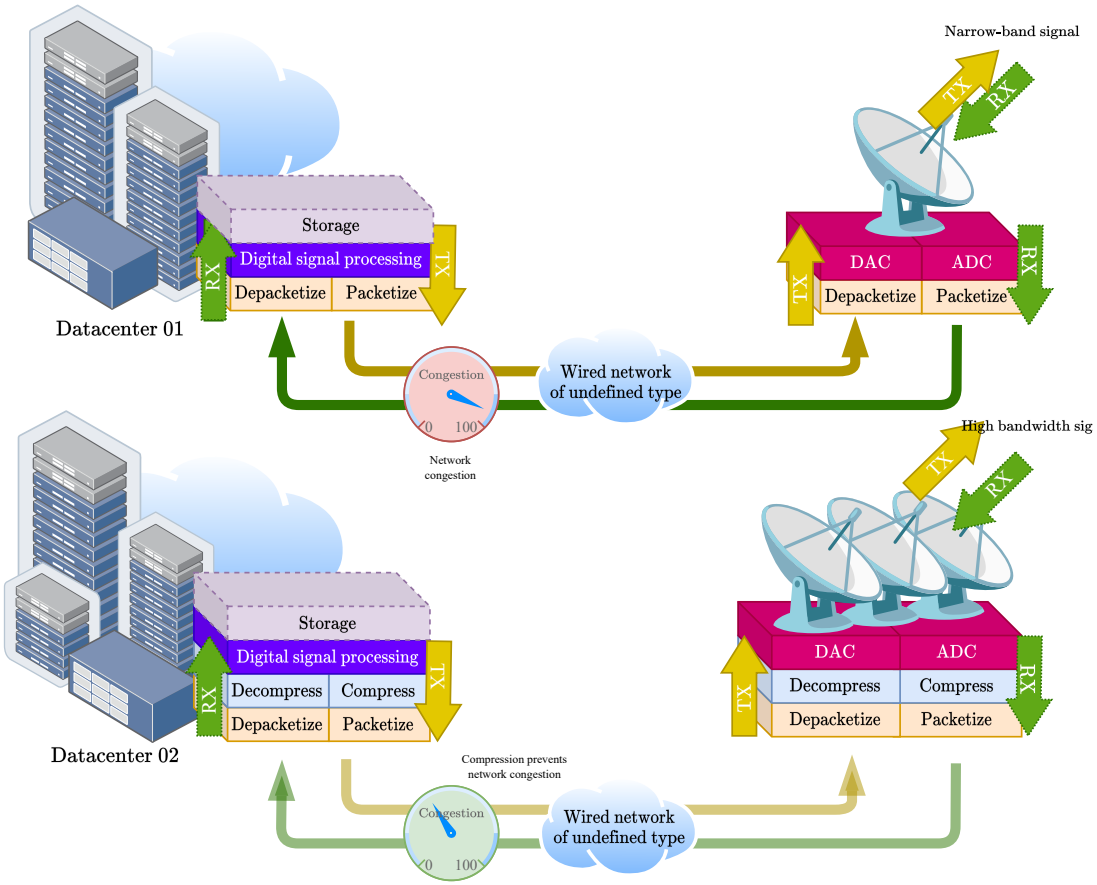



**Figure 1.2:** Overview of remote/centralized modem processing architectures used in satellite communications systems. Compressing the digitized raw signals is the key to limit the congestion with an affordable network infrastructure.

## 1.3 Hypotheses & challenges

### 1.3.1 Origins

As research began in 2020, the interest of SAFRAN in virtualization was evident, primarily through the NuRoN initiative [13]. At that juncture, the literature on satellite communications and compression within the framework of centralized networks was scant, with notable contributions including:

- The work by Alain Thomas on the quantization of M-PSK & OQPSK waveforms [14].
- An exploration of the feasibility of employing centralized networks for satellite communications [14].

However, the discourse on Centralized Networks had already permeated the realm of mobile networks. Furthermore, foundational discussions on the operational prerequisites of such architectures were conducted [15], along with investigations into compression methodologies, predominantly lossless [16]–[18].

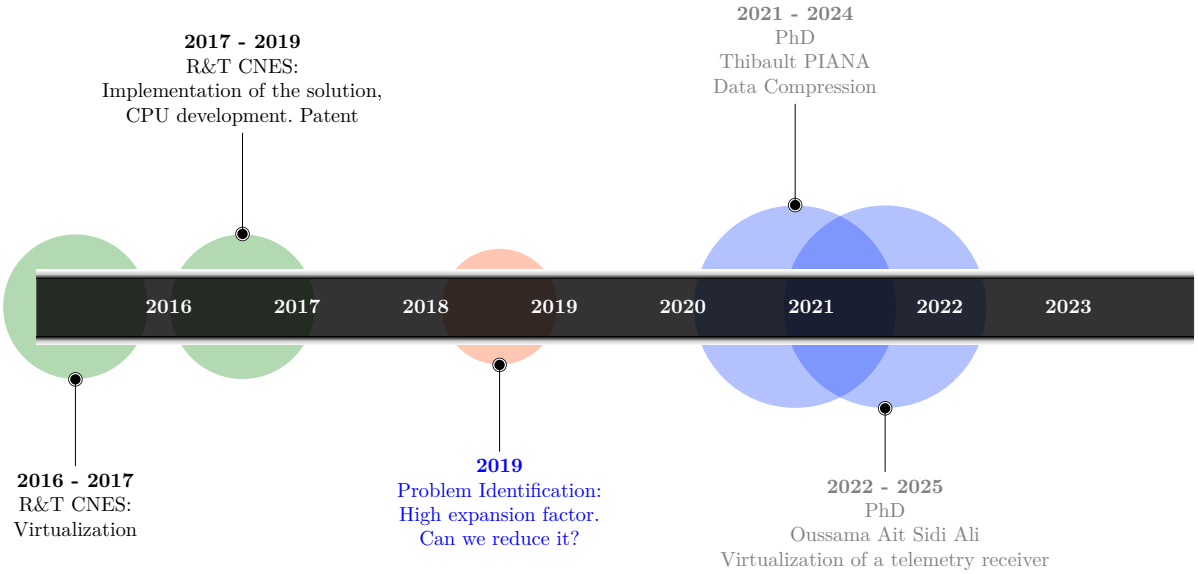



**Figure 1.3:** Evolution of related projects at Safran Data Systems

Given the increasing importance of this topic and the existing gaps in the state of the art, the recent literature attempts to advance compression techniques, both lossless [16], [18]–[22] and lossy [23], although within the context of mobile networks. This underlines a prominent research void in satellite communications, necessitating a comprehensive exploration to bridge this knowledge gap. Consequently, the main guidelines of this thesis are as follows.

1 · To rigorously examine the applicability and efficiency of centralized network architectures in satellite communications, drawing parallels and contrasts with their implementation in mobile networks.

2 · Innovating and improving data compression techniques, with a dual focus on lossless and lossy methods, tailored specifically for the satellite communications domain.

This project seeks to address current research deficiencies and notably improve the knowledge base in satellite communications, thereby contributing to the development of network architectures that are more robust, efficient, and scalable.

### 1.3.2 Challenges of lossless compression

The pursuit of effective lossless compression algorithms suitable for both transmit (TX) and receive (RX) environments presents a significant challenge in the field of data compression. The primary objective is to find an algorithm that can efficiently compress data without loss of information. Given the nature of lossless compression, the study and selection of an appropriate algorithm are somewhat simplified compared to lossy methods. The focus shifts towards identifying the most suitable algorithm for the given data set and implementing it reliably, without needing to assess the impact of compression on the data integrity, as the process guarantees no loss of information.

However, the principal challenge lies not just in algorithm selection, but in ensuring that the chosen method can be applied in real-time scenarios. This involves considerations of implementation feasibility, performance benchmarks, and operational efficiency. The algorithm must not only compress the data effectively but also meet the demands of real-time processing, which includes quick execution times and minimal resource consumption. This necessity for a balance between compression efficacy and operational practicality makes the search for an ideal lossless compression algorithm a complex endeavor.

### 1.3.3 Challenges of lossy compression

The complexity of lossy compression lies in identifying the optimal balance between the compression ratio and the resultant degradation of data. Various approaches prioritize different aspects of this balance. Some strategies aim to minimize the Mean Squared Error (MSE) through techniques such as nonuniform quantization [24], [25], while others focus on maximizing compression. However, the objective within the scope of this study is to achieve the highest possible compression ratio **without exceeding a predefined level of acceptable degradation**. This target introduces a significant challenge, as minimizing MSE does not invariably lead to minimizing actual Bit-error rate (BER) or maintaining the quality of compressed data according to objective standards.

Moreover, the lossy compression process must exhibit a high degree of stability. It is imperative that the method can be consistently **replicated and simulated**, delivering equivalent results in multiple iterations. This stability requirement ensures that the compression algorithm not only achieves the desired balance between compression ratio and data degradation, but also **remains reliable and predictable over time**. Achieving such stability, especially in the face of varying data sets and operational conditions, further complicates the development and implementation of effective lossy compression algorithms.

## 1.4 Achievements & innovations

This section outlines the anticipated contributions of this thesis, organized by thematic areas such as compression techniques, algorithmic development, and code implementation. Each contribution aims to advance the field of signal processing in telecommunications, specifically focusing on the compression of baseband signals.

**Lossless compression techniques**

- A comprehensive review of the current state of baseband signal compression as of 2024, highlighting recent advancements and identifying remaining challenges in the field.
- Study of the usage and performance of off-the-shelf algorithms such as gZIP or zSTD and their application in the compression of baseband signals.
- Examination of the limitations inherent to Linear Prediction Coding (LPC) approximations in the case of signals with a low number of samples per symbol.
- Development and evaluation of a novel neural network-based approach for lossless compression, demonstrating an increase in compression percentage compared to existing methods.

**Lossy compression techniques**

- A detailed study on the synergistic use of various compression methods, such as quantization and resampling, to optimize the balance between compression efficiency and signal fidelity.
- A proof-of-concept implementation that showcases the application of sparse coding techniques to enhance the compression ratio without significant loss of information.
- An in-depth analysis of the impact of noise on all DVB-S2 modcods and coding schemes, focusing on their compression capabilities and performance under different levels of quantization.

**Algorithm implementation and simulation**

- Provision of an open-source Python implementation of the Linde-Buzo-Gray (LBG) algorithm, optimized using the Just-In-Time compiler Numba. This implementation is noted to be the fastest and requires the least memory among current pure Python versions.
- Development of an open-source Python implementation of the Golomb-Rice compression algorithm, similarly optimized with Numba for enhanced performance.
- Creation of Python codes for various L0 sparse coding algorithms, including Orthogonal Matching Pursuit (OMP), Generalized Orthogonal Matching Pursuit (gOMP), Iterative Hard Thresholding (IHT) and Simultaneous Orthogonal Matching Pursuit (SOMP). Each optimized with Numba to ensure maximum efficiency.
- A comprehensive MATLAB®/Python simulation tool designed to accurately measure and compare the performance of different lossy compression algorithms.

## 1.5 Methodology

### 1.5.1 Notations and definitions

In this section, we will try to give an exact definition of terms which will be used along this manuscript, and some pieces of information regarding the choice of these definitions.

#### 1.5.1.1 Signal to noise ratio

Signal-to-Noise Ratio (SNR) for a baseband signal is a measure used in communication systems to assess the quality of a signal relative to the level of background noise. It is defined as the ratio of the power of the signal (useful information) to the power of the noise, typically expressed in Decibel (dB). A higher SNR indicates a clearer signal with less noise interference, leading to better communication quality.

In communication theory, the noise is usually supposed to be white and Gaussian. Defined as as Additive White Gaussian Noise (AWGN). **In the following chapters, we will always consider an AWGN noise.**

#### 1.5.1.2 Bit error rate (BER)

The Bit-error rate (BER) is a fundamental metric used in the field of telecommunications to assess the quality of a signal transmission system. It quantifies the frequency of errors that occur in a stream of bits transmitted over a communication channel [1], [26]. Specifically, BER is defined as the ratio of the number of erroneous bits to the total number of bits transmitted during a specified time interval. This parameter is crucial for evaluating the reliability and efficiency of communication systems. A lower BER indicates a more accurate and reliable transmission, signifying fewer errors and higher fidelity in the received signal compared to the original. BER is influenced by various factors, including the quality of the transmission medium, signal strength, noise levels, and the effectiveness of the error correction methods employed in the system. It serves as a critical benchmark for optimizing and comparing different communication technologies and protocols.

#### 1.5.1.3 Degradation

The notion of *degradation* is crucial in assessing the performance of different **lossy compression algorithms**. Degradation is specifically characterized as the difference, quantified in decibels (dB), between two Bit-error rate (BER) curves at a specified BER point, notably at $10^{-7}$. The selection of $10^{-7}$ as the BER benchmark value is supported by its importance in guaranteeing a strong level of data integrity and reliability in communication systems.

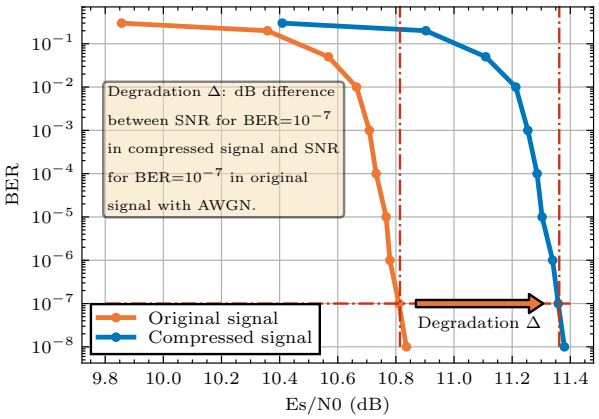



**Figure 1.4:** Degradation measurement (in *dB*)

This threshold strikes a balance between the demand for high-quality transmission and the practical limitations of communication channels. It represents a stringent criterion in which even a minor introduction of errors by a compression algorithm can be deemed unacceptable. Consequently, measuring the degradation at this BER value offers a precise gauge of the impact of lossy compression on the fidelity of the signal transmission. The smaller the degradation, the more effective the compression algorithm is in maintaining the integrity of the signal despite the reduction in data size. Thus, this metric

serves as a critical tool to assess and compare the performance of different lossy compression strategies, highlighting their efficiency in preserving signal quality amid data size reduction.

#### 1.5.1.4 Compression ratio

The compression ratio ($\mathcal{C}$) is a measure that indicates the effectiveness with which a compression algorithm reduces the size of the data. A widely accepted definition of the compression ratio is the proportion of the reduction in size achieved by the compression process, which can be expressed mathematically as follows:

$$\mathcal{C}\{y, \breve{y}\} = 1 - \frac{\text{output size in bits}}{\text{input size in bits}} = 1 - \frac{\text{Size of } \breve{y} \text{ (in bits)}}{\text{Size of } y \text{ (in bits)}} = 1 - \frac{\#\{\breve{y}\}}{\#\{y\}} \tag{1.1}$$

This definition is advantageous over alternative formulations, such as the simple ratio of output to input size, for several reasons. First, it provides a direct measurement of space savings, offering a clear and intuitive understanding of the benefit: the closer $\mathcal{C}$is to 1, the more substantial the size reduction and thus the savings in transmission costs and storage capacity. Moreover, this formulation lends itself to a more practical analysis of the trade-offs between compression and quality loss, which is crucial for lossy compression schemes where a balance between data fidelity and size reduction must be achieved. It is directly correlated with the quantity of interest, the percentage reduction, which aids in setting and comparing performance benchmarks for compression algorithms.

#### 1.5.1.5 Expansion factor

The expansion factor symbolized by $\mathcal{H}_F$, quantifies the bits' overhead introduced in the transmission process to recover one useful bit at the receiving end before Forward Error Correction (FEC) is applied. It can be calculated using:

$$\mathcal{H}_F = 2 \times \mathcal{F} \times N_q \tag{1.2}$$

For example, for a baseband Quadrature Phase Shift Keying (QPSK) signal with two samples per symbol at an 8-bit quantization level, the expansion factor would be $\mathcal{H}_F = 2 \times 2 \times 8 = 32$. This tells us that to deliver one useful bit of the original data (before modulation), 32 bits need to be transmitted between the antenna and the demodulator.

When a signal $y$ undergoes compression to a form $\breve{y}$ characterized by a compression ratio $\mathcal{C}$, the expansion factor of the compressed signal $\mathcal{H}_F\{\breve{y}\}$ is proportionately reduced in accordance with the formula:

$$\mathcal{H}_F\{\breve{y}\} = \mathcal{H}_F\{y\} \times (1 - \mathcal{C}\{y, \breve{y}\}) \tag{1.3}$$

This relation signifies that the expansion factor after compression directly reflects the reduction in transmission burden, making it a key factor in the design of efficient communication systems. Thus, the expansion factor serves as a crucial performance indicator in optimizing both the compression methods and the design of transmission schemes.

The subsequent chapters will further examine these metrics, shedding light on their impact on system performance and design decisions in the specific context of space telecommunications.

#### 1.5.1.6 Time factor

The Time Factor, denoted as $\mathcal{T}_F$, is an essential metric that quantifies the efficiency of a compression-decompression cycle in terms of time. The formal definition of the Time Factor is given by the following equation:

$$\mathcal{T}_F = \frac{\text{Compression Time} - \text{Decompression Time}}{\text{Compression Time} + \text{Decompression Time}} \tag{1.4}$$

This metric provides valuable information on the operational performance of a compression algorithm. A lower value of $\mathcal{T}_F$ ($< 0$) indicates a faster decompression process relative to the compression time. In contrast, a higher value $\mathcal{T}_F$ ($> 0$) suggests that decompression takes significantly longer than compression, which could be acceptable in scenarios where the decompression speed is less critical. The Time Factor thus serves as a key parameter for evaluating and comparing different compression methodologies, especially in systems where time efficiency is a major consideration.

## 1.5.2 Data sources

The authenticity and precision of the data utilized in the simulations are critical to the validity of any research on digital signal processing and telecommunications. This section delineates the data sources employed for the simulations presented in this thesis.

### 1.5.2.1 Synthesized signal data

The foundation data set comprises synthesized signal samples created using *MATLAB®*. A pristine, uncorrupted signal is generated, sampled at twice the symbol rate to facilitate adequate Nyquist sampling and subsequent processing stages. The resultant samples are quantified at 16 bit resolution and archived in *.mat* format files, each containing approximately 50 million samples. These samples serve as the primary corpus for simulation and are subject to various transformations, such as resampling and requantification, to replicate different signal processing scenarios.




**Figure 1.5:** MATLAB®, created by MATHWORKS®, is a commercial programming language with multiple paradigms and a numerical computing environment. It enables manipulation of matrices, plotting of functions and data, algorithm implementation, development of user interfaces, and interaction with programs in different languages.

For the generation of noise-afflicted signals, the same repository of unnoised signal files described above is utilized. To simulate real-world channel conditions, we superimposed white Gaussian noise on the signal samples. This noise overlay is conducted using the Python programming language, employing the *Numpy* library to generate Additive White Gaussian Noise (AWGN) with specific statistical properties aligned with theoretical models and practical requirements of telecommunication systems.

### 1.5.2.2 Real-life signal data

In addition to synthesized datasets, certain analyzes require the incorporation of real-life signal data. For this purpose, packet streams compliant with the Vita-49 standard are acquired through a *Nuron Engine* from *Safran Data Systems*. These signal packets provide a more tangible representation of in-field data conditions and are explicitly indicated in the respective simulation analyzes. The utilization of these real-time data sources allows for the validation of compression algorithms under genuine operational conditions and enhances the relevance of the simulation results.

## 1.5.3 The importance of shaping when compressing signals

Shaping in digital communication, often referred to as "signal shaping" or "pulse shaping," is a technique used to alter the waveform of digital signals for efficient transmission and to minimize potential interference and signal distortion. The primary goal of shaping is to make the transmitted signal more suitable for the physical communication channel, ensuring reliable data transmission with minimal errors.

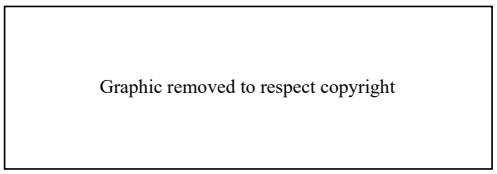



**Figure 1.6:** A Nuron Engine architecture.
**Source:** *Safran Data Systems*

Signal shaping encompasses several crucial aspects that significantly enhance the performance of the communication system. First, **it focuses on bandwidth efficiency by meticulously designing the signal form** to optimally utilize the available bandwidth. This efficient use of bandwidth is achieved through precise signal tailoring, ensuring that signals fit neatly within the designated spectral space. Second, signal shaping plays a **pivotal role in reducing Inter-Symbol Interference (ISI)**. ISI is a phenomenon in which signal overlap occurs, leading to confusion and errors at the receiving end. Through shaping, signals are crafted in a way that minimizes or entirely negates this overlap, thus mitigating ISI's detrimental effects. Lastly, signal shaping is tailored to match the specific characteristics of a communication channel, which significantly boosts performance. This aspect is particularly beneficial in channels with unique attributes, such as frequency-dependent loss. When signals are shaped to counteract these specific losses, the overall integrity and efficiency of the communication process are improved.

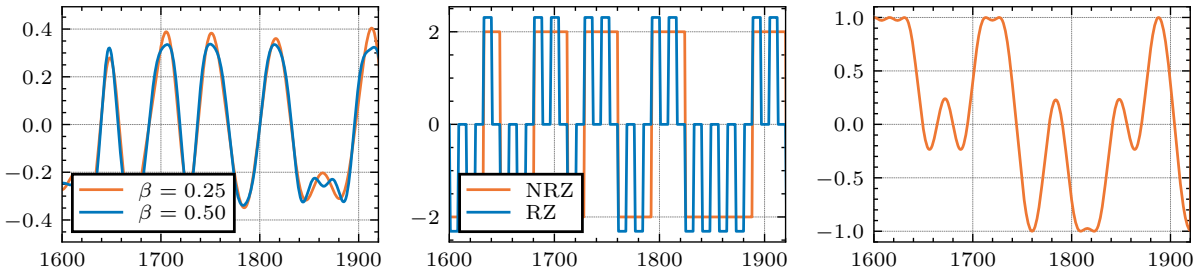
Examples of Shaping Techniques:

1 · **Raised Cosine Filtering**: This is a popular shaping technique that is used to limit the bandwidth of a signal and reduce ISI. The raised cosine filter smoothly transitions the signal between different symbol states, ensuring minimal overlap. The parameter $\beta$ is used to control the filter roll-off factor, which is a measure of the excess bandwidth of the filter, , *i.e.* the bandwidth occupied beyond the Nyquist bandwidth (Figure 1.7a). The frequency response of the raised cosine filter is given by the following equation:

$$H(f) = \begin{cases} 1, & |f| \leq \frac{1-\beta}{2T} \\ \frac{1}{2}\left[1 + \cos\left(\frac{\pi T}{\beta}\left[|f| - \frac{1-\beta}{2T}\right]\right)\right], & \frac{1-\beta}{2T} < |f| \leq \frac{1+\beta}{2T} \\ 0, & \text{otherwise} \end{cases} \tag{1.5}$$

2 · **Gaussian filtering**: Used in Gaussian Minimum Shift Keying (GMSK), this technique shapes the signal with a Gaussian filter, resulting in smooth phase transitions and efficient bandwidth usage (Figure 1.7c).

3 · **Non return to zero (NRZ) and Return to zero (RZ) shaping**: These techniques alter the signal waveform to improve signal detection and reduce ISI. NRZ signals maintain a constant voltage level during the symbol period, while RZ signals return to zero at the midpoint of each symbol period (Figure 1.7b).




**(a)** RRC shaping **(b)** NRZ/RZ shaping **(c)** Gaussian shaping

**Figure 1.7:** Visualization of various shaping for a dataset represented on a Binary Phase Shift Keying (BPSK) constellation.

By spreading signal energy, shaping actually plays an important role in the ways in which compression algorithms can be applied to the data. In fact, the more data is spread, the less the compression algorithm will be able to find patterns in the data. This is why we will categorize the signals by their

shaping.

Moreover, depending on the shaping applied, certain compression algorithms demonstrate varying levels of efficiency. For instance, we will observe that Lempel-Ziv algorithms perform well on NRZ/RZ signals, while their efficiency tends to diminish with other shaping types, such as RRC.

### 1.5.4 Techniques and tools

This section outlines the computational tools and libraries utilized in our research to develop and evaluate the proposed compression and prediction models.

- The development and testing of algorithms were primarily conducted using a custom-built **Matlab®/Python simulator**. This simulator facilitated the exploration of different compression strategies and their effects on signal fidelity and compression efficiency.
- The **SPAMS library** [27], [28] from Julien Mairal provided robust optimization algorithms, which were instrumental in sparse coding and dictionary learning tasks integral to our compression techniques.
- For optimization of Python performance, the **Numba library** was used [29]. Numba enabled just-in-time compilation of Python code to machine code, significantly speeding up the execution of computationally heavy parts of our algorithms.
- Neural network models were implemented and trained using the **PyTorch library** [30]. PyTorch's dynamic computation graph and extensive library of pre-built layers and optimization algorithms greatly facilitated the development of complex predictive models.
- The conformity of real-life signal data with the **Vita-49 standard** [6] was ensured for all data sets used in this study. This standardization was crucial for validating the applicability of our models to practical satellite communication systems.

## 1.6 Limitations and scope

Although comprehensive in its exploration of novel compression algorithms for satellite communications, this manuscript acknowledges some limitations that frame its scope and interpretations. The primary constraint lies in the research's concentration on theoretical foundations and simulation-based evaluations of the proposed compression methods.

The use of a MATLAB®/Python-based simulation environment, as detailed in earlier sections, serves as a robust platform for testing and validating algorithms under controlled conditions that strive to emulate the complexities of satellite communication systems. However, the intrinsic nature of simulations means that they can only approximate, rather than fully replicate, the **myriad of variables and unpredictable factors present in real-world operational environments**. This discrepancy between the simulated and actual conditions introduces **a layer of uncertainty about the performance and efficiency of algorithms** when implemented in field settings.

Moreover, the absence of empirical validation using real field data from satellite transmissions or Earth Observation (EO) datasets further constrains the breadth of our findings. While simulations allow for a detailed and nuanced understanding of algorithmic behaviors under predefined scenarios, they cannot capture the full spectrum of challenges and anomalies that might arise in live operational contexts. Such challenges include, but are not limited to, signal degradation due to atmospheric conditions, hardware limitations, and the dynamic nature of space weather.

Therefore, while the results presented here offer valuable insight into the potential benefits and efficiencies of the proposed compression techniques, they should be interpreted with caution. **The findings serve as a preliminary step toward understanding the applicability and reliability of these algorithms in satellite communications**, which requires further research that includes empirical testing and real-world deployment.

Future work will aim to bridge this gap by incorporating field data into the validation process and exploring partnerships with satellite operators and other stakeholders in the space industry. This approach will not only enhance the robustness of the research, but also ensure that the algorithms are tested and refined in the context of actual operational demands and constraints.

## 1.7 Structure of the document

This document is structured in three main parts. Each part focuses on a specific set of techniques and tools, and contains two or three chapters.

The first part is dedicated to two "classical" compression techniques that have been used in the industry. These are mainly methods based on common techniques of digital signal processing. **This part will be our reference for the rest of the document**.

The second part is dedicated to the study of "dictionary"-based compression methods: these methods try to find a good representation of the signal in a dictionary of atoms. **This part will be our main contribution to the field of lossy compression**.

The third part is dedicated to the study of predictive compression, or prediction by partial matching. These methods are based on the idea that the signal can be predicted from its past and that the difference between the prediction and the real signal can be compressed efficiently. **This part will be our main contribution to the field of lossless compression**.

Finally, a conclusion will summarize the main results of the document and provide some perspectives for future work. Some research is also presented in the annexes to give more details on some specific points. They are referenced in the main document and can be read independently.

Chapter 2

# STATE-OF-THE-ART

***Things not to do again***


2.1 Entropy coding . . . . . **46**
2.1.1 General algorithmic principles . . . . . 46
2.1.2 Universal codes . . . . . 47
2.2 Dictionary compression . . . . . **49**
2.3 Quantization based compression . . . . . **50**
2.3.1 Uniform quantization . . . . . 50
2.3.2 Non-uniform & vector quantization . . . . . 51
2.3.3 Adaptive quantization . . . . . 51
2.3.4 Quantization of complex-valued data . . . . . 51
2.3.5 Companding techniques: $\mu$-law & A-law . . . . . 52
2.4 Audio compression methods . . . . . **53**
2.4.1 Lossless compression techniques . . . . . 53
2.4.2 Lossy compression techniques . . . . . 55
2.5 Predictive coding . . . . . **56**
2.5.1 Background . . . . . 56
2.5.2 Formal approach . . . . . 56
2.5.3 Linear predictive coding (LPC) . . . . . 57
2.6 Emerging techniques . . . . . **58**
2.6.1 Sparse coding . . . . . 58
2.6.2 Compressed sensing . . . . . 59
2.7 Neural-network based compression for digital communication signals . . . . . **61**
2.8 Upcoming challenges . . . . . **61**


Within the constantly progressing domain of space telecommunications, the effective use of bandwidth is crucial, driving the need for ongoing improvements in data compression methods. The constant increase in data requirements, together with the physical limitations set by space activities, necessitates the creation of advanced algorithms that can diminish data rates without sacrificing the authenticity and dependability of the conveyed data. This section offers an exhaustive review of the current cutting-edge data compression technologies, approaches, and norms relevant to the transmission of RF sampled signals over IP networks.

An exhaustive exploration of the existing literature reveals a multifaceted approach to data compression, ranging from theoretical foundations to practical implementations. It encompasses a wide array of strategies, including entropy coding, lossless and lossy compression algorithms, the latest advances in predictive coding, and some aspects and fields of neural networks. Furthermore, the role of hardware implementations, particularly in Field-Programmable Gate Array (FPGA) technology, is examined to understand their impact on real-time data processing in space environments.

## 2.1 Entropy coding

### 2.1.1 General algorithmic principles

Entropy coding is a compression strategy that is based on the statistical model of the source data. The foundation of entropy coding is the concept of entropy, which in information theory is a measure of the unpredictability or information content.

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

**Essential concept of Information Theory**

In information theory, a dataset characterized by a lower entropy presents a higher possibility of compression. The minimum theoretical number of bits required to compress each element of the dataset is given by $H(X)$.

Entropy coding techniques aim to reach the maximum compression potential defined by theory.

The principle of entropy coding is then to **assign shorter codes to more frequent elements and longer codes to less frequent elements**. This approach is rooted in the entropy concept defined by Claude Shannon, which quantifies the average information content per data unit.

Two widely adopted entropy-coding techniques are **Huffman coding** and **Arithmetic coding**. These methods are presented below.

| **Symbol** | **Encoding** | Size **(in bits)** |
|---|---|---|
| -5 | 1101010000 | 10 |
| -4 | 110101001 | 9 |
| -3 | 110100 | 6 |
| -2 | 1100 | 4 |
| -1 | 10 | 2 |
| 0 | 0 | 1 |
| 1 | 111 | 3 |
| 2 | 11011 | 5 |
| 3 | 1101011 | 7 |
| 4 | 11010101 | 8 |
| 5 | 1101010001 | 10 |

**Table 2.1:** Huffman encoding dictionary of Gaussian Data between $-5$ and 5

**Huffman Coding** [31] is a popular method that uses a variable-length code table to encode a source symbol (such as a character in a file). The method constructs the code table based on the frequency of occurrence of each symbol in the source data. A distinct binary code is assigned to each symbol, with shorter codes for more frequent symbols. This method ensures that the overall encoded data size is nearly minimal, making it very efficient for data with known symbol frequencies.

**Arithmetic Coding** Unlike *Huffman coding* [32] processes all data as a unit rather than encoding each symbol independently and converts all input data into a single floating point number between 0 and 1. Although this method is theoretically more efficient than Huffman coding, **it is more complex and computationally intensive, leading to its less frequent use in practice.**

Both methods require the usage of a probability dictionary, which assigns to each word of the dataset a likelihood of occurrence. This dictionary can be computed before the compression –*offline*– or during the compression process –*online*–. In the latter scenario, **the probability dictionary must be exchanged between the compressor and the decompressor, which increases the necessary bandwidth**. It should be noted that separate compression strategies and methods can be employed to convey the probability dictionary effectively.

| | Huffman Coding | Arithmetic Coding |
|---|---|---|
| **Pros** | ■ **Simplicity and effectiveness** for many types of data.<br>■ Deterministic decoding process which is **easy to implement**.<br>■ **Well suited for hardware implementations** due to its table-driven approach. | ■ Can result in **closer approximation** to the entropy limit, especially for large sequences.<br>■ It does not require a prefixed code table, which can be adapted to the actual input sequence. |
| **Cons** | ■ **Not as efficient as arithmetic coding** for sources with a large number of symbols or where the symbol frequencies are very similar.<br>■ The need for a known frequency distribution or an additional pass to determine frequencies can be a **drawback for some real-time applications**. | ■ More susceptible to **rounding errors** due to the use of floating-point arithmetic.<br>■ Higher computational complexity and potentially **slower execution** due to arithmetic operations. |

**Table 2.2:** Comparison between Huffman Coding and Arithmetic Coding

In summary, according to Table 2.2, *Huffman coding* is more appropriate for settings where there are restrictions on latency and speed, while *Arithmetic coding*, which requires more resources, is more suitable for scenarios where high performance is the priority [33].

**Compression fundamental principle**

Any compression/decompression process invariably requires the exchange of mutual understanding between the compressor and the decompressor. This understanding can be disseminated *offline* (prior to compression) or *online* (subsequent to compression)

### 2.1.2 Universal codes

Universal codes are a family of entropy-coding methods that do not require *a priori* knowledge of the probability of symbols. These codes are particularly useful for scenarios where the distribution of source symbols is not known in advance or cannot be accurately estimated. But satisfy some general properties –like, centered around 0–. Universal coding techniques are adaptable and can efficiently encode integers or lengths, often used in conjunction with other coding methods.

Examples of universal codes include the Levenshtein coding [34], [35], Elias codes –gamma, beta & omega– [36], the Golomb code [37], and the Rice code [38], [39]. All of these codes are designed to be simple and efficient, making them suitable for a wide range of applications.

**Elias codes** such as the Elias gamma code and the Elias delta code, are used to encode a sequence of positive integers without prior knowledge of the size or distribution of the integer.

**Algorithm 1** Elias Gamma Encoder

**Require:** $n$: positive integer
1: $q \leftarrow \lfloor \log_2 n \rfloor$
2: Write $q$ zeroes
3: Write binary representation of $n$ with $q$ bits

**Algorithm 2** Elias Gamma Decoder

1: Read and count consecutive zeroes until a 1 is encountered, let $q$ be the count
2: Read the next $q$ bits and convert them to an integer, let it be $n$
3: Output $2^q + n$

**Algorithm 3** Elias Delta Encoder

**Require:** $n$: positive integer
1: $q \leftarrow \lfloor \log_2(n+1) \rfloor$
2: Write Elias Gamma code for $q$
3: Write binary representation of $n$ with $q$ bits

**Algorithm 4** Elias Delta Decoder

1: Read and decode Elias Gamma code to get $q$
2: Read the next $q$ bits and convert them to an integer, let it be $n$
3: Output $2^q + n - 1$

**Golomb codes** are particularly efficient for encoding nonnegative integers with a geometric-like distribution. They are widely used in applications where the probability distribution of the source symbols is known or can be approximated.

**Algorithm 5** Golomb Encoding

**Require:** $n$: nonnegative integer, $m$: positive integer (parameter)
1: $q \leftarrow \lfloor \frac{n}{m} \rfloor$
2: Write unary code for $q$
3: Write binary representation of $n \mod m$ with $\lceil \log_2 m \rceil$ bits

**Algorithm 6** Golomb Decoding

**Require:** Unary code and binary representation of the Golomb-coded integer
1: Read and count consecutive zeros until a 1 is encountered, let $q$ be the count
2: Read the next bits $\lceil \log_2 m \rceil$ and convert them to an integer, let it be $r$
3: Output $q \cdot m + r$

**Rice codes** are a particular case of the Golomb codes where $m$ is a power of two. It offers a more flexible approach to encoding integers. They are particularly useful for fast integration (hardware or software) since they directly manipulate bit representation of data.

**Algorithm 7** Rice encoder

**Require:** $s$: integer, $k$: integer
1: $m \leftarrow 2^k$
2: Output $\leftarrow \lfloor \frac{s}{m} \rfloor$ using unary code ($\lfloor \frac{s}{m} \rfloor + 1$ bits)
3: Output $\leftarrow$ the $k$ least significant bits of $s$

**Algorithm 8** Rice Decoder

**Require:** $s$: integer (number of consecutive ones in the input), $x$: next $k$ input bits
1: $s \leftarrow s \times 2^k + x$

In summary, entropy-coding techniques are fundamental to general data compression. Huffman and Arithmetic coding are principal methods with their specific application scenarios, while universal codes offer adaptable solutions when probability distributions are unknown or nonuniform.

## 2.2 Dictionary compression

Dictionary compression methods are among the most versatile and widely used algorithms for data reduction. They operate on the principle of **substituting repeating strings or bytes with shorter reference codes derived from a dictionary**.

The effectiveness of dictionary methods is especially pronounced in datasets with considerable redundancy and byte-level patterns.

**Lempel-Ziv (LZ) family algorithms [40], [41]** are at the core of various dictionary-based compression techniques. They are inherently suited for binary data compression due to their agnostic approach to data interpretation, treating input as a stream of bytes rather than encoded text characters.




**Figure 2.1:** Logo of gZIP. gZIP is based on the deflate algorithm, which is a combination of the LZ77 and Huffman algorithms

**Algorithm 9** LZ77 Encoding Algorithm

```
Initialize search buffer and look-ahead buffer
while there is unprocessed input do
    Find the longest match between the buffers
    if match found then
        Output ⟨distance, length, next symbol⟩
    else
        Output ⟨0, 0, symbol⟩
    end if
    Slide the window forward
end while
```

**Algorithm 10** LZ78 Encoding Algorithm

```
Initialize the dictionary with the empty string
Initialize current string S to empty
for each symbol c in the input do
    if S + c is in the dictionary then
        S = S + c
    else
        Output the code for S and add S + c to the dictionary
        S = c
    end if
end for
if S is not empty then
    Output the code for S
end if
```

The choice of dictionary compression technique depends largely on the specific requirements for **compression speed, decompression speed, memory usage, and the nature of redundancy within binary data**. Numerous libraries that support these algorithms are available, including the popular 7-Zip [42] and its enhanced variant by Reichardt [43].

## 2.3 Quantization based compression

Quantization plays a crucial role in data compression, as it involves the conversion of a wide range of input values into a more compact set. This is a critical step in digital signal processing, where continuous signals are transformed into discrete symbols. In terms of data compression, quantization effectively reduces the bit count needed to store transmitted data by approximating a signal to a limited number of levels. Numerous quantization techniques exist [24], [25], each with its own advantages and disadvantages in terms of complexity and degradation.

> Quantization techniques are really simple and known techniques, and have been used for a long time in the domain of digital signal processing. The effect of quantization is well documented and studied [44]–[46], and its application, along with other traditional methods, has been studied for *RF over IP* applications [23].

### 2.3.1 Uniform quantization

Uniform quantization is the simplest form of quantization, where the range of signal values –typically considered as $[-1\ ;\ 1]$– is divided into intervals of equal size. The quantization process can be expressed mathematically as follows:

**Mid-riser quantization**

$$Q(x) = \Delta \left\lfloor \frac{x}{\Delta} \right\rfloor$$

**Mid-tread quantization**

$$Q(x) = \Delta \left\lfloor \frac{x}{\Delta} + \frac{1}{2} \right\rfloor$$

where $\Delta$ is the step size of the quantizer, $x$ is the input signal, and $Q(x)$ is the quantized output. Step size is related to bit depth of quantization, which, in turn, affects the Signal-to-Noise Ratio (SNR) of the quantized signal. Figure 2.2 illustrates the quantization process for a 4-bit uniform quantizer.

After quantization, the signal is represented by a finite number of words. In digital applications, it is common to take a number of words equal to a power of two, $2^{N_q}$.

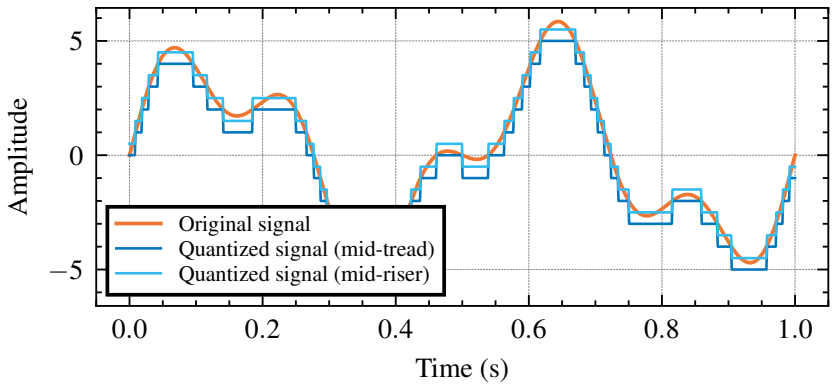



**Figure 2.2:** Illustrations of 4-bit uniform quantization of a signal.

### 2.3.2 Non-uniform & vector quantization

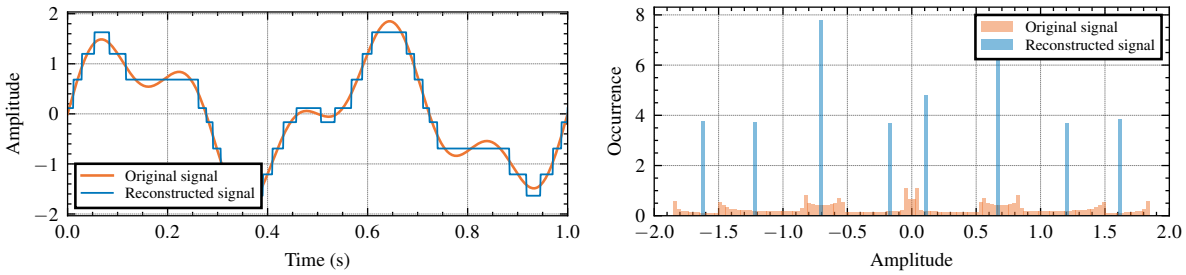
Non-uniform quantization is engineered to process signals more effectively, using quantizers tailored to the data that we aim to quantize. This procedure requires a profound understanding of the data features (histogram, moments, etc.) to guarantee satisfactory results.

Vector Quantization (VQ) extends the concept of scalar and nonuniform quantization by encoding blocks or vectors of samples rather than individual samples. VQ operates by partitioning the multidimensional space into regions, each represented by a code vector in a codebook. The process of creating an effective codebook is critical and can be performed using the Linde-Buzo-Gray (LBG) [25], [47] algorithm or K-means clustering.

The LBG algorithm, also known as the *generalized Lloyd algorithm*, iteratively adjusts the code vectors to minimize the distortion, which can be expressed as:

$$D = \sum_{i=1}^{N} \sum_{x \in R_i} \|x - q_i\|^2 \tag{2.2}$$

where $N$ is the number of code vectors, $R_i$ is the region corresponding to the $i$-th code vector $q_i$, and $D$ is the total distortion.




**(a)** Original vs reconstructed signal

**(b)** Histogram of original and reconstructed signal

**Figure 2.3:** Illustrations of 3-bit nonuniform vector quantization of a signal. A codebook has been precomputed using the LBG algorithm [47].

### 2.3.3 Adaptive quantization

Adaptive quantization [24], [25] dynamically changes the size of the quantization step to match the varying statistics of the input signal. This is particularly useful for signals with a large dynamic range or varying statistical properties over time.

### 2.3.4 Quantization of complex-valued data

When dealing with complex-valued data, such as 2D I/Q data in telecommunication systems, both the real components (I, for in-phase) and imaginary components (Q, for quadrature) must be quantized. Quantization can be applied independently to each component using the techniques mentioned above. **However, the correlation between the I and Q components can be exploited for more efficient compression using joint quantization schemes.**

### 2.3.5 Companding techniques: $\mu$-law & A-law

The $\mu$-law algorithm is a companding technique commonly used in digital communication systems to optimize the dynamic range of an audio signal. Companding, a portmanteau of compressing and expanding, involves compressing the signal's dynamic range before transmission and expanding it back to its original state at the receiver. The primary aim of the $\mu$-law algorithm is to reduce the effects of noise and variations in signal strength, particularly in environments with a wide range of audio amplitudes. By applying a logarithmic function to the amplitude of the audio signal, the $\mu$-law effectively reduces the dynamic range, making quieter sounds more prominent while limiting the volume of louder sounds. This nonlinear approach ensures that signals of lower amplitude are encoded with higher resolution, improving the overall Signal-to-Noise Ratio (SNR) and making the technique highly suitable for telecommunications, especially in analog-to-digital conversion processes.

Beyond its traditional application in telecommunications to improve voice quality, the $\mu$-law algorithm can be adapted ingeniously for nonuniform quantization in various digital signal processing applications. By rounding the output of the compression of the $\mu$-law, one can achieve nonuniform quantization effectively. **This method allows for the allocation of more bits to represent lower amplitude signals**, thus enhancing the precision and fidelity of the reconstructed signal. This innovative adaptation of the $\mu$-law not only extends its utility beyond traditional voice transmission, but also opens new avenues for efficient data representation in fields such as audio processing, image compression, and beyond, where managing dynamic range with high efficiency is paramount. $\mu$-law process (compression and decompression) can be described as follows:

$$F(x) = \operatorname{sgn}(x)\frac{\ln(1+\mu|x|)}{\ln(1+\mu)}, \quad -1 \leq x \leq 1 \tag{2.3}$$

$$F^{-1}(y) = \operatorname{sgn}(y)\frac{(1+\mu)^{|y|}-1}{\mu}, \quad -1 \leq y \leq 1. \tag{2.4}$$

Here, $\mu$ is a parameter that defines the compression curve and $\operatorname{sgn}(x)$ is the sign function. A-law has a similar formulation and is preferred in European systems. It is not directly a quantization method; however, by modifying it a little, we can actually transform it into a really efficient method for Gaussian-like signals.

$\mu$-law has been one of the algorithms studied in the new compression algorithm scheme, notably by the O-RAN alliance [10, Annex A.3].

## 2.4 Audio compression methods

Exploring the domain of audio data compression offers significant insight even for fields that are not directly related to audio processing. The extensive research and development dedicated to audio compression algorithms over the years have culminated in a variety of techniques that target different aspects of signal processing, **making it a rich source of inspiration**. This relevance is particularly pronounced in our context, where efficient data handling is crucial. Audio compression methods, with their nuanced approach to reducing data size while preserving or selectively discarding information, provide a framework that can be adapted to optimize data transmission and storage solutions in telecommunications and aerospace systems. Exploring these techniques not only helps to grasp the basic concepts of signal compression but also ignites new strategies for managing large data volumes in various digital communication contexts.

### 2.4.1 Lossless compression techniques

Lossless audio compression algorithms ensure the reduction of audio file size without compromising any bit of the original information content. They are indispensable for high-fidelity audio applications, including archival purposes and professional-grade music production, where the exact original sound must be preserved. Most of the codecs presented in this section are extracted from the Audio Codec comparison from Martjin Van Beurden [49] and Fathiah Muin [50].

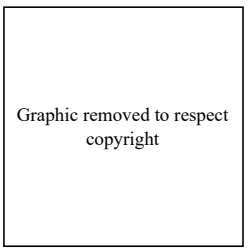



**Figure 2.4:** Logo of FLAC audio codec. The FLAC codec is a lossless audio codec that uses predictive LPC coding [48] combined with a Rice entropy coder [38] to compress the audio stream.
**Source:** *xiph.org. Open-source image*

**FLAC (Free Lossless Audio Codec) [51], [52]** : FLAC employs linear prediction to convert the audio waveform into a more compact form and then encodes the residuals, the differences between the prediction and the actual signal, resulting in a significant reduction in file size. The FLAC predictor can be considered as a simplified form of Linear Prediction Coding (LPC) [48], which estimates future samples based on past audio samples. This technique, coupled with variable block sizes, adaptive entropy coding using Rice codes [38], and partitioned Rice coding, contributes to FLAC's versatility and effectiveness. Although originally conceived for audio, the basic approach of FLAC could be adapted for any signals that exhibit regular patterns and where a prediction model could be established, such as in the compression of time-series data in various scientific applications.

**ALAC (Apple Lossless Audio Codec) [53]:** ALAC, developed by Apple, is a proprietary lossless audio compression format that, like FLAC (Free Lossless Audio Codec), compresses audio files without quality loss. However, while both codecs employ prediction and Rice coding techniques, they differ significantly in terms of accessibility and compatibility. FLAC is an open-source standard widely supported across various platforms, whereas ALAC is deeply embedded within Apple's proprietary ecosystem. This makes ALAC particularly suited for Apple devices and services, where it can be used natively without additional software.

**WavPack [54], [55]:** WavPack offers a unique hybrid mode that creates a compressed lossy file and a correction file to recover lossless audio. It applies advanced compression techniques, including dynamic noise shaping and joint stereo modes, to increase compression efficiency. The core principle of creating a base approximation and then applying a correction layer is applicable not only to audio data, but also to

other data types where a high-fidelity reconstruction is necessary, such as in certain medical imaging or scientific measurement data.

**Monkey's Audio [56]:** This codec provides high compression efficiency through its optimized prediction and entropy coding mechanisms. Monkey's Audio uses a highly refined prediction algorithm that can be adjusted based on the type of audio being compressed, which allows it to achieve higher compression ratios than some other codecs, though often at the expense of increased encoding time. Similar predictive techniques can be applied to other signal types that demonstrate periodic or predictable behavior over time or space.

**OptimFROG [57]:** Designed for achieving high compression ratios, OptimFROG meticulously analyzes the audio signal to enable extreme compression with an excellent decompression speed. The codec's strength lies in its detailed analysis of audio data and error correction capabilities, suggesting its suitability for complex signals across various domains requiring detailed component separation.

**SHORTEN [58]:** SHORTEN is one of the earlier lossless audio codecs and was once widely used for trading live music recordings. It applies a form of LPC to simplify the signal before encoding it. The codec is less complex than newer algorithms and does not achieve the same compression ratios, but it laid the groundwork for many principles used in current codecs. Although it has largely been superseded by more modern formats like FLAC, the fundamental principles behind SHORTEN, such as LPC and fixed prediction, can be effectively utilized in other areas where legacy support or lower computational overhead is desired.

These lossless compression techniques leverage the redundancy inherent in audio signals, but are not confined to them. Their principles are broadly applicable to any type of data with similar redundancy patterns, including a wide spectrum of digital signals encountered in telecommunications, remote sensing, and scientific research.

### 2.4.2 Lossy compression techniques

While lossless compression techniques aim to preserve the exact original data, lossy compression techniques accept some level of data loss to achieve higher compression ratios. Typically, audio compression algorithms are designed specifically for audio data because **they take advantage of the characteristics of human auditory perception**. This specialization allows them to remove audio components that are less likely to be perceived by the ear, thus achieving more significant data reduction while maintaining acceptable audio quality. However, understanding the principles behind these techniques can provide information on general data reduction strategies and inform the development of compression algorithms for other types of data.

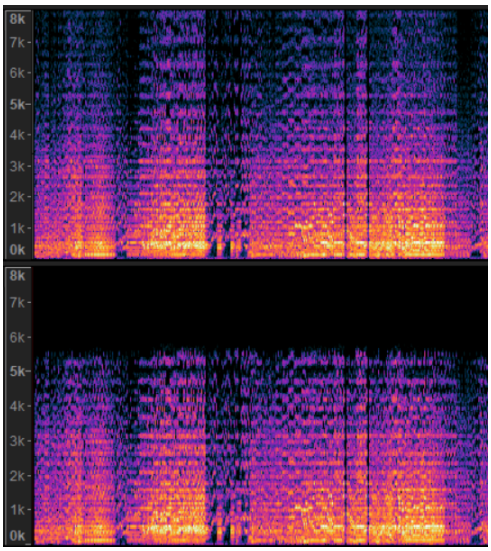



**Figure 2.5:** Spectrogram comparison of the same music track encoded in FLAC (top) and MP3 at 48 kbps (bottom). The MP3 spectrogram reveals a significant reduction in frequency range and detail, particularly above 5 kHz, illustrating the effects of lossy compression at a lower bitrate.

**Psychoacoustic Models:** The majority of lossy audio compression algorithms, such as MP3 [59] and AAC [60], [61], use psychoacoustic models to determine **which parts of an audio signal can be removed without noticeably affecting the listening experience**. These models are based on the perceptual limitations of human hearing, such as auditory masking, where certain sounds are perceived as inaudible in the presence of louder dominant frequencies. By discarding such masked sounds, algorithms can significantly reduce the amount of data needed to represent the audio without compromising the perceived quality.

**Transform Coding:** Techniques such as the Modified Discrete Cosine Transform (MDCT) used in MP3 and AAC, convert the time-domain audio signals into the frequency domain, where it is easier to identify and discard less perceptible frequencies. The efficiency of transform coding lies in its ability to represent the signal compactly in the frequency domain, which is closely aligned with the way humans perceive sound.

**Bitrate Reduction:** Algorithms like Ogg Vorbis [62] and Opus [63] offer variable bitrate encoding, which adjusts the level of compression based on the complexity of the audio signal and the desired quality. This adaptability makes these codecs versatile and efficient, as they can dynamically allocate more bits to complex segments of audio while compressing simpler sections more aggressively.

**Temporal Noise Shaping (TNS):** Used in codecs such as AAC, TNS is a technique that applies noise shaping in the time domain to reduce the perceptual impact of quantization noise in the signal. It manipulates the spectral distribution of noise, directing it to the less critical parts of the audio spectrum where it will be less noticed.

The underlying principles of these lossy audio compression techniques, such as perceptual modeling, transform coding, and noise shaping, are not restricted solely to audio and can be applied to other signal processing applications. For example, similar strategies might be used to compress visual data by taking advantage of the characteristics of the human visual system, or any other data type where human perception can influence the acceptable level of data reduction.

## 2.5 Predictive coding

### 2.5.1 Background

Predictive coding, also known in specific contexts as Prediction by Partial Matching (PPM) [64], [65], represents a class of algorithms primarily used in lossless data compression –originally designed for text compression–. At its core, predictive coding relies on the concept of using historical data to predict future values in a data sequence. This prediction is then used to encode the data more efficiently by representing deviations from the predicted values rather than directly encoding the raw data. In essence, if the prediction is accurate, the deviation (or error) will be smaller, and hence require less space to encode.

The mechanism underlying predictive coding is based on the statistical tendencies of data sequences, be they audio signals, image pixels, or textual content. By constructing a model **based on the previous data points, the algorithm attempts to predict the upcoming values**. The actual data are then compared with these predictions and only the difference, referred to as the "residual" or the "prediction error," is stored or transmitted –typically after processing through an entropy coder–.

The roots of predictive coding go back to the early days of information theory and digital communication. However, it was not until the advent of digital computing and data storage that predictive coding techniques found widespread application. In the late 1950s and early 1960s, research into predictive coding began in earnest, influenced by the burgeoning field of digital signal processing. One of the first applications was the predictive coding of speech signals for transmission over telecommunication networks, an area where bandwidth was at a premium.

The 1970s and 1980s saw significant advances, with the development of adaptive predictive coding techniques where the prediction model could change based on the characteristics of the incoming data. This period also witnessed the emergence of the PPM algorithm, developed by John Cleary and Ian Witten in 1984 [64]. PPM was a groundbreaking method in the field of text compression, **using context modeling and adaptive prediction to achieve superior compression ratios**.

The subsequent decades have seen predictive coding evolve with the advent of more sophisticated statistical models and computational techniques. Its application has expanded into numerous fields, including image and video compression. **Today, predictive coding is an integral part of modern data compression techniques, reflecting a continuous interplay between theoretical development and practical application. The field remains vibrant, with ongoing research aimed at improving prediction accuracy, reducing computational complexity, and exploring new areas of application.**

### 2.5.2 Formal approach

A predictor is a function, denoted as $\mathcal{P}$, that can provide a good estimation of the next sample of a signal, given the knowledge of the last W samples of the same signal:

$$\forall k, \quad y_k = \underbrace{\mathcal{P}\left(y_{k-1}, \dots y_{k-\mathrm{W}}\right)}_{\tilde{y}_k} + \varepsilon_k \tag{2.5}$$

$(\varepsilon_k)_{k\in\mathbb{N}}$ is called the residual signal, and $(\tilde{y}_k)_{k\in\mathbb{N}}$ is the predicted signal. The value of W usually depends on the system studied and the type of predictor.

The objective of this compression system **is to apply entropy coding to the residual signal** $(\varepsilon_k)_{k\in\mathbb{N}}$ instead of $(y_k)_{k\in\mathbb{N}}$ and transmit the encoded data as compressed output. This method is effective because it can achieve **a reduction of signal entropy if the prediction is accurate**. As a result, some

information is effectively removed from the original signal, leading to a lower overall entropy compared to the original signal $(y_k)_{k\in\mathbb{N}}$ (Figure 2.6).

The challenge with this type of compressor lies in identifying effective predictors while keeping deployment and implementation costs low.

### 2.5.3 Linear predictive coding (LPC)

Linear Prediction Coding (LPC) is a method that is widely used in speech compression and other audio processing applications, utilizing the properties of linear prediction. A linear predictor in this context can be described mathematically by the following equation:

$$\tilde{y}_k = \sum_{i=1}^{\mathrm{W}} a_i y_{k-i}, \tag{2.6}$$

where $\tilde{y}_k$ represents the predicted value of the signal at time $k$, W is the number of past samples considered (the predictor order), and $a_i$ are the predictor coefficients, which are determined so as to minimize the prediction error in a least squares sense.

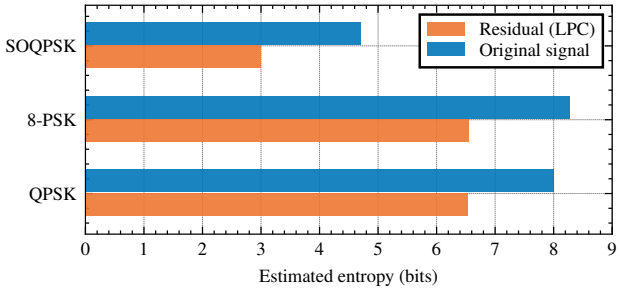



**Figure 2.6:** Estimated entropy of the original signal and its residual, computed with an LPC-based predictor. In this simulation, each signal sample is encoded using 16 bits, and all signals are transmitted using $\mathcal{F} = 2$ samples per symbol. The analysis is performed using MATLAB®.

LPC operates by estimating the coefficients $a_i$ that model the current sample as a linear combination of past samples. These coefficients are then used to predict future samples and the difference between the actual and predicted samples (prediction error) is encoded. By effectively modeling the signal with appropriate coefficients, LPC can significantly reduce redundancy in the signal, leading to efficient compression.

**The process involves several steps:**

1 · Analyze the input signal to derive the short-term spectral characteristics.

2 · Estimate the coefficients $a_i$ that minimize the mean squared error between the actual signal and its linear prediction.

3 · Use these coefficients to predict future values of the signal.

4 · Encode the prediction error, which is the difference between the actual signal and its prediction.

The predictive model in LPC assumes that the signal is stationary in short segments, which is a reasonable assumption for speech signals. This technique not only provides a highly efficient prediction, but also forms the basis for more complex codecs that can handle a broader range of audio applications by adapting the predictor coefficients dynamically. It is the fundamental method of compression used in both FLAC and SHORTEN audio codecs.

## 2.6 Emerging techniques

### 2.6.1 Sparse coding

Sparse coding is based on the principle that natural signals can be represented as a sparse linear combination of basis elements from a *overcomplete* dictionary. This principle is essential in various applications, including signal processing, image recognition, and machine learning. The mathematical model for sparse coding can be described as follows:

$$\boldsymbol{y} \approx \boldsymbol{\mathcal{D}}\boldsymbol{x} \tag{2.7}$$

where $\boldsymbol{y} \in \mathbb{R}^M$ is the signal of interest, $\boldsymbol{\mathcal{D}}$ is the dictionary, and $\boldsymbol{x} \in \mathbb{R}^K$ is the sparse code vector. The primary goal in sparse coding is to learn the sparse vector $\boldsymbol{x}$ that provides the sparsest representation of input signals $\boldsymbol{y}$. The optimization problem for finding the sparse code $\boldsymbol{x}$ is generally NP-hard and can be formulated as:

$$\min_{\boldsymbol{x}} \frac{1}{2}||\boldsymbol{y} - \boldsymbol{\mathcal{D}}\boldsymbol{x}||_2^2 + \lambda_0||\boldsymbol{x}||_0 \tag{2.8}$$

where $\lambda$ is the **regularization parameter** that controls the sparsity of $\boldsymbol{x}$. The $\ell_0$ norm is a pseudo-norm that is equal to the number of nonzero values of vector $\boldsymbol{x}$, minimizing it directly promotes sparsity in the solution. Note that there are other formulations that will be detailed in Section 6.3.1.

Several algorithms based on the $\ell_0$ norm have been developed to solve this optimization problem. One notable example is the Orthogonal Matching Pursuit (OMP) [66] algorithm. OMP iteratively selects the dictionary atoms that are most correlated with the residual part of the signal and updates the sparse code vector. Another important algorithm is Iterative Hard Thresholding (IHT) [67], which iteratively updates the sparse code vector by applying a hard thresholding operator to the gradient step, promoting sparsity by setting small coefficients to zero.

**Although $\ell_0$ norm minimization directly enforces sparsity, it is computationally challenging due to its NP-hard nature**. Therefore, alternative techniques based on $\ell_1$ norm minimization have been proposed. The $\ell_1$ norm, defined as the sum of the absolute values of the vector elements, serves as a convex relaxation of the $\ell_0$ norm. This relaxation transforms the optimization problem into a convex one, which is more tractable [68]. Donoho proven that, for nonnoisy signal, minimizing $\ell_1$ norm will give the exact same solution that minimizing $\ell_0$ [69], and it gives a good approximation with noisy signal [70].

$$\min_{\boldsymbol{x}} \frac{1}{2}||\boldsymbol{y} - \boldsymbol{\mathcal{D}}\boldsymbol{x}||_2^2 + \lambda_1||\boldsymbol{x}||_1 \tag{2.9}$$

Basis Pursuit (BP) [71] algorithm is a popular method that uses $\ell_1$ norm minimization to find the sparse code vector. BP solves the optimization problem using linear programming techniques, ensuring an optimal solution under certain conditions. Another widely used algorithm is the Least Absolute Shrinkage and Selection Operator (LASSO) [72], which employs $\ell_1$ norm regularization in regression problems to achieve sparsity.

**The advantage of $\ell_1$ minimization techniques over $\ell_0$ minimization lies in their computational efficiency and the availability of well-established optimization methods.** Additionally, $\ell_1$ minimization often provides solutions that are sufficiently sparse for practical applications, making it a widely adopted approach in sparse coding.

Sparse coding is intrinsically linked with Compressed Sensing (CS), which similarly leverages sparsity to efficiently represent signals. Understanding this relationship enriches the broader context of sparse representations in signal processing.

## 2.6.2 Compressed sensing

### 2.6.2.1 Presentation

Compressed Sensing (CS) is a revolutionary technique that exploits the sparsity of a signal to recover it from a small number of linear measurements. The theory of CS, pioneered by Emmanuel Candès, Terence Tao [73]–[76] and David Donoho [77], **proves that certain signals can be recovered from far fewer samples than traditional methods use**. The core framework of compressed sensing is described by:

$$\boldsymbol{y_s} = \boldsymbol{\Phi x} \tag{2.10}$$

where $\boldsymbol{Y_s} \in \mathbb{R}^L$ represents the measurement vector, $\boldsymbol{\Phi}$ denotes the measurement matrix (a rectangular matrix with more columns than rows), and $\boldsymbol{x}$ is the sparse signal representation. According to compressed sensing theory, if $\boldsymbol{\Phi}$ satisfies certain fundamental properties such as the *Restricted Isometry Property (RIP)* [75], [78] and minimizes *mutual coherence* [68], [76], then by transmitting $\boldsymbol{Y_s}$, it is possible to reconstruct $\boldsymbol{x}$ with a certain probability.

In these conditions, the signal $\boldsymbol{x}$ can be recovered by solving the optimization problem:

$$\min_{\boldsymbol{x}} ||\boldsymbol{x}||_0 \text{ subject to } \boldsymbol{y_s} = \boldsymbol{\Phi x} \tag{2.11}$$

This problem is often relaxed to the convex problem:

$$\min_{\boldsymbol{x}} ||\boldsymbol{x}||_1 \text{ subject to } \boldsymbol{y_s} = \boldsymbol{\Phi x} \tag{2.12}$$

Julien Mairal, among others, has contributed to the field of compressed sensing by investigating various aspects of signal reconstruction and establishing theoretical limits for signal recovery [79], [80]. Mairal's work has also focused on the application of compressed sensing in image processing, resulting in the development of effective image-reconstruction algorithms that outperform traditional methods.

The link between sparse coding and compressed sensing is profound. Sparse coding provides a way to represent signals sparsely, which is a prerequisite for compressed sensing. Compressed sensing, in turn, leverages this sparsity to achieve efficient signal recovery from fewer measurements. This synergy has led to significant advancements in fields such as medical imaging, radar systems, and communications.

> This method is already used in image processing, but has also been utilized in ultra-wideband systems for channel estimation [81] and GNSS data compression [82]. However, it has yet to be widely applied to compress actual samples of data directly.

The field of compressed sensing continues to evolve, with numerous resources available to deepen understanding and explore new applications [83], [84].

### 2.6.2.2 Measurement requirements in compressed sensing

The efficacy of Compressed Sensing is dependent on the number of measurements $L$ [85], [86], which are typically much smaller than the number of samples dictated by the Nyquist rate for sparse

or compressible signals. A general relation [74] gives the minimum number of measurements needed to recover a sparse signal of length $K$ with $k$ nonzero coefficients with *high probability* is given by:

$$L \geq C \cdot k \cdot \log\left(\frac{K}{k}\right) \tag{2.13}$$

where $C$ is a constant that depends on the properties of the signal and the measurement process. Different recovery algorithms may require different constants $C$, reflecting a trade-off between computational complexity and recovery guarantee.

Figure 2.7 illustrates the progression of the estimator 2.13. It is observable that when $k > 100$, the estimation becomes invalid as the graph begins to decline. This highlights an essential aspect of compressed sensing, which will be crucial in our analysis: **Compressed Sensing** is typically used when $K \ll k$, with $k > 0$.

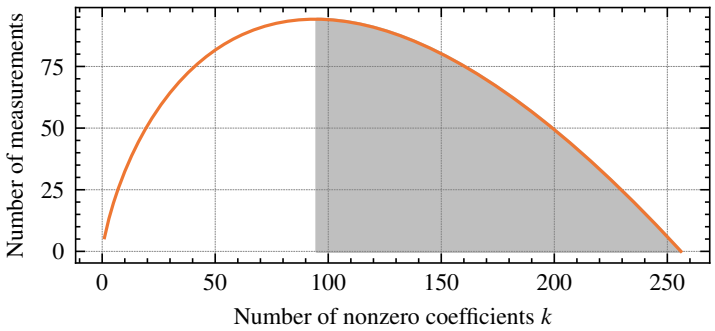



**Figure 2.7:** Evolution of the estimator 2.13 depending on the number of nonzeros coefficients. $K = 256$.

For example, greedy algorithms like Orthogonal Matching Pursuit (OMP) [66] and iterative methods such as Iterative Shrinkage-Thresholding Algorithm (ISTA) [87] offer practical solutions with different limits on measurement requirements. The performance of these algorithms is also influenced by the Restricted Isometry Property (RIP) [75] of the measurement matrix [68], which ensures that all subsets of $k$ columns are nearly orthogonal.

## 2.7 Neural-network based compression for digital communication signals

Data compression is an essential process in digital communication to minimize the storage space and bandwidth required for signal transmission. Recent advances in Neural Network (NN) models have revolutionized the field of data compression, offering high-efficiency adaptive algorithms. NNs are capable of learning from data to extract patterns and compactly represent information, which is particularly beneficial for compressing complex signals in digital communications.

Neural Network provide a flexible framework for designing compression algorithms that can be optimized for various types of data and transmission conditions. This adaptability allows for more efficient use of resources and can lead to significant improvements in performance compared to traditional methods. Techniques such as autoencoders, Recurrent Neural Networks (RNNs), and Generative Adversarial Networks (GANs) have shown promise in achieving high compression ratios while preserving the quality of reconstructed signals.

The integration of NNs in data compression for digital communication signals is still an evolving field. The ability of NNs to generalize from training data to unseen data makes them an attractive solution for dynamic and diverse communication environments. However, the exact form and implementation of NN-based compression in the context of our specific research remain to be determined. The potential of NNs is evident, and their applications are vast and varied.

## 2.8 Upcoming challenges

This chapter has underscored the importance of advanced data compression techniques in meeting the escalating demands of data transmission. The exploration of traditional methods such as **entropy coding** and **dictionary-based algorithms**, alongside **predictive coding** and emerging approaches such as **sparse coding** and **neural network-based compression**, highlights the diverse strategies available to improve data compression.

The integration of these techniques, particularly the innovative applications of **predictive coding and neural networks**, presents a promising path forward. As the field evolves, the focus must remain on developing adaptive, high-performance algorithms capable of addressing the unique challenges of digital communication systems.

Part I

# Compression based on signal processing & analysis

---

This part explores signal processing and analysis techniques to optimize data compression. In the following chapters, we will **address both uniform and nonuniform quantization techniques**, examining their roles in enhancing data compression and their effects on system performance. Additionally, by focusing on **sampling methods and their application** to digital communications, the following chapters will provide an in-depth analysis of resampling techniques and their practical implications, as well as a discussion on the impact of these methods on the performance of communication systems.

Chapter 3

# QUANTIZATION

***Or the lost art of shifting bits...***


3.1 Essential insights and familiar concepts . . . . . **64**
3.2 Uniform quantization . . . . . **65**
3.2.1 Basic principles and implementation . . . . . 65
3.2.2 Bit depth and dynamic range: the AGC dilemma . . . . . 65
3.2.3 Effectiveness with different signal types . . . . . 67
3.2.4 Comparative summary . . . . . 68
3.3 Non-uniform & vector quantization . . . . . **69**
3.3.1 Codebooks . . . . . 69
3.3.2 Algorithmic codebook construction . . . . . 70
3.3.3 Practical implementation . . . . . 73
3.4 Conclusion . . . . . **76**


## 3.1 Essential insights and familiar concepts

Quantization is a fundamental process in digital signal processing that serves as a bridge between the analog and digital realms. It involves the approximation of a continuous range of values by a finite set of discrete symbols or levels. This chapter delves into the nuances of quantization, exploring both uniform and nonuniform techniques. Uniform quantization, which allocates the same quantization step size across the entire signal range, is widely used because of its simplicity and ease of implementation. However, its limitations become apparent when dealing with signals that have a wide dynamic range or require a higher level of precision in certain areas of the signal. Nonuniform quantization, on the other hand, adapts the step sizes to the signal characteristics, offering a more efficient representation for such signals but at the cost of increased complexity. The following sections provide a detailed analysis of the impacts and trade-offs of these quantization methods in the context of telecommunications and signal compression.

## 3.2 Uniform quantization

### 3.2.1 Basic principles and implementation

#### 3.2.1.1 Quantization

Uniform quantization is characterized by the subdivision of a signal's range into equal-sized intervals, which are then mapped to distinct numerical values or codes. This form of quantization is implemented at the heart of Analog-to-Digital Converter (ADC), where **it translates the continuous amplitude of an analog signal into a finite number of digital levels**. The ADC's crucial role in initial quantization lies in its precision and resolution, which are defined by the number of bits used to represent each quantized value. With higher bit depths, an ADC can more accurately represent a signal, at the cost of increased complexity and resource use.

In the context of uniform quantization, each level is uniformly spaced, ensuring that every step of the analog signal is quantized with the same granularity. This makes the ADC design straightforward, as each quantization level is simply a fixed increment over the previous one. The consistency across levels means that the same error margin is applied throughout the signal's dynamic range, a factor that simplifies error estimation and signal reconstruction in the Digital-to-Analog Converter (DAC).

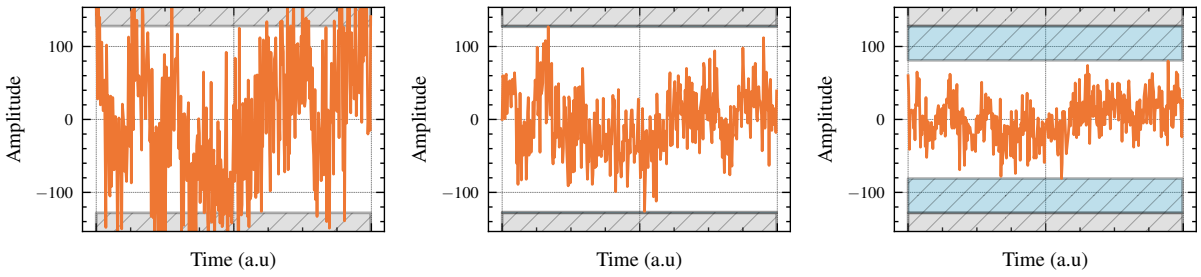
#### 3.2.1.2 Requantization

Requantization, or the process of further refining the quantized data, is inherently simple in the realm of uniform quantization. The principle operations involved are less about the complexity of computation and more about selecting the appropriate level of precision for the given context. Requantization can often be performed using efficient **bit shift operations** on a CPU, which adjust the bit depth of a signal to match the requirements of a specific application or to reduce the amount of data for transmission. In hardware platforms such as FPGAs or ASICs, requantization can be efficiently managed by the **selection of specific wires, which essentially translates into choosing which bits to keep or discard**. This hardware-centric approach to requantization is resource-efficient, as it does not necessitate complex arithmetic but rather a reconfiguration of connections, allowing for a streamlined adaptation to different resolution requirements.

### 3.2.2 Bit depth and dynamic range: the AGC dilemma

In digital communication systems, Automatic gain control (AGC) plays a crucial role in ensuring the optimal performance of the system. It achieves this by automatically modifying the gain of the input signal in order to avoid saturation, optimize the signal level -usually in decibels relative to full scale (dBFS)- and adapt to any variations in signal power. This functionality is crucial to avoid significant signal distortion that can impair data integrity and to ensure that the signal remains within an optimal dynamic range. By doing so, AGC not only improves the Signal-to-Noise Ratio (SNR), but also reduces bit errors, making it essential for applications that demand high reliability. Its adaptability to fluctuating signal levels, especially in variable communication environments like satellite communications, underscores its importance in maintaining consistent signal quality and minimizing transmission errors.

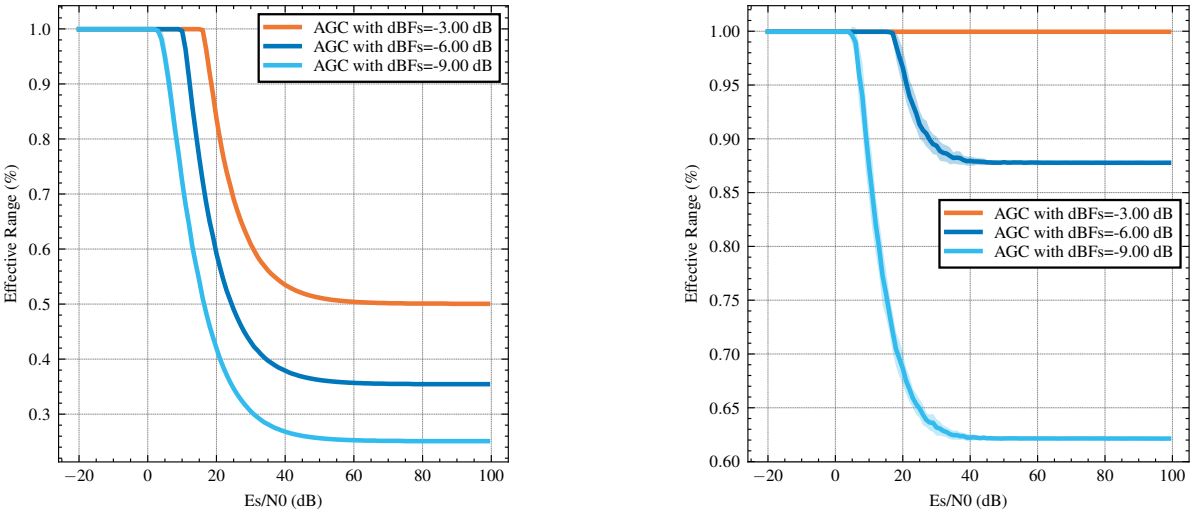
However, the AGC systems are not without challenges, particularly in quantized systems. Automatic gain adjustment can inadvertently **increase quantization noise**, especially with low-level input signals, thus reducing SNR. The dynamic nature of AGC can **introduce response delays**, potentially affecting the reception of fast signals or abrupt transitions. Rapid gain changes can also lead to **harmonic distortion**, especially when the system approaches saturation or undergoes sudden changes in signal level. Furthermore, while AGC aims to make efficient use of the available dynamic range, it can actually **limit the effective dynamic range** of the system in the presence of wide-ranging signal levels. This limitation can be critical in high-dynamic range applications, highlighting a trade-off between




**(a)** 0 dBFs **(b)** −6 dBFs **(c)** −10 dBFs

**Figure 3.1:** Quantization of a 8-PSK | $\text{RRC}_{0.25}$ signal with a signal-to-noise ratio of $\texttt{Es/N0} = 10$ dB using three different Automatic gain control (AGC) techniques on an 8-bit system. The first AGC approach causes signal saturation, leading to significant signal loss (EF $> 1$). In contrast, the second AGC method accurately matches the dynamic range of the signal (EF $= 1$). The final one does not use the complete range of available values, resulting in the waste of certain quantization levels that are never utilized (EF $< 1$).

maximizing signal integrity and managing the inherent limitations of AGC in quantized systems.




**(a)** 8-PSK | NRZ. $N_q = 12$ bits **(b)** 8-PSK | $\text{RRC}_{0.25}$. $N_q = 12$ bits

**Figure 3.2:** Evolution of Effective Range of a signal depending on AGC level and $\texttt{Es/N0}$. –Saturated at 1–.

Figure 3.2 presents the actual *Effective Range* of a signal depending on the $\texttt{Es/N0}$ level (in dB) and the AGC scale parameter. The effective range (EF) of a signal $y$ quantized on $N_q$ bits is defined as:

$$EF(y) = \frac{\max(y)}{2^{N_q - 1}} \tag{3.1}$$

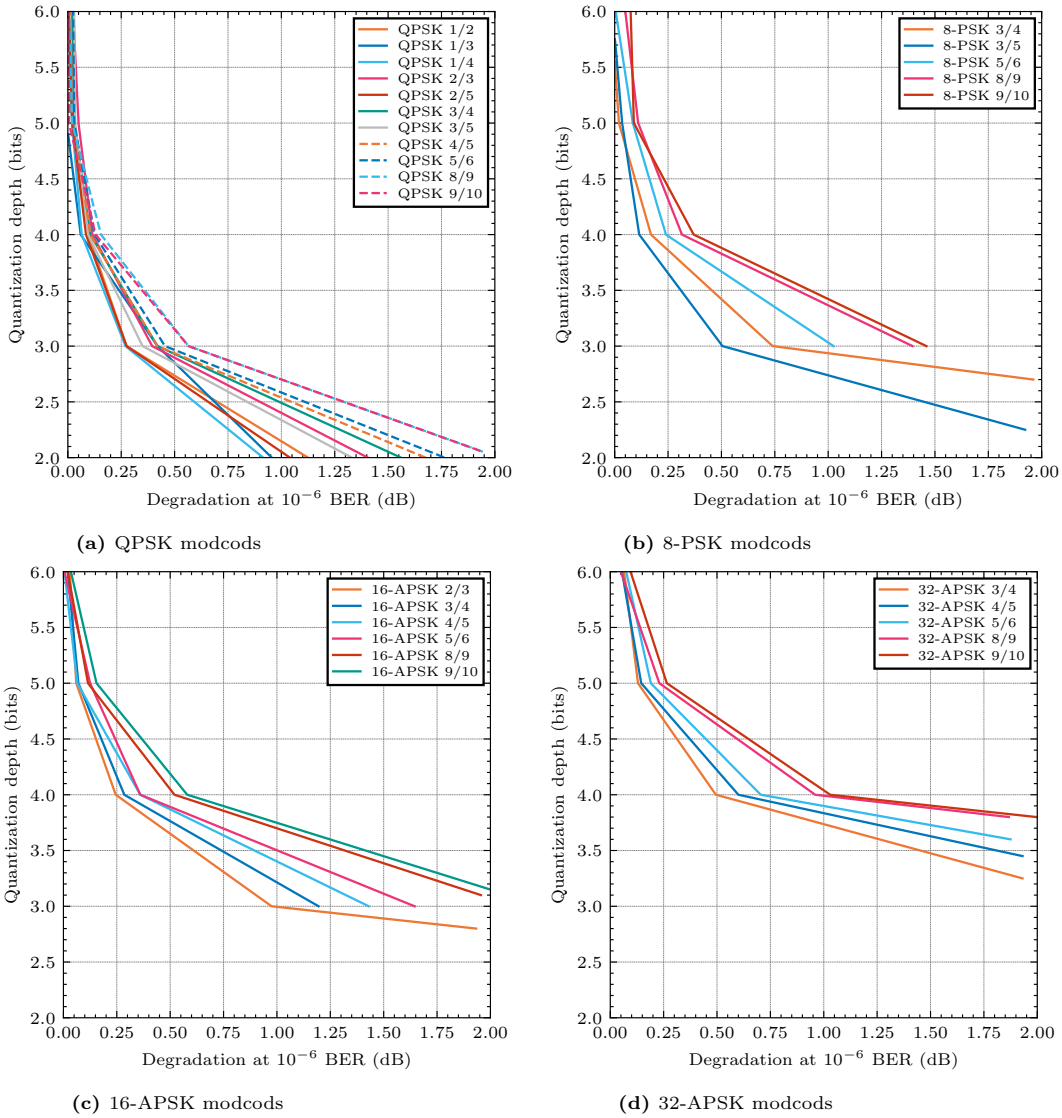
An EF $\geq 1$ means that the signal is saturated and data is actually lost. A value $< 1$ means that the dynamic range of the signal is not fully used and that the loss introduced by quantization is not minimal. **An AGC that is well-suited would always maintain the value slightly below 1.** In practical applications, AGC is typically implemented to prevent saturation when the SNR reaches the minimum level specified by the system's specifications.

**A common value when working with a classical telecommunication standard (like DVB-S2 or SCCC) is an AGC -6 dBFs**. As presented in Figure 3.2, this configuration ensures an effective range of at least 90% even at high SNR. Since the DVB-S2 modulation standard works mainly

with PSK-modulated signals shaped with RRC filters, the value fits perfectly.

### 3.2.3 Effectiveness with different signal types

Currently, there is a lack of comprehensive and trustworthy research on the degradation caused by quantization in the presence of a Forward Error Correction (FEC) system. Although *Safran Space Lab* has conducted some studies on OQPSK waveforms [14], a thorough analysis is still required for more conventional waveforms.




**(a)** QPSK modcods

**(b)** 8-PSK modcods

**(c)** 16-APSK modcods

**(d)** 32-APSK modcods

**Figure 3.3:** Quantization depth and its effect on degradation for **DVB-S2 modcods**. This evaluation is carried out on a pure Gaussian channel without any synchronization processes, employing a ($\text{RRC}_{0.25}$) filter.

The system simulated to obtain these degradation curves consists of a DVB-S2 transmission and reception chain, utilizing a soft-bit decoding method with an approximation of the Log-Likelihood Ratio (LLR). The Low-Density Parity-Check codes (LDPC) decoding is performed using the Belief Propagation (BP) algorithm, set to 50 iterations. For this study, long frames are used, with a frame length of 64,800 bits, as defined by the DVB-S2 standard.

In Figure 3.3, we present an examination of the deterioration and compression capabilities of all DVB-S2 modulation and coding schemes (modcods) when implemented in a system with a -6 dBFs AGC. A modcod in DVB-S2 refers to the combination of a specific modulation scheme (such as QPSK, 8PSK, 16APSK, or 32APSK) and a coding rate, which determines the error correction capability. The analysis reveals that, in numerous modcods, it is possible to achieve a quantization of 4 bits with minimal degradation (less than 1 dB). This achievement can be mainly attributed to the effective error correction provided by LDPC. Note that in practical applications, additional subsystems, such as synchronization, must also be considered and will affect the results of these simulations.

An examination of the simulation results reveals a nuanced interplay between the quantization depth and the performance degradation across different modulation and coding schemes (modcods). The simulation was structured to encompass a suite of modcods: Quadrature Phase Shift Keying (QPSK), 8-Phase Shift Keying (8-PSK), 16-Amplitude Phase Shift Keying (16-APSK) and 32-Amplitude Phase Shift Keying (32-APSK), with varying code rates.

We observe that, as the quantization depth increases, there is a notable improvement in the system's robustness to degradation. Specifically, higher bits of quantization (closer to 6 bits) result in lower degradation, which manifests itself as a flattening of the curves at lower degradation levels. This indicates that higher quantization depths are more effective in mitigating the noise introduced by the quantization process itself. We observe that lower order modulations, such as QPSK, are less susceptible to quantization degradation, even at lower quantization depths, compared to higher order modulations, such as 32-APSK. This is evidenced by the steeper slopes observed in the graphs for higher-order modcods, indicating a more pronounced sensitivity to quantization depth.

### 3.2.4 Comparative summary

In summary, the simulation data collectively suggest that a careful balance between quantization depth and modulation order, along with code rate selection, is paramount for optimizing system performance. The ability to maintain lower degradation at higher code rates without significantly increasing the quantization depth could be key in designing efficient and robust digital communication systems, particularly when considering the overhead and power consumption associated with higher quantization depths.

## 3.3 Non-uniform & vector quantization

### 3.3.1 Codebooks

A codebook denotes a predetermined collection of symbols or codes used to represent different quantized values of the input signal. In contrast to uniform quantization, where the intervals between quantization levels remain constant, nonuniform quantization adapts these intervals according to the probability distribution of the input signal, leading to enhanced data compression efficiency. The codebook essentially links the input signal ranges to distinct symbols, facilitating effective signal encoding and reconstruction while reducing information loss.

The effectiveness of the nonuniform quantization process is highly dependent on the structure and formulation of the codebook. **Typically, the codebook is designed to enhance signal representation by considering the statistical characteristics of the signal**, such as its variance and probability density function. This enhancement process frequently employs advanced algorithms like the Lloyd-Max algorithm or k-means clustering, which are aimed at reducing the quantization error by strategically positioning the codebook entries based on the signal distribution. Consequently, a series of quantization levels is generated, with a denser concentration in regions where the signal is more probable, thus leading to improved accuracy in those specific areas.

Furthermore, the efficiency of data compression achieved through nonuniform quantization and its associated codebook is directly linked to the entropy of the signal. By aligning the codebook entries with the most probable input values, the quantizer can exploit the redundancy inherent in real-world signals. This alignment allows for the use of shorter codes for more probable values and longer codes for less probable ones, an approach known as entropy coding. Techniques such as Huffman coding or arithmetic coding are often employed in conjunction with nonuniform quantization to further compress the data without significant loss of information.

In practical applications, **the creation and optimization of the codebook are not trivial tasks**. They require careful consideration of the specific characteristics of the signal to be quantized and the desired balance between compression efficiency and computational complexity. The dynamic nature of real-world signals also poses a challenge, necessitating adaptive codebook strategies that can respond to changes in the signal's statistical properties over time. This adaptability ensures that the quantization process remains efficient in a wide range of signal conditions, thus enhancing the overall performance of digital communication systems.

### 3.3.2 Algorithmic codebook construction

#### 3.3.2.1 The LLoyd-max algorithm

The Lloyd-Max algorithm [24], [88], [89], often simply referred to as the Lloyd algorithm, is a widely used iterative method in signal processing for quantization, which is the process of mapping input values from a large set (often a continuous set) to output values in a (countable) smaller set. The aim of this algorithm is to minimize the quantization error, making it an optimal scalar quantizer for a given probability density function (PDF) of the input data. The algorithm works as follows:

**General Lloyd algorithm**

**Algorithm 11** Lloyd-Max Quantization Algorithm

```
Initialize quantization levels
repeat
    Expectation Step:
    for each data point in input data do
        Assign to the nearest quantization level
    end for
    Maximization Step:
    for each quantization level do
        Compute centroid of assigned data points.
        Update quantization level to this centroid.
    end for
until quantization levels converge or max iterations reached
```

Algorithm 11 presents the implementation of the Lloyd-Max algorithm. This algorithm is critically dependent on the availability of the Probability Density Function (PDF) and the Cumulative Distribution Function (CDF) of the data. When these distributions are analytically known, the algorithm can directly compute optimal points for quantization. However, in practical scenarios, especially with real-world data, these distributions might not be known a priori and need to be estimated. Histogram analysis provides a method for this estimation, enabling the application of the Lloyd-Max algorithm even when direct analytical methods are not feasible.

**Case 1: Usage with known analytical expressions for PDF and CDF** In situations where the PDF and CDF of the data are known, the Lloyd-Max algorithm can be implemented with precise calculations for the decision and reconstruction levels. This scenario allows for the direct application of mathematical formulas to determine optimal quantization points, which typically results in a more efficient convergence and reduced computational cost.

**Algorithm 12** Lloyd-Max Quantization with Known PDF and CDF

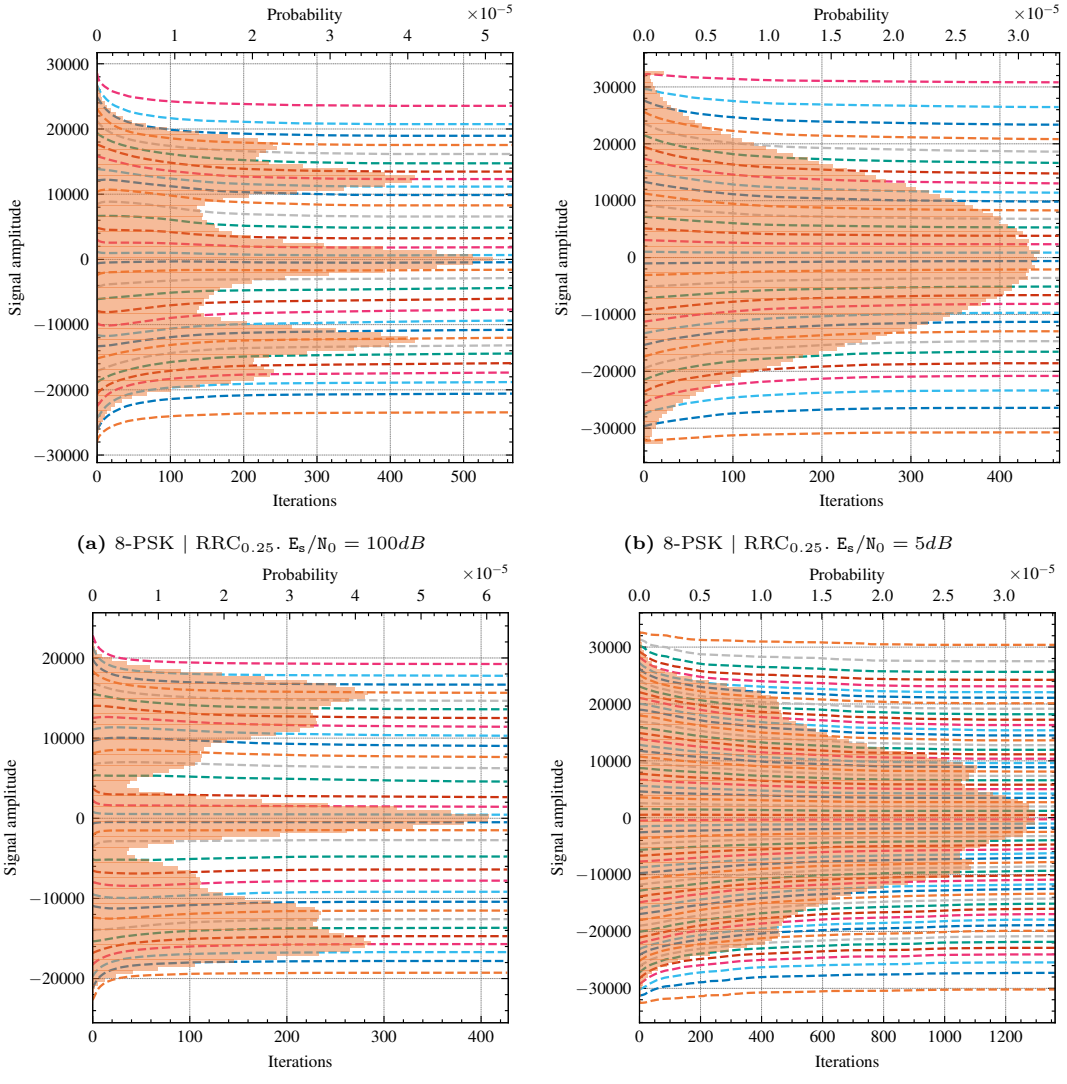
**Require:** Initial set of reconstruction levels $r_k$, number of levels $N$, tolerance $\epsilon$
**Ensure:** Optimized reconstruction levels $r_k$ and decision levels $t_k$
Initialize decision levels $t_k$ using any suitable method
Initialize error $D \leftarrow \infty$
**repeat**
  $D_{\text{old}} \leftarrow D$
  **for** $k = 1$ to $N$ **do**
    Calculate $r_k \leftarrow \dfrac{\int_{t_k}^{t_{k+1}} x p_X(x) dx}{\int_{t_k}^{t_{k+1}} p_X(x) dx}$
  **end for**
  **for** $k = 1$ to $N - 1$ **do**
    Calculate $t_{k+1} \leftarrow \dfrac{r_k + r_{k+1}}{2}$
  **end for**
  Compute total distortion $D \leftarrow E\left[|x - \hat{x}|^2\right]$
**until** $|D - D_{\text{old}}| < \epsilon$

**Case 2: Usage with unknown expressions, using histogram analysis** When PDF and CDF are not analytically available, the algorithm must rely on empirical data analysis to estimate these functions. Histogram analysis is used to approximate the PDF from sampled data, which is then used to compute the quantization levels. This method is more data-dependent and can vary in effectiveness based on the quality and quantity of the data samples.

**Algorithm 13** Lloyd-Max Quantization with Histogram Analysis

**Require:** Sample data set $X$, number of bins $M$, number of levels $N$, tolerance $\epsilon$
**Ensure:** Optimized reconstruction levels $r_k$ and decision levels $t_k$
Construct histogram of $X$ with $M$ bins to approximate $p_X(x)$
Estimate initial decision levels $t_k$ from histogram
Initialize error $D \leftarrow \infty$
**repeat**
  $D_{\text{old}} \leftarrow D$
  **for** $k = 1$ to $N$ **do**
    Calculate $r_k \leftarrow \dfrac{\sum_{x \in [t_k, t_{k+1}]} x \cdot \text{count}(x)}{\sum_{x \in [t_k, t_{k+1}]} \text{count}(x)}$
  **end for**
  **for** $k = 1$ to $N$ **do**
    Calculate $t_{k+1} \leftarrow \dfrac{r_k + r_{k+1}}{2}$
  **end for**
  Compute total distortion $D \leftarrow \sum_{k=1}^{N} \sum_{x \in [t_k, t_{k+1}]} |x - r_k|^2 \cdot \text{count}(x)$
**until** $|D - D_{\text{old}}| < \epsilon$

The Lloyd-Max algorithm iteratively adjusts the quantization levels to match this distribution, thus minimizing the Mean Squared Error (MSE) between the original and quantized data. **This process is analogous to finding the best representative points (quantization levels) so that the average distortion (error due to quantization) is minimized**. Furthermore, since the signal distribution is perfectly known, the actual range of values exploited by the signal is maximized.




**(a)** 8-PSK | $\mathrm{RRC}_{0.25}$. $\mathtt{E_s/N_0} = 100dB$

**(b)** 8-PSK | $\mathrm{RRC}_{0.25}$. $\mathtt{E_s/N_0} = 5dB$

**(c)** GMSK | $BT = 0.5$. $\mathtt{E_s/N_0} = 20dB$

**(d)** 32-APSK | $\mathrm{RRC}_{0.25}$. $\mathtt{E_s/N_0} = 20dB$

**Figure 3.4:** Convergence of Lloyd-max algorithm according to waveform distribution after shaping.

The concept of one-dimensional quantization through the Lloyd-Max algorithm is actually the basis for several straightforward compression techniques. This approach is implemented in the Open Radio Equipment Interface (ORI) [90, Section 7.2.4, "*Nonlinear quantization process*"].

Figure 3.4 presents four examples of Lloyd-max algorithm processing for different waveforms, the dotted lines represent the evolution of each bounds of the quantization levels. The convergence is reached when the bounds do not move significantly between two iterations ($\varepsilon = 10^{-6}$).

#### 3.3.2.2 Linde-Buzo-Gray algorithm

The Linde-Buzo-Gray (LBG) algorithm, often referred to as the *generalized Lloyd Algorithm*, is an optimization of the *K-Means* clustering algorithm [91], explicitly designed to **produce a codebook whose size is a power of two**. This feature is particularly beneficial in the context of vector quantization, where the objective is to minimize distortion in representing a large set of data with a smaller set of representative vectors.

**Algorithm 14** Linde-Buzo-Gray Algorithm

**Require:** Data vectors $X \in \mathbb{R}^{M \times L}$, desired codebook size $N = 2^k$, convergence threshold $\epsilon$
**Ensure:** Optimized codebook of size $N$
Initialize codebook $C$ with one code-vector, the mean of all data vectors
Set initial average distortion $D$ to a high value
**while** size of $C < N$ **do**
    Split each code-vector in $C$ into two by adding and subtracting a small vector $\epsilon$
    **repeat**
        Assign each data vector to the nearest code-vector in $C$
        Update each code-vector in $C$ to be the centroid of all data vectors assigned to it
        Calculate new average distortion $D_{\text{new}}$
    **until** change in distortion $|D - D_{\text{new}}| < \epsilon$
    Update $D \leftarrow D_{\text{new}}$
**end while**
**return** $C$

A 2D Linde-Buzo-Gray (LBG) codebook can be effectively visualized using Voronoi diagrams, which are a valuable tool for understanding the spatial relationships within data. Voronoi diagrams partition a plane into regions based on distances to a specified set of points, known as seeds. **Each seed corresponds to a code vector in the LBG codebook, and the region around each seed (Voronoi cell) comprises all points closer to that seed than to any other**. When discussing LBG for 2D codebooks, each seed represents a different code-vector, and the Voronoi diagram illustrates how each vector partitions the data space, effectively showing the area of influence each vector has over the data set. This graphical representation is particularly useful for analyzing the density and distribution of data points and understanding how they relate to the chosen centroids in vector quantization, making it easier to evaluate the efficiency and coverage of the quantization process. Figure 3.5 displays various codebook representations for distinct noise-free waveforms commonly employed in space communications.

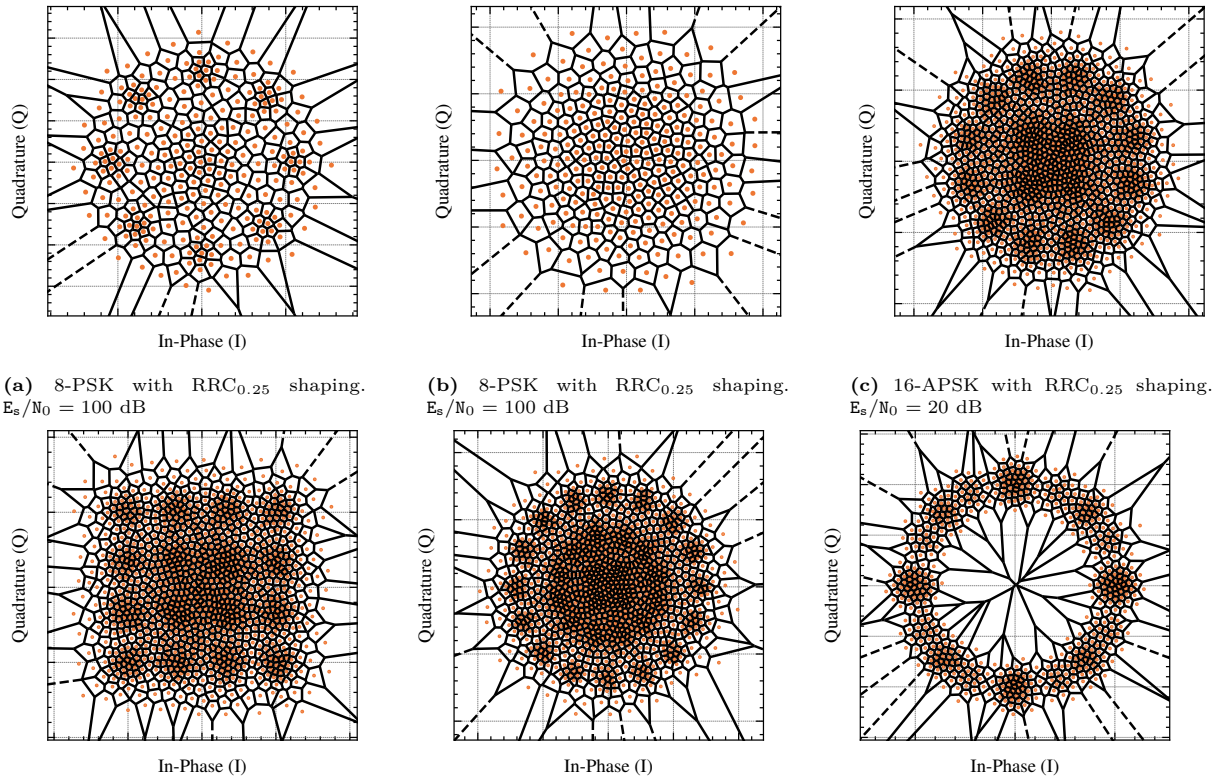
The LBG algorithm offers several advantages, particularly in applications requiring efficient and effective quantization:

- **Optimized for Power-of-Two Codebook Sizes:** By explicitly designing the algorithm to generate codebooks of sizes that are powers of two, the LBG algorithm is well-suited for digital systems where such dimensions are optimal due to hardware and algorithmic efficiencies.
- **Flexibility in Dimensions:** The algorithm can be applied to data in any number of dimensions, making it versatile

### 3.3.3 Practical implementation

#### 3.3.3.1 Non-uniform quantization of noisy signals

The main difficulty in nonuniform quantization lies in accurately identifying the suitable codebook for the signal under consideration. When a baseband signal undergoes quantization, **the choice of a codebook (essentially a set of quantization levels) greatly influences the efficiency and**




**(a)** 8-PSK with $\mathrm{RRC}_{0.25}$ shaping. $\mathtt{E_s/N_0} = 100$ dB

**(b)** 8-PSK with $\mathrm{RRC}_{0.25}$ shaping. $\mathtt{E_s/N_0} = 100$ dB

**(c)** 16-APSK with $\mathrm{RRC}_{0.25}$ shaping. $\mathtt{E_s/N_0} = 20$ dB

**(d)** 16-QAM with $\mathrm{RRC}_{0.22}$ shaping. $\mathtt{E_s/N_0} = 20$ dB

**(e)** 32-APSK with $\mathrm{RRC}_{0.25}$ shaping. $\mathtt{E_s/N_0} = 100$ dB

**(f)** GMSK with $BT = 0.5$. $\mathtt{E_s/N_0} = 20$ dB

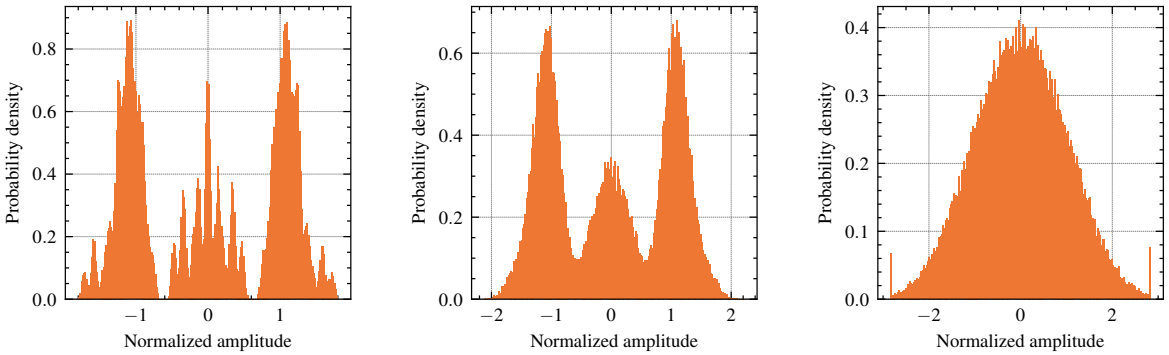
**Figure 3.5:** Distinct codebook based on the LBG algorithm. The resulting centroids are the centroids of a Voronoi diagram that align with the edges of the codebook.

**precision of this process**. The difficulty escalates when external noise is introduced into the system, as this can rapidly render the previously optimized codebook inadequate, leading to a decrease in the quantizer's performance.

When considering the addition of noise (assumed Gaussian and white) to a signal, one observes a tendency towards a Gaussian distribution in its histogram. This transformation is particularly noticeable as the noise begins to dominate the signal characteristics. The shift towards a Gaussian profile in noisy environments stems from the Central Limit Theorem, which suggests that the sum of many independent random variables, regardless of their specific distributions, will tend towards a Gaussian distribution.

Due to the tendency of signals to exhibit Gaussian properties under high noise conditions, **a strategic method in nonuniform quantization involves using a Gaussian codebook**. This approach offers several benefits:

- Removing the necessity to dynamically determine the codebook in real-time, which can be both resource-demanding and prone to errors. Instead, it is possible to precompute a collection of codebooks designed for Gaussian-distributed signals, with different sizes to match various signal fidelity and bandwidth needs.
- By using a codebook specifically optimized for Gaussian-distributed signals, optimal quantization can be achieved in situations where noise levels cause the signal to approximate this distribution. This optimization is based on minimizing quantization error, which is essential for preserving signal




**(a)** $\mathtt{E_s/N_0} = 100\text{dB}$ **(b)** $\mathtt{E_s/N_0} = 20\text{dB}$ **(c)** $\mathtt{E_s/N_0} = 0\text{dB}$

**Figure 3.6:** Evolution of a BPSK waveform histogram, shapped with $\text{RRC}_{0.25}$, depending on signal-to-noise ratio value.

integrity during compression and transmission.

These strategies underscore the importance of adapting quantization techniques to the characteristics of the signal, especially in environments where noise is a significant factor. Using the statistical properties of noise and signal distribution, one can enhance the efficiency and accuracy of nonuniform quantization methods.

#### 3.3.3.2 Residual entropy & effective compression

If the codebook is correctly structured, the statistical distribution of the signal post-quantization should be uniform. However, imperfections in the codebook and variable noise effects on signals often lead to a histogram that resembles a Gaussian distribution. In such scenarios, employing a post-processing entropy coder might help in reducing the bit usage for signal encoding. Yet, generally, if the codebook is well designed, using a post-entropy coder does not improve the overall compression ratio significantly.

**Lossy compression dilemna**

Balancing the use of a nonuniform quantizer between compression ratio and MSE gain is challenging. Direct comparisons with other methods are not straightforward. For example, it is uncertain whether a simple nonuniform quantization step would outperform a uniform quantization step followed by an entropy coder when degradation levels are the same. **This observation is relevant for all methods of lossy compression.**

## 3.4 Conclusion

In this discussion on quantization methods, we explored the capabilities and limitations of both uniform and nonuniform quantization techniques. Uniform quantization, characterized by its simplicity and straightforward implementation via bit-shifting, does not require a codebook and easily maps signals to quantized values. However, it exhibits rapid degradation in the Mean Squared Error (MSE) and Bit-error rate (BER) when the quantization depth $N_q$ is reduced below six bits. The simplicity of this method also limits its tunability, **as the only adjustable parameter is $N_q$, resulting in a limited range of achievable compression ratios**. **Nevertheless, due to its simplicity, this method can be complemented with other compression strategies –such as entropy coding– to improve the overall compression ratio [23].**

On the other hand, nonuniform quantization offers a more flexible approach by leveraging codebooks that are recalculated using algorithms such as Lloyd Max and k-means. Although this method allows for finer adjustment of the quantization parameters, it introduces its own set of challenges.

- **1D Non-uniform Quantization:** This approach is highly sensitive to phase error, which can significantly affect performance.
- **2D Non-uniform Quantization (Vector Quantization):** It offers a larger parameter space, with possible values in the range $[\![0, 2^{2N_q}]\!]$, compared to $[\![0, 2^{N_q}]\!]$ for 1D methods. This increased range allows for more potential values of compression ratio. However, in terms of degradation, **it does not provide substantial performance improvements over 1D quantization**, although it may offer better resistance to phase error. It also requires a large amount of resources to be calculated and stored, hundreds of times the amount needed for 1D quantization.

Although all nonuniform methods reduce MSE, linking this reduction directly to effective compression is challenging. After MSE is minimized, the resulting histogram tends to be relatively flat or uniform, making subsequent data compression through entropy coders ineffective. **Consequently, the degree of compression depends solely on the quantization process itself.**

**Conclusion on quantization methods**

Given our goal of developing adaptable and tunable compression methods with low resource consumption for implementation on targets such as CPUs or FPGAs, nonuniform quantization methods fall short of our expectations in terms of efficient lossy compression. **This analysis suggests the need for further exploration of alternative compression approaches that better align with our requirements in digital communication systems.**

Chapter 4

# SAMPLING METHODS

***To have one's cake and eat it too***


4.1 From one sample to another . . . . . **77**
4.1.1 Up-sampling & downsampling . . . . . 77
4.1.2 Baseband signal sampling . . . . . 78
4.1.3 Application to digital communications . . . . . 80
4.2 Discussion . . . . . **84**
4.2.1 About uniform sampling . . . . . 84
4.2.2 About non-uniform sampling . . . . . 84
4.3 Conclusion & actual usage . . . . . **85**


## 4.1 From one sample to another

Resampling, a well-known and widely used concept in digital signal processing, refers to the modification of the sampling rate of a digital signal, that is, the number of samples per unit of time. In the case of digital communications, we more often refer to it as the number of samples per symbol. In this manuscript, we will denote it as $\mathcal{F}$.

### 4.1.1 Up-sampling & downsampling

The theory behind resampling is grounded in the Nyquist-Shannon sampling theorem [1], which stipulates that a continuous signal can be completely represented in its **sampled form and perfectly reconstructed if it is sampled at a rate that is at least twice its highest frequency component** –this is called the Nyquist rate–. Resampling can occur in two forms:

1 · **Downsampling**, where the sample rate is reduced, thus decreasing the number of samples and potentially saving on storage space and processing power; and upsampling, which increases the sample rate, often used to match the sampling rates of different systems or for higher precision in digital processing. Downsampling must be preceded by a low-pass filtering step to prevent aliasing, a phenomenon in which high-frequency components of the signal are improperly mapped to lower frequencies, leading to distortion.

2 · **Upsampling**, on the other hand, is succeeded by interpolation to complete the extra sample points. These procedures allow digital signals to be adjusted in a versatile manner to meet the diverse needs and limitations of signal processing systems.

Figure 4.1 illustrates the effect of upsampling and downsampling on a signal on a small example using a FFT resampling technique.

It is important to highlight that the notions of **sampling frequency $\mathcal{F}_s$ and symbol rate $\mathbf{R}_\mathcal{S}$ can be bypassed by focusing on the number of samples per symbol $\mathcal{F}$**. Moreover, we have the relation $\mathcal{F} = \mathbf{R}_\mathcal{S} \times \mathcal{F}_s$.

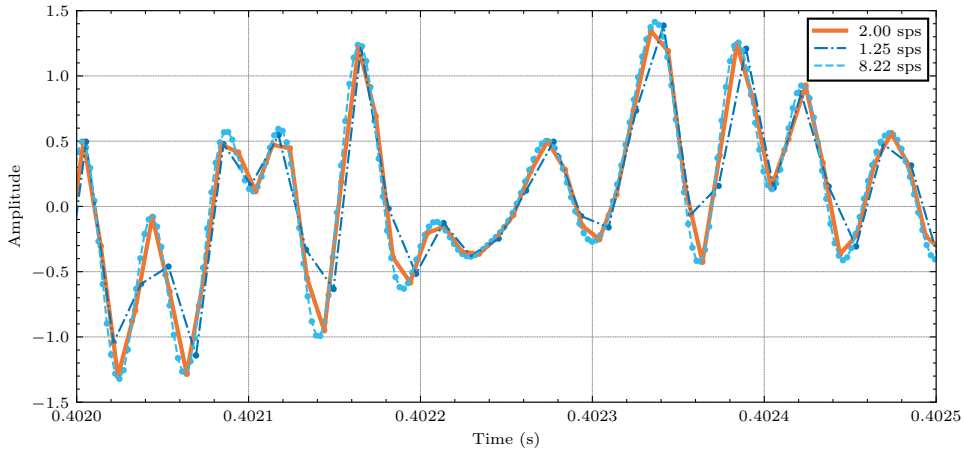



**Figure 4.1:** Effect of upsampling and downsampling on a signal (only real part is shown). The signal is a 16-APSK ($\text{RRC}_{0.25}$) modulated signal with an arbitrary initial sampling frequency. Resampling is performed using a FFT resampler.

### 4.1.2 Baseband signal sampling

In the examination of baseband signals, the common method is to operate with a whole number of samples for each symbol, with a minimum of two samples per symbol. This convention is mainly influenced by various important considerations.

1 · **Synchronization Requirements**: Common bit synchronization loops (such as the Gardner algorithm [92], [93]) typically require a minimum of two samples per symbol for optimal functioning.

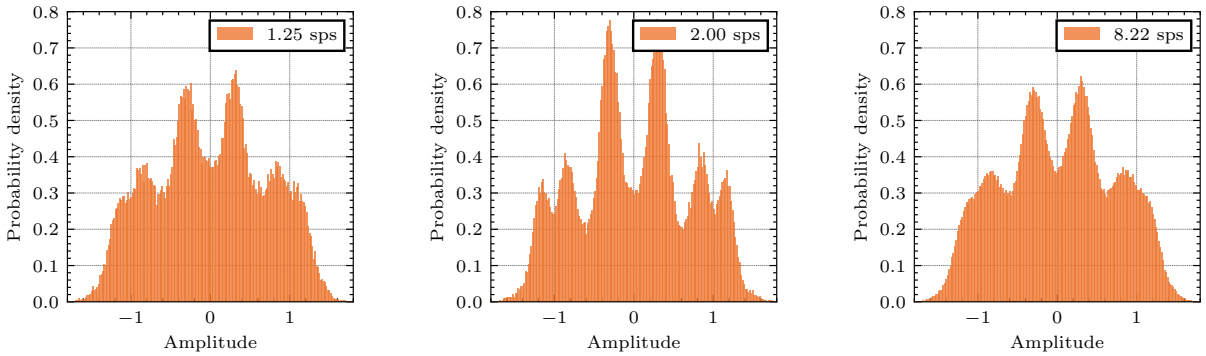
2 · **Signal Shaping Degradation**: Certain signal shaping techniques, such as Non return to zero (NRZ) and Return to zero (RZ), begin to rapidly degrade the signal quality below two samples per symbol. This degradation is attributed to their spectral spreading.

3 · **Anti-Aliasing Filter Constraints**: As the number of samples per symbol decreases, anti-aliasing filters can become too narrow, leading to practical implementation challenges. Scenarios often arise where either the resources for implementing an adequate filter are insufficient or the filter length becomes disproportionately large relative to the signal length. Both cases require compromises that ultimately result in signal distortion or degradation. However, it is critical to recognize that such degradation is a consequence of the filtering process, not of the resampling itself.

**Regarding Shannon's theorem**

Contrary to a common misconception, **Shannon's theorem does not require that the number of samples per symbol exceed two**. In fact, the need for a higher number of samples per symbol is primarily dependent on the type of shaping filter applied. For example, a signal shaped with a (Square)-Root-raised-cosine filter (RRC) filter, characterized by a roll-off factor denoted $\beta$, only exhibits degradation when the number of samples per symbol $\mathcal{F}$ is lower than $1 + \beta$.

However, by adding new intermediate samples, resampling actually changes the signal histogram, and thus the signal entropy (Figure 4.2).




**(a)** $\mathcal{F} = 1.25$ **(b)** $\mathcal{F} = 2.00$ **(c)** $\mathcal{F} = 8.22$

**Figure 4.2:** Evolution of signal histogram depending on the number of sample per symbol. Resampling has been performed using a FFT resampler.

#### 4.1.2.1 Methods & limitations

Details about the following methods and many other methods can be found in the beautiful book by Frederic J Harris [94]. The following section attempts to give a brief overview of the most common methods used in digital communications.

**Up/Down FIR & Poly-phase filters** The up and down sampling process, fundamental in digital signal processing, involves increasing (upsampling) and then decreasing (downsampling) the sampling rate of a signal. This approach, while straightforward, often involves some computational load –mainly multiplications– due to the need to apply a Finite Impulse Response filter (FIR) filter across the entire up-sampled signal before down-sampling it.




**Figure 4.3:** Up & Down sampling process using a FIR filter. In MATLAB®, this kind of filter can be applied using *upfirdn* function

In contrast, a polyphase filter refines this method by decomposing the FIR filter into several subfilters or phases. Each phase operates on different segments of the input signal, drastically reducing unnecessary computations. For example, with an upsampling factor of (M) and a downsampling factor of $(N)$, the traditional up-and-down method would compute $M \times N$ coefficients for every output. However, the polyphase approach only calculates the coefficients of $(M)$ or $(N)$, depending on the resampling requirement. This not only leads to a reduction in the computational burden, but also maintains the quality of the resampled signal, thereby offering an optimized solution for digital signal resampling.

Both of these methods offer a simple and efficient way to resample a signal. However, they are not suitable for variable resampling ratios and are not able to handle a noninteger resampling ratio. Despite these limitations, they are still widely used in digital communications, especially in hardware implementation.

**Farrow filter & Lagrange interpolation** The Farrow filter is a prominent solution for resampling, especially renowned for its ability to handle arbitrary resampling ratios with high precision. Unlike the

traditional polyphase filter, which is optimized for fixed integer resampling ratios, the Farrow filter employs a polynomial approximation method to allow continuous variation in the resampling factor [95], [96].

The core concept of the Farrow filter lies in its structure, which consists of several polynomial filters. Each filter computes a different coefficient of the polynomial that approximates the desired resampling function. The input signal is passed through these polynomial filters, and their outputs are linearly combined to produce the resampled signal. This method provides a significant advantage in scenarios requiring variable resampling rates as it negates the need for multiple fixed-ratio polyphase filter banks, thus simplifying the architecture and reducing computational complexity.

The performance of the Farrow filter is most evident in applications demanding high precision and flexibility, such as in digital audio processing and telecommunications, where varying sampling rates are common. Its ability to adjust the resampling ratio on the fly, without the need for a complete filter redesign, makes it a really interesting tool. However, its hardware implementation can be difficult.

**Fourrier / FFT resampler** Resampling via Fast Fourier Transform (FFT) offers a distinct approach in digital signal processing, particularly beneficial for its efficiency in handling large data sets. This method leverages FFT to convert the signal from the time domain to the frequency domain, allowing for the manipulation of the signal's spectral components.

The process begins by applying an FFT to the input signal, which decomposes the signal into its constituent frequencies. Once in the frequency domain, the signal can be efficiently modified to achieve the desired resampling rate. This modification typically involves either zero-padding or truncating the frequency spectrum, corresponding to upsampling or downsampling, respectively. Following the spectral adjustment, an inverse FFT is applied, transforming the signal back to the time domain with the new sampling rate.

The primary advantage of FFT-based resampling lies in its ability to handle large-scale data efficiently, making it particularly suited for applications where computational speed is crucial. Additionally, this method avoids the potential artifacts introduced by time-domain filters, offering a high-quality resampling output. However, it's important to note that FFT-based resampling is inherently limited by the spectral resolution of the input signal and is most effective when the desired resampling rate is a rational multiple of the original rate.

### 4.1.3 Application to digital communications

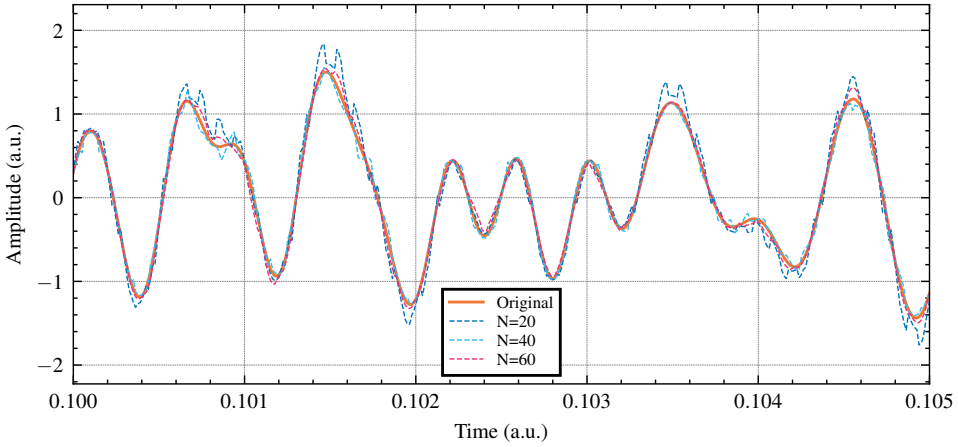
#### 4.1.3.1 Minimum sampling rate

Determining an optimal minimum sampling rate for digital communications signals without incurring significant degradation is a nuanced challenge. The complexity arises from the signal's inherent characteristics, such as whether Forward Error Correction (FEC) is applied or the specific type of signal shaping utilized. Despite these variables, we aim to offer a structured approach to guide this determination.

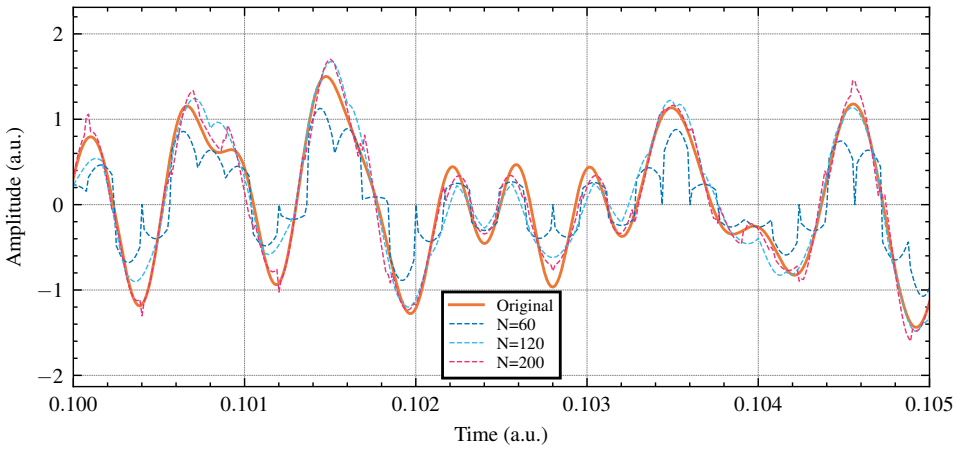
The **IRIG-106 standard [97] provides a foundational guideline for telemetry signals**, suggesting a sampling rate that captures 99% of the signal energy. This recommendation serves as a conservative benchmark to ensure signal integrity. However, it is crucial to recognize that the signal energy distribution across the frequency spectrum is not uniform. Consequently, **it is feasible to reconstruct the signal effectively at a lower sampling rate than what the 99% energy criterion might suggest.**

A critical factor in optimizing the sampling rate is the **consideration of the guard band**,

especially when employing a filter-based resampling approach. The guard band, the frequency buffer between signal bandwidth and sampling rate, plays a crucial role in preventing aliasing. The size of the guard band should be carefully selected based on the characteristics of the filter and the signal itself. By judiciously choosing the guard band, one can achieve a balance between efficient sampling and maintaining the fidelity of the signal, thus navigating the complexities of determining the minimum sampling rate for digital communications. The lower the guard band, the stepper the filter, and the more resources –multiplications, influenced by the number of coefficients $N$– are needed to implement it, as shown in Figure 4.4.




**(a)** $\beta = 2.0$




**(b)** $\beta = 1.25$

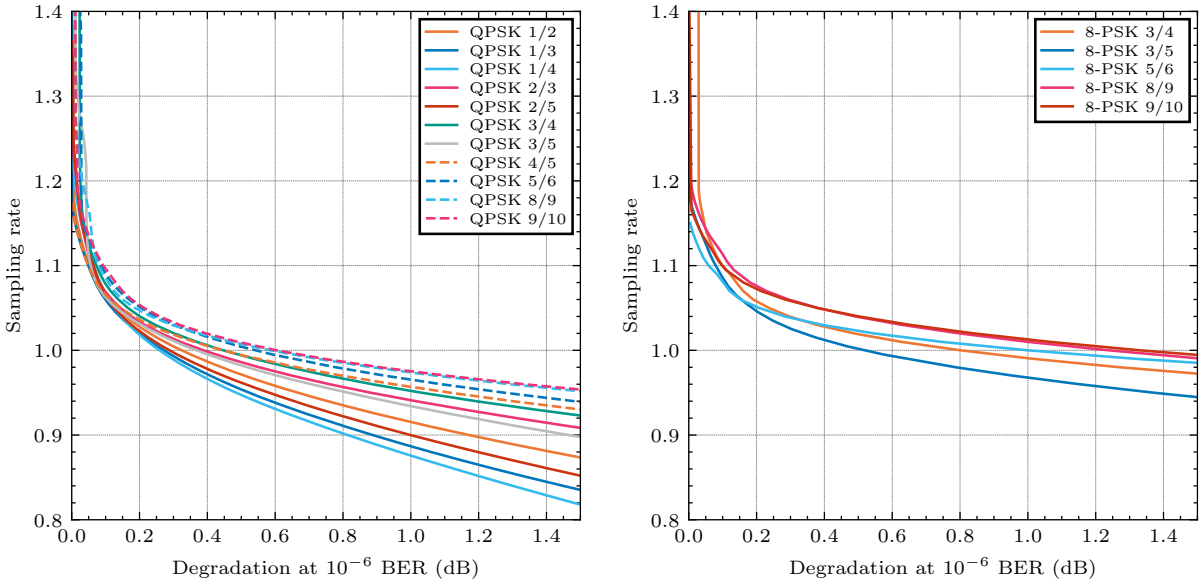
**Figure 4.4:** Operation of double resampling of a 8-PSK | $\text{RRC}_{0.25}$ signal from $\mathcal{F} = 16$ to $\mathcal{F} = \beta$, then back to $\mathcal{F} = 16$ using a polyphase filter with $N$ coefficients. The lower $\beta$, the more coefficients are needed. Note that the filter is designed using the *firls* function from MATLAB®.

#### 4.1.3.2 Joint coding

A practical and reliable method to determine the optimal minimum sampling rate involves the use of simulation techniques. Initially, the signal is generated and subsequently resampled at varying sampling rates. After resampling, the signal undergoes demodulation, after which the Bit-error rate (BER) is calculated. An iterative adjustment of the sampling rate is performed until the BER reaches acceptable limits. Although this methodology proves to be highly effective in determining the minimum sampling rate necessary for maintaining signal integrity, it is notably time-consuming.

Figure 4.5 illustrates the application of these methods in all DVB-S2 modcods –long frames–. It is evident that the **minimum resampling rate for a specific level of degradation is strongly influenced by both the modulation index and the FEC code rate and decoding parameters used**. The relationship between these factors mirrors the findings discussed in Section 3.2.3. To be more specific and in parallel to the conclusions of Section 3, the lower the FEC ratio is, the better the signal will be resistant to degradation introduced by resampling.

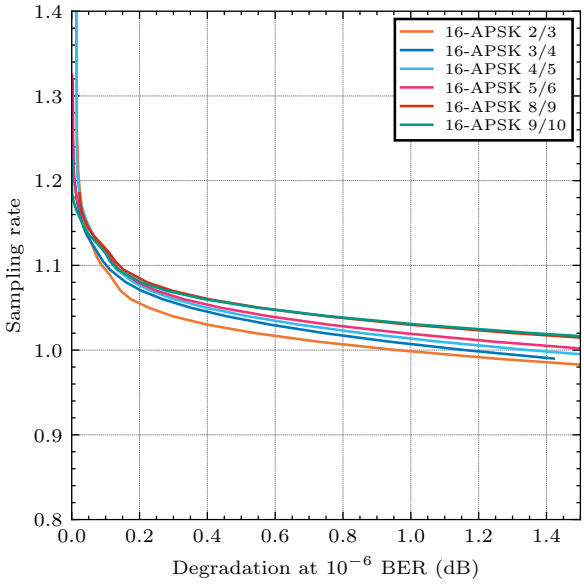
In the subsequent section that focuses on lossy compression, **we will leverage the utilization of FEC to improve our compression efficiency**. This particular compression technique will be termed **joint coding**.



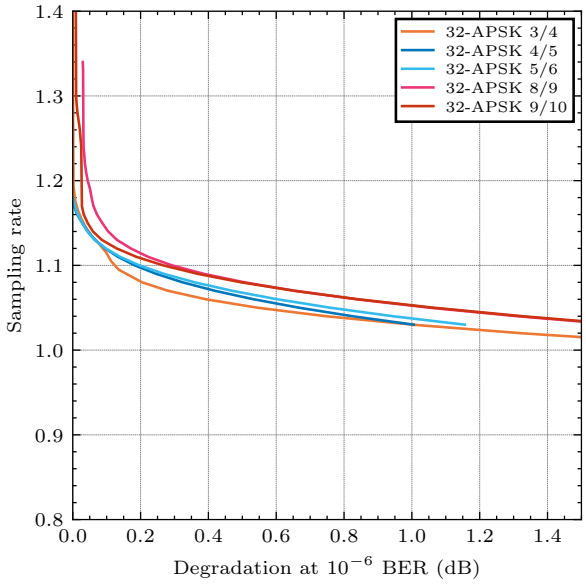

**(a)** QPSK modcods




**(b)** 8-PSK modcods




**(c)** 16-APSK modcods




**(d)** 32-APSK modcods

**Figure 4.5:** Sampling rate and its effect on degradation for **DVB-S2 modcods**. This evaluation is carried out on a pure Gaussian channel without any synchronization processes, employing a ($\text{RRC}_{0.25}$) filter.

## 4.2 Discussion

### 4.2.1 About uniform sampling

Uniform sampling appears as a **straightforward and resource-efficient solution** that is **well understood and mastered in digital communication systems**. Furthermore, the hardware implementations of resamplers, such as polyphase filters, are extremely efficient today and allow for high-frequency operations and the processing of increasing data rates. However, abuse will rapidly introduce inter-symbol interference into the signal, resulting in quick and high degradation. When used properly, **its efficiency makes it a viable first step in a more complex compression system**.

### 4.2.2 About non-uniform sampling

Nonuniform sampling, unlike uniform sampling, where samples are collected at regular time intervals, involves gathering data points at irregular or variably spaced intervals. This method is especially beneficial in scenarios where signal characteristics fluctuate over time or in applications where capturing every data point is not practical due to constraints such as power, memory, or processing capabilities.

**Existing Methods:**

- **Random Sampling [98]:** Samples are taken at random intervals. This method is often used in stochastic signal processing and image processing, representing a specific case of Compressed Sensing (CS).
- **Adaptive Sampling [99]:** The sampling rate is adjusted based on the characteristics of the signal. More samples are taken during periods of rapid change and fewer when the signal is stable, optimizing data storage and processing .
- **Event-Triggered Sampling [100]:** Sampling is triggered by specific events or thresholds within the signal, ensuring critical changes are captured.
- **Compressed Sensing [77]:** Allows the reconstruction of sparse signals from fewer samples than traditionally required by the Nyquist rate, relying on the sparsity principle. These methods will be elaborated on in subsequent chapters.

**Limitations:**

- **Reconstruction Complexity:** Signal reconstruction from nonuniformly sampled data can be computationally more complex than uniform sampling.
- **Dependency on Signal Characteristics:** The efficiency of nonuniform sampling greatly depends on the nature of the signal.
- **Requirement for Prior Knowledge:** Some methods may need prior knowledge of signal characteristics.
- **Hardware Constraints:** Implementing nonuniform sampling might require specialized hardware for variable sampling rates, which can be more complex and costly.

Given the specific characteristics of communication signals, such as the difficulty in identifying temporally significant zones, the optimal use of bandwidth, and existing hardware constraints, **it seems challenging to envision nonuniform sampling that brings about any significant improvement without a reasonable resource cost and without changing the entire reception chain (including the ADC).** Thus, the study of nonuniform sampling methods is not further pursued in this manuscript. However, Chapter 6 contains a note on compressed sensing usage & limitations for our applications.

## 4.3 Conclusion & actual usage

Resampling is a fundamental method in digital signal processing that is essential for efficient data management and transfer in a variety of applications, such as audio processing and telecommunications. Its main advantage is the ability to lower the data rate by altering the sampling rate, a method widely used in high-quality audio compression [101], [102].

Adapting this approach for use in telecommunications, however, introduces distinctive difficulties. Unlike audio signals, where certain frequencies can be preferentially sampled or omitted without significant loss of quality, telecommunication signals distribute data uniformly across the spectrum. This uniform distribution means that reducing the sampling rate, without careful consideration, can lead to immediate and substantial increases in the Bit-error rate (BER), as demonstrated in Figure 4.5. Rapid degradation of signal quality underscores the limitations of straightforward resampling applications in this field.

**Conclusion about resampling techniques**

Similar to quantization methods, the use of only resampling strategies for compression is challenging and tends to severely deteriorate BER. Additionally, these methods are highly sensitive to effects such as the Doppler shift. Nevertheless, when applied in accordance with Shannon's theorem, it continues to be an effective approach for reducing the overall volume of data that need to be transmitted.

**For these reasons, it is advisable to restrict resampling to a "safe" frequency band, –such as the roll-off value of a RRC filter–, and finalize compression with a different method.**

Part II

# Compression by pattern recognition & dictionary construction

When talking about data compression, various methodologies leverage dictionaries and patterns to achieve efficient signal representation. Traditional techniques, such as those based on well-known compression algorithms (BZip, Deflate, ...), are complemented by more nuanced approaches like *sparse coding*, which exploit the inherent sparsity in data to represent signals using a minimal number of nonzero coefficients, often leading to substantial compression gains without significant loss of information. The following chapters delves into these diverse methods, highlighting both widely-used and subtle techniques, to provide a comprehensive understanding of the state-of-the-art in signal compression.

Chapter 5

# Dictionary-based lossless compression

*To compress things like grandpa did.*


5.1 Introduction & overview . . . . . **88**
5.1.1 LZ algorithms & their derivatives . . . . . 88
5.1.2 Scope and significance in telecommunications . . . . . 89
5.2 Methodology . . . . . **89**
5.2.1 Simulation environment . . . . . 89
5.2.2 Influence of signal parameters . . . . . 91
5.2.3 A word about binary data representation . . . . . 93
5.3 Results . . . . . **94**
5.3.1 Performance report . . . . . 94
5.3.2 Discussions . . . . . 97
5.4 Summary of key findings & conclusion . . . . . **98**


## 5.1 Introduction & overview

### 5.1.1 LZ algorithms & their derivatives

Dictionary-based methods have emerged as a cornerstone technique, offering robust and efficient solutions to reduce data size. Among these, the Lempel-Ziv (LZ) algorithms stand out for their elegance and effectiveness. This chapter aims to delve into the principles and applications of dictionary compression methods, with a particular emphasis on Lempel-Ziv algorithms.

At its core, dictionary compression operates by replacing sequences of symbols in the input data with shorter references to a dictionary constructed during the compression process. Unlike methods that rely on statistical properties of the input, dictionary techniques are adaptive; **they build the dictionary dynamically, based on the observed data stream**. This adaptability makes them particularly suitable for data with **no prior statistical knowledge or for sources with varying statistical properties**.

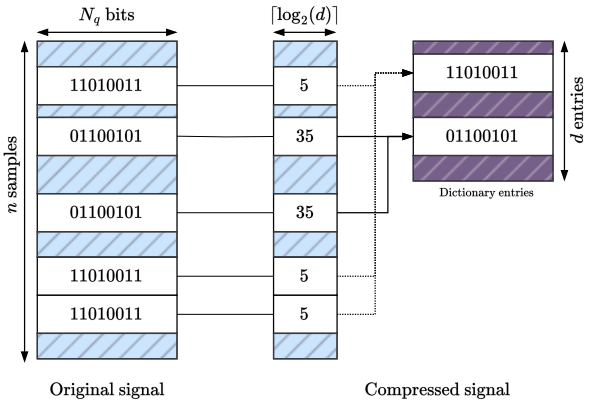



**Figure 5.1:** Illustration of the Lempel-Ziv compression process. The input data are scanned, and recurring patterns are replaced with references to a dictionary. This process results in a compressed output that can be efficiently decompressed using the same dictionary.

The Lempel-Ziv family of algorithms [40], [41], developed by Abraham Lempel and Jacob Ziv in the late 1970s and early 1980s, exemplifies the power of dictionary-based compression. These algorithms have found widespread application, on the basis of popular formats like GIF and PNG, and influence file compression tools such as *gzip* and *zip*. The LZ algorithms are celebrated for their simplicity, efficiency, and the fact that they do not require prior knowledge of the statistical properties

of the input data. They work by iteratively scanning the input data, creating a dictionary of unique patterns, and then replacing recurring patterns with shorter references to this dictionary. What makes dictionary methods, particularly Lempel-Ziv algorithms, suitable for a broad spectrum of data types is their agnostic nature. They can efficiently compress text, images, and even complex data formats. The universality of these algorithms lies in their ability to **identify and exploit redundancy**, which is a common characteristic in various data forms, from plain text in natural languages to pixel patterns in images.

In this chapter, we will explore the intricate workings of dictionary compression methods, with a detailed examination of the Lempel-Ziv algorithms. We will dissect the algorithmic strategies employed, analyze their performance and efficiency, and consider their applications in the context of sampled RF signals. This exploration will not only enhance the understanding of these methods, but will also provide insight into selecting appropriate compression techniques for specific data types and applications.

### 5.1.2 Scope and significance in telecommunications

In telecommunications, we are often dealing with data streams that have well-defined structures and patterns. This is mainly due to how we encode and modulate data for transmission. Even if transmitted bits are random, the resulting frequency spreading of data, their variation, and their statistical properties are not. This is particularly true for digital communication systems, where data is often encoded and modulated using specific schemes, leading to predictable patterns and structures in transmitted data.

For instance, GMSK modulation belongs to a group of waveforms known as "constant envelope" waveforms. This characteristic provides a notably steady and statistically foreseeable amplitude and dynamic range to the signal. This stability is not present in audio signals and should be utilized to the fullest to take advantage of its distinctiveness when trying to compress GMSK waveforms samples.

## 5.2 Methodology

### 5.2.1 Simulation environment

This section outlines the simulation environment used to evaluate dictionary compression algorithms. The environment is designed to accurately assess the performance of these algorithms under various conditions. The key components of the simulation environment include:

#### 5.2.1.1 Signal generation and noise addition

The initial signal for the simulation is generated using MATLAB®, following the specifications outlined in the introduction section of this document. To simulate real-world conditions, AWGN noise is artificially introduced into the generated signal using Python. This approach allows for a controlled environment where the impact of noise on compression efficiency can be precisely measured.

#### 5.2.1.2 Algorithms implementation

The compression algorithms under test are based on the efficient implementations provided by Tino Reichardt [43], [103]. These implementations are noteworthy for their performance and flexibility. Notably, **Reichardt's work enables these algorithms to be used as extensions for the 7-zip archiving tool**, facilitating their application on binary data directly.

In addition to the classical *LZ77* and *LZ78* algorithms, the implementation also includes more modern algorithms such as Brotli, BZip2, and LZMA. Table 5.1 presents some of the algorithms tested

in the simulation environment and their main characteristics.

**Table 5.1:** List of some efficient dictionary compression algorithms

| **Algorithm** | **Description** |
|---|---|
| LZW [104] | *Lempel-Ziv-Welch* is known for its balance between speed and compression efficiency. Originally designed to improve on its predecessors, it is behind the *CompuServe GIF* image format and has been used in *TIF* images and PDF files due to its byte-oriented compression capability. |
| gZIP | Utilizes the *LZ77* algorithm and *Huffman coding* via the *DEFLATE* method. It is a standard for file compression that seamlessly handles byte streams. *gZIP* creates a dynamic dictionary during compression, improving the compression efficiency between various types of data and formats. |
| LZ4 | Designed for maximum speed, LZ4 is favored in scenarios requiring high compression and decompression rates, such as in-memory caches or real-time communication systems handling binary data streams. Its simplicity allows it to run on FPGA [105]. |
| zSTD [106] | Optimized for high compression ratios with a scalable dictionary. *zStandard* is well suited to compress large volumes of binary data, *zStandard* dynamically adjusts the dictionary based on the processed data, maintaining high efficiency across diverse datasets. |
| Brotli [107], [108] | Excels in environments prioritizing compression ratio, such as web asset compression. Its effectiveness in handling binary content makes it suitable for compressing mixed media formats or binary executable files. |
| LZMA | Known for its superior compression ratio, *LZMA* is preferred for software distribution, packing binary data into smaller archives, especially where bandwidth or storage savings are crucial. It is the algorithm behind the popular *7z* software. |
| Bzip2 | Bzip2 [109, Section 6.26] is a high-quality open-source data compression algorithm that utilizes the Burrows-Wheeler transform ($BWT$) [110] and *Huffman coding* [31] to achieve significant compression ratios. It is known for its efficiency in compressing large files. |

#### 5.2.1.3 Data representation

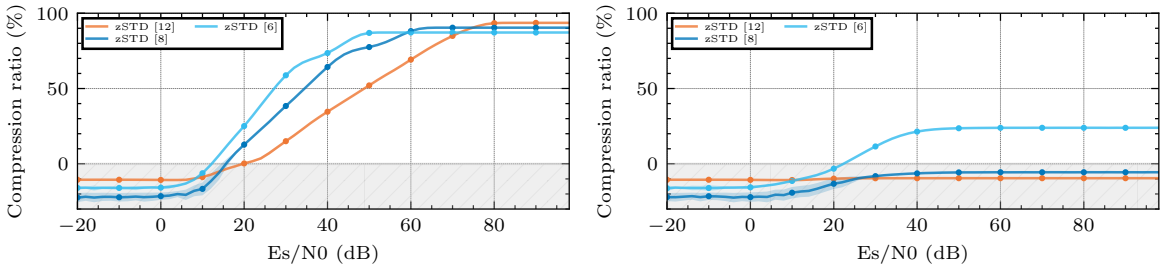
Handling complex data requires careful consideration, particularly regarding the representation of real and imaginary components. In the simulation environment, these components are systematically arranged in a specific format before compression. This method ensures that the integrity of complex data is maintained throughout the compression process. For further details on the binary data representation approach adopted, the reader is referred to Section 5.2.3. This section provides comprehensive information on the rationale behind the chosen data arrangement and its impact on compression outcomes.

Together, these elements form a robust simulation environment that is pivotal for performing an in-depth analysis of dictionary compression algorithms. By meticulously controlling factors such as noise addition and data representation, the simulation environment enables a thorough evaluation of algorithm performance across a range of scenarios.

### 5.2.2 Influence of signal parameters

#### 5.2.2.1 Quantization depth $N_q$

A higher number of quantization bits will result in a higher fidelity of the signal, but will also result in a higher number of bits to compress. This will result in a higher dictionary size and thus a lower compression ratio.




**(a)** QPSK | NRZ **(b)** 8-PSK | $\text{RRC}_{0.25}$

**Figure 5.2:** Compression ratio evolution for different waveforms with different numbers of quantization bits (from 6 to 12 bits per sample).

Figure 5.2 illustrates the changes in the compression ratio between the different waveforms as the number of quantization bits varies. It can be seen that the compression ratio is higher with fewer quantization bits and decreases rapidly as the number of quantization bits increases. This occurrence, as previously observed, is due to the fact that a lower quantization depth leads to fewer patterns and, therefore, to a smaller generated dictionary. Regarding SNR, a rapid decrease in the compression ratio is observed as the noise power increases; we observe that no further compression occurs when $\texttt{E}_\texttt{s}/\texttt{N}_0$ is below 20 dB. **Furthermore, the compression ratio and performance of the zSTD algorithm appear to be strongly influenced by the shaping used –RRC shaping in Figure 5.2–. We will revisit this topic later.**

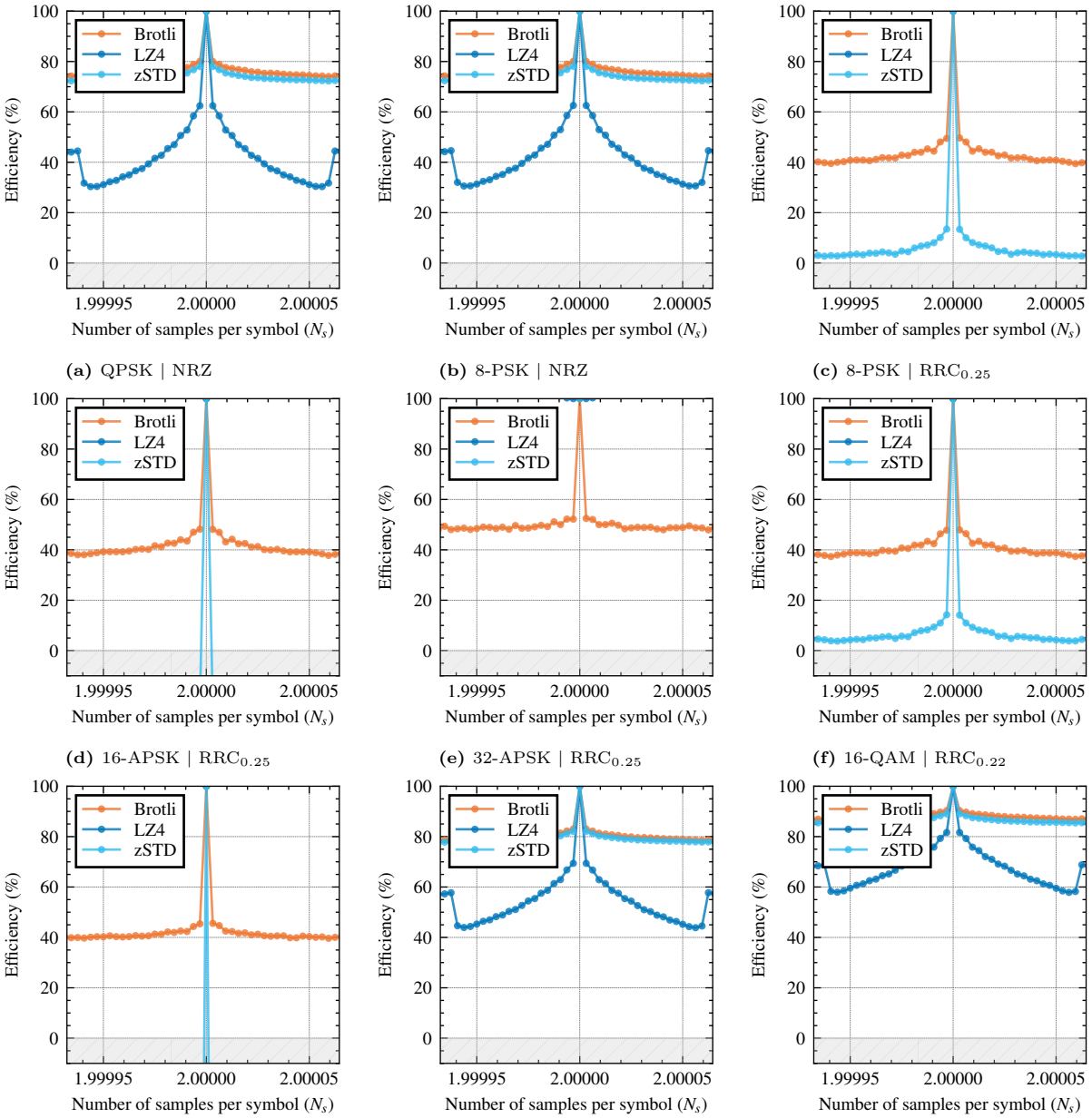
#### 5.2.2.2 Number of samples per symbol $\mathcal{F}$

Dictionary compression techniques are highly sensitive to the oversampling ratio (the number of samples per symbol) $\mathcal{F}$. As depicted in Figure 5.3, the compression effectiveness varies with different signals and their respective number of samples per transmitted symbol. It is evident that the compression ratio increases significantly when the number of samples per symbol has an integer value. This can be attributed to the algorithm's ability to identify and utilize efficient, recurring patterns within the data. These patterns are prone to vanish or reduce in length as the number of samples for each symbol decreases.

Certainly, if the count of samples for each symbol does not have an integer value, the algorithm will fail to identify repetitive patterns in the data, thereby making it impossible to compress. This is the reason why our tests will exclusively use integer values for samples per symbol.

**Sampling rate condition**

Dictionary compression methods need an integer number of samples per symbol $\mathcal{F}$ to work at full capacity.




**(a)** QPSK | NRZ **(b)** 8-PSK | NRZ **(c)** 8-PSK | $\mathrm{RRC}_{0.25}$

**(d)** 16-APSK | $\mathrm{RRC}_{0.25}$ **(e)** 32-APSK | $\mathrm{RRC}_{0.25}$ **(f)** 16-QAM | $\mathrm{RRC}_{0.22}$

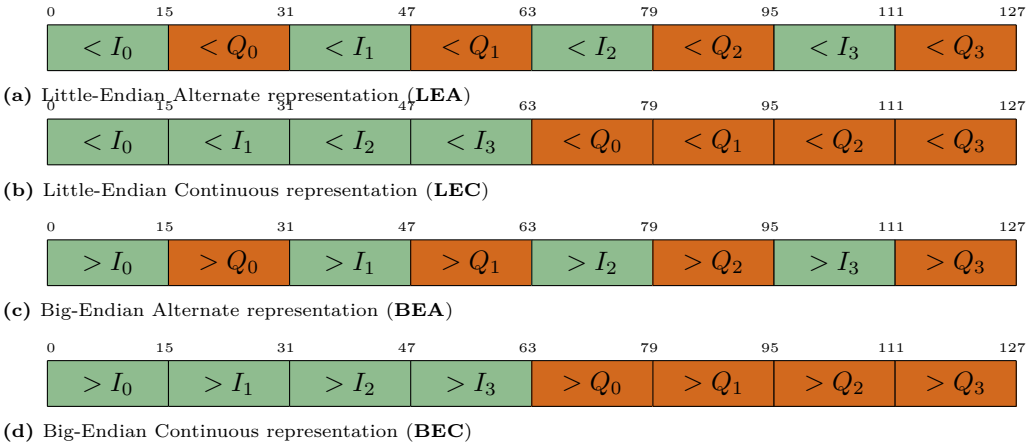
**(g)** 64-QAM | $\mathrm{RRC}_{0.22}$ **(h)** GMSK | BT 0.5 **(i)** GMSK | BT 0.3

**Figure 5.3:** Compression efficiency for different waveforms with different numbers of samples per symbol. Compression efficiency, within a parameter range, is determined by the ratio of the actual compression ratio to the maximum achievable compression ratio within the specified parameter range.

### 5.2.3 A word about binary data representation

Considering that we are dealing with binary information, it is essential to establish the data representation before the compression process. Due to quantization, the data intended for compression may not necessarily be in multiples of 8 bits. However, conventional algorithms such as Lempel-Ziv operate on bytes. Therefore, it is necessary to determine the data representation method that aligns with these algorithms for compatibility.

To do that, data are extended to 16 bits (by adding leading zeros), and then algorithms **that use an entropy coder** are used to compress the data. The entropy coder will compress the nonused range of values. In this way, we can represent data of any size. Different representations have been tried to see how this parameter impacts the compression ratio. Figure 5.4 shows the different representations tried.




**(a)** Little-Endian Alternate representation (**LEA**)

**(b)** Little-Endian Continuous representation (**LEC**)

**(c)** Big-Endian Alternate representation (**BEA**)

**(d)** Big-Endian Continuous representation (**BEC**)

**Figure 5.4:** Different representations of complex data. Symbol $<$ means that samples are stored in *little endian* format, and symbol $>$ means that samples are stored in *big-endian* format

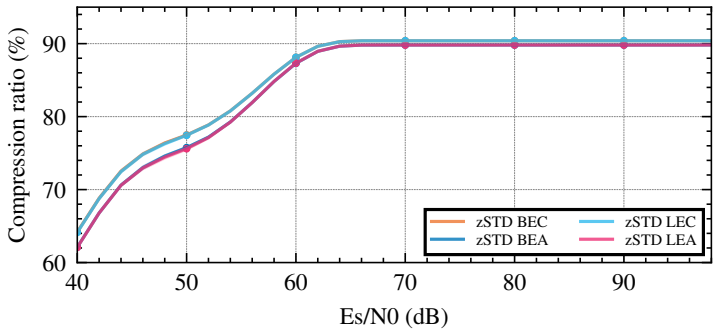



**Figure 5.5:** Compression ratio $\mathcal{C}$ depending on the chosen representation & SNR

Figure 5.5 illustrates the compression ratio based on the selected representation. It is evident that all representations achieve similar compression ratios, but both **BEC** and **LEC** exhibit a slightly higher compression ratio compared to each other. This underlines that continuous representation is the best choice when using this kind of algorithm. **Furthermore, the choice between *little-endian* or *big-endian* representation does not appear to influence compression efficiency**.

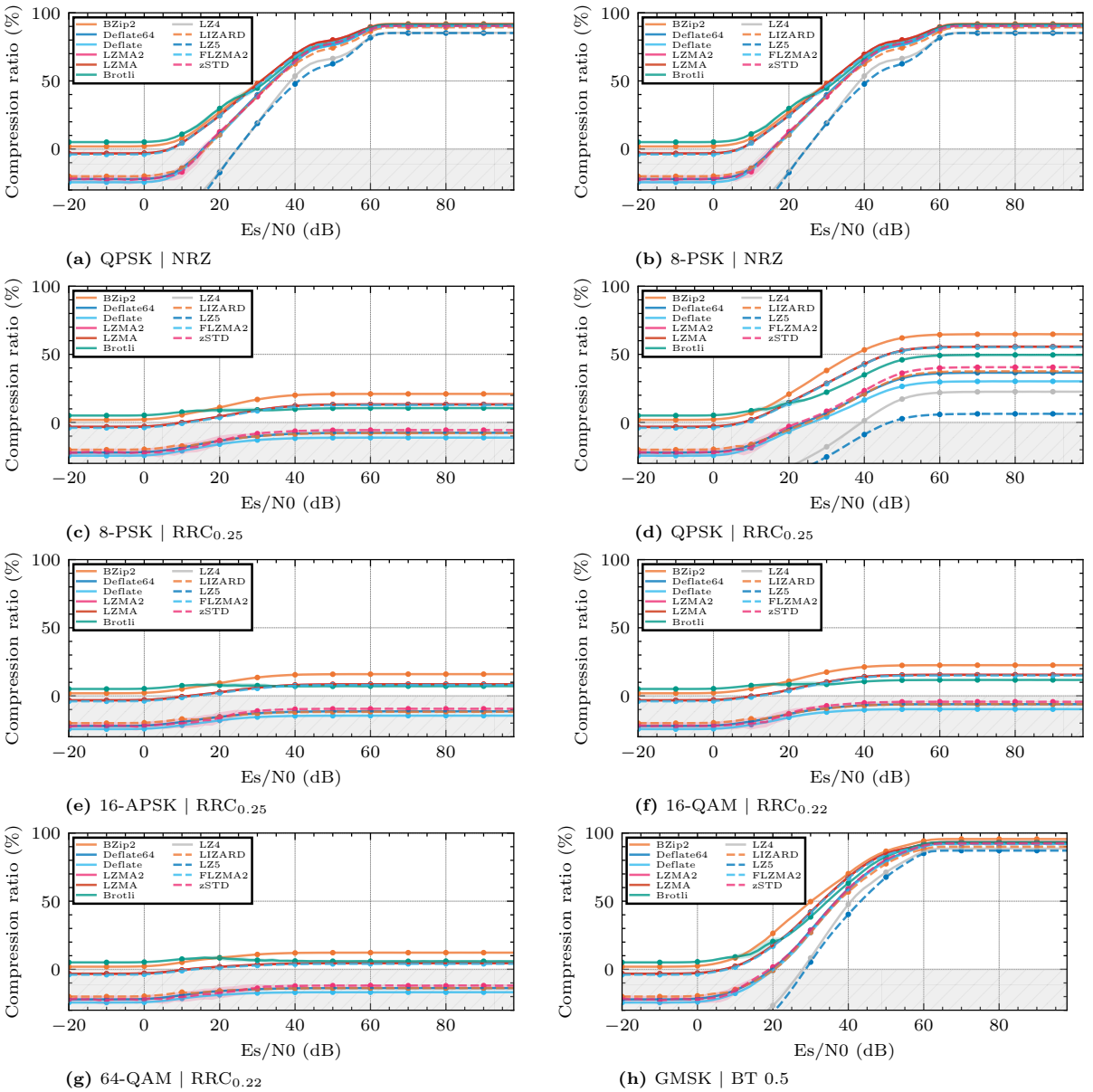
The results presented in Section 5.3 only present the performance obtained for the best representation for each algorithm. However, all representations have been tested.

## 5.3 Results

### 5.3.1 Performance report

#### 5.3.1.1 Compression ratio

*Compression Ratio* metric measures the effectiveness of a compression algorithm in reducing data size. It is defined as the percentage of size reduction achieved by the compression process. A value closer to 100% indicates a greater reduction, highlighting significant savings in storage and transmission costs.




**(a)** QPSK | NRZ **(b)** 8-PSK | NRZ

**(c)** 8-PSK | $\mathrm{RRC}_{0.25}$ **(d)** QPSK | $\mathrm{RRC}_{0.25}$

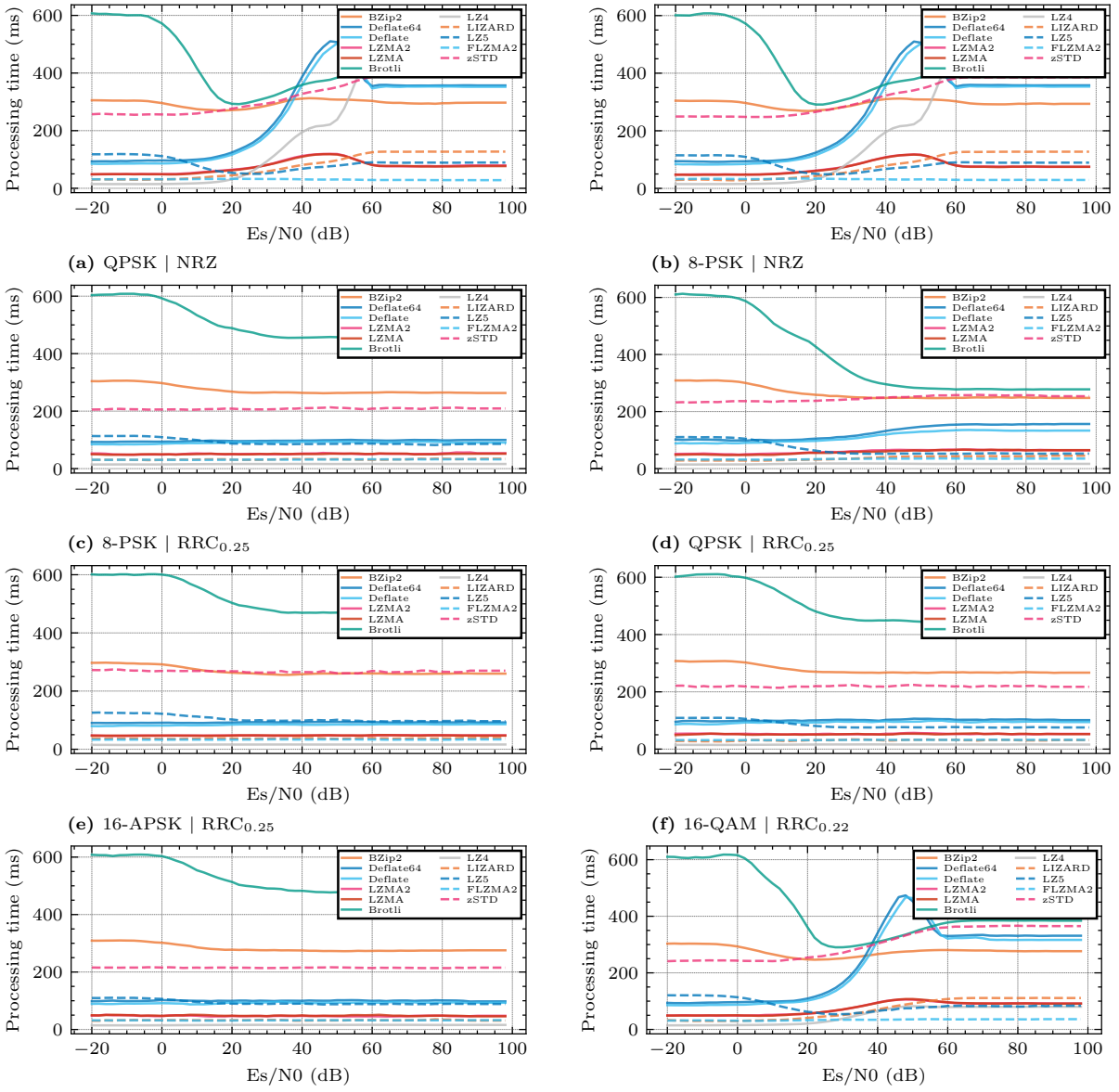
**(e)** 16-APSK | $\mathrm{RRC}_{0.25}$ **(f)** 16-QAM | $\mathrm{RRC}_{0.22}$

**(g)** 64-QAM | $\mathrm{RRC}_{0.22}$ **(h)** GMSK | BT 0.5

**Figure 5.6:** Evolution of compression ratio $\mathcal{C}$ –as defined in Section 1.5.1.4– for different waveforms and value of $\mathtt{E_b/N_0}$ in dB. All signals are sampled on 12 bits and 2 samples per symbol.

#### 5.3.1.2 Total processing time

*Total Processing Time* metric measures the total time required for both compression and decompression processes. It is simply the sum of the compression time and decompression time, providing a comprehensive measure of the overall time efficiency of a compression algorithm.




**(a)** QPSK | NRZ

**(b)** 8-PSK | NRZ

**(c)** 8-PSK | $\mathrm{RRC}_{0.25}$

**(d)** QPSK | $\mathrm{RRC}_{0.25}$

**(e)** 16-APSK | $\mathrm{RRC}_{0.25}$

**(f)** 16-QAM | $\mathrm{RRC}_{0.22}$

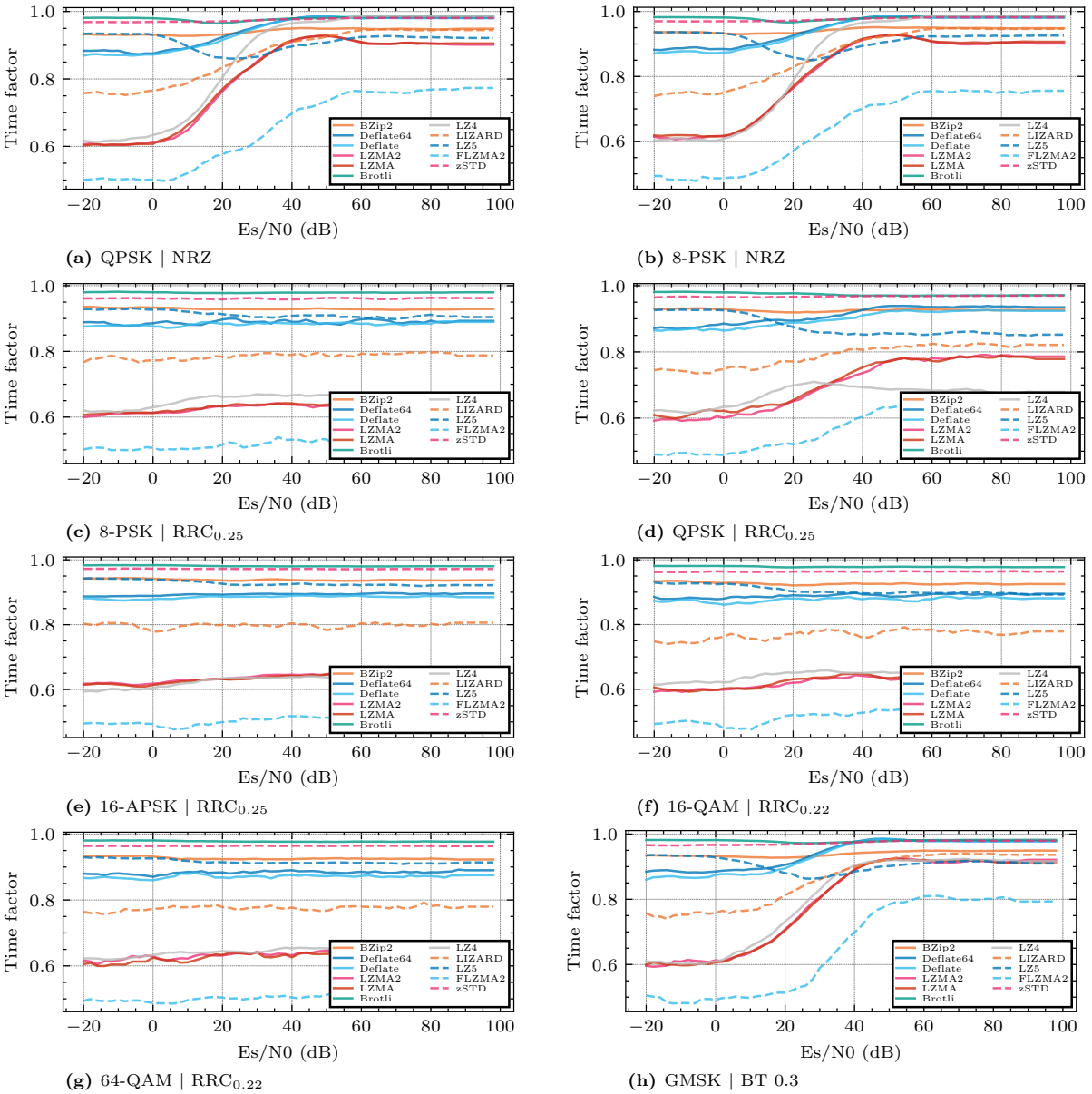
**(g)** 64-QAM | $\mathrm{RRC}_{0.22}$

**(h)** GMSK | BT 0.3

**Figure 5.7:** Total processing time for different waveforms and value of $\mathtt{E_b/N_0}$ in dB. All signals are sampled on 12 bits and 2 samples per symbol.

#### 5.3.1.3 Time factor

*Time Factor* metric quantifies the efficiency of the compression-decompression cycle in terms of time. It is calculated as the difference between compression and decompression times divided by their sum. A value less than 0 indicates faster decompression, while a value greater than 0 suggests longer decompression times, useful for comparing time efficiency of algorithms.




**(a)** QPSK | NRZ

**(b)** 8-PSK | NRZ

**(c)** 8-PSK | $\mathrm{RRC}_{0.25}$

**(d)** QPSK | $\mathrm{RRC}_{0.25}$

**(e)** 16-APSK | $\mathrm{RRC}_{0.25}$

**(f)** 16-QAM | $\mathrm{RRC}_{0.22}$

**(g)** 64-QAM | $\mathrm{RRC}_{0.22}$

**(h)** GMSK | BT 0.3

**Figure 5.8:** Evolution of Time factor $\mathcal{T}_F$ –as defined in Section 1.5.1.6– for different waveforms and value of $\mathtt{E_b/N_0}$ in dB. All signals are sampled on 12 bits and 2 samples per symbol.

### 5.3.2 Discussions

The first thing we can notice is that the compression ratio is really linked to the kind of shaping applied to the signal. Indeed, we can see that the compression ratio is really high for NRZ/RZ signals whereas it appears more moderate for other types of shaping, such as RRC.

#### 5.3.2.1 NRZ/RZ shaped signals

Non return to zero (NRZ) and return-to-zero Return to zero (RZ) shaped signals exhibit distinctive characteristics when subjected to compression using the Lempel-Ziv algorithms. The primary advantage of compressing NRZ/RZ shaped signals with these algorithms lies in the inherent structure of the data. Given that NRZ/RZ signals do not exhibit a large data spread, Lempel-Ziv algorithms can efficiently identify and exploit repetitive patterns within the signal, leading to significant data compression.

The effectiveness of compression, measured by the compression ratio, is notably high when the Signal-to-Noise Ratio (SNR) is high. In high SNR conditions, Lempel-Ziv algorithms demonstrate remarkable efficiency in compressing NRZ/RZ shaped signals. On the contrary, at low SNR levels, the performance of these algorithms deteriorates significantly. The compression ratio plummets, indicating a pronounced decrease in efficiency. It is important to recognize that while all algorithms possess varying operational ranges, certain algorithms retain the capability to achieve compression at an $\mathtt{E_b}/\mathtt{N}_0$ approaching 10 dB, like LZMA and BZip2.

#### 5.3.2.2 RRC shaped signals

(Square)-Root-raised-cosine filter (RRC) shaped signals present a contrasting scenario when subjected to compression using Lempel-Ziv algorithms. The dispersion of data inherent in RRC shaped signals raises a considerable challenge for these algorithms, hindering their ability to discern efficient patterns for compression. As a result, the compression ratio for RRC shaped signals is markedly low, underscoring the inefficacy of Lempel-Ziv algorithms in this context.

However, the potential to achieve a degree of compression exists, particularly when the BZip2 algorithm is employed. Despite the general difficulties associated with compressing RRC-shaped signals, BZip2 can attain a compression ratio ranging between 20% and 30% under high SNR conditions. This demonstrates that, depending on specific application requirements, a modest, yet noteworthy level of compression may be achievable.

The relationship between the modulation index and the compression ratio further illustrates the challenges of compressing RRC shaped signals. As the modulation index increases, the compression ratio correspondingly decreases. For example, signals modulated with 64-QAM are subject to less compression than those with 16-QAM, and similarly, 32-APSK signals experience lower compression ratios compared to 16-APSK signals, even in scenarios characterized by high SNR. This dynamic underscores the nuanced interplay between signal characteristics and compression algorithm performance.

#### 5.3.2.3 Gaussian shaped signals (GMSK like)

Given that the TX constellation of GMSK includes a limited set of symbols with no substantial dispersion around them, the findings are identical to those for the NRZ shaped signal. However, this conclusion should be approached with caution as it does not apply to all types of CPM signals, given that the phase of the symbol is significant in pattern recognition.

## 5.4 Summary of key findings & conclusion

This chapter has explored the dynamics and efficiency of dictionary compression algorithms in different signal modulation formats. Our findings underscore the critical impact of the number of samples per symbol on the effectiveness of these algorithms. A notably high compression ratio is achieved when this number is an integer. However, this ratio declines precipitously when the number is noninteger. This sensitivity is paramount in real-world applications where controlling this parameter precisely can be challenging, especially on the receiver side, where factors such as Analog-to-Digital Converter (ADC) specifications and sampling clocks play significant roles.

For NRZ/RZ-shaped signals, the Lempel-Ziv algorithms exhibit excellent performance, achieving compression ratios as high as 90% at high SNR. However, their effectiveness wanes substantially at lower SNR levels, although some algorithms such as Brotli, BZip2, and LZMA manage to maintain compression capabilities down to $E_s/N_0$ close to 10 dB.

On the contrary, signals shaped with (Square)-Root-raised-cosine filter (RRC) filters present a formidable challenge for these algorithms, with generally low compression ratios. Despite this, the BZip2 algorithm shows potential in delivering compression between 20% and 30% at high SNR levels, although this is highly dependent on the modulation index, with higher indices resulting in poorer performance.

**Conclusion:** Dictionary-based compression algorithms are optimally used on the transmitter side (TX), particularly with NRZ / RZ signals under high SNR conditions. Although their application to RRC-shaped signals is feasible, especially with the BZip2 algorithm, the performance remains sub-optimal. The inherent limitations of these general-purpose algorithms in handling complex data types suggest a pressing need for the development of specialized algorithms or enhancements to existing ones to better accommodate the nuances of digital communications. This could potentially lead to significant improvements in data compression practices, particularly in telecommunications, where the efficiency of data transmission and storage is paramount.

**Conclusion: Lossless compression capability**

The ability to compress a baseband signal without loss using dictionary algorithms is mainly influenced by the number of samples per symbol $\mathcal{F}$ and the type of shaping filter used. Techniques such as BZip2 or zSTD provide effective compression for TX signals. However, its application in RX appears to be challenging **if not of interest, in most industrial cases**

.

Chapter 6

# Sparse coding & sparse dictionary learning

***Sparsity compression in high dimension***


6.1 Presentation . . . . . **99**
6.1.1 Introduction to sparse coding . . . . . 99
6.2 Dictionary creation . . . . . **100**
6.2.1 Exposition & pertinence . . . . . 100
6.2.2 Data preprocessing . . . . . 101
6.2.3 Choice of learning algorithm . . . . . 102
6.3 Sparse coefficients: analysis & compression . . . . . **103**
6.3.1 One problem for three equations . . . . . 103
6.3.2 Managing loss with sparse coding . . . . . 104
6.3.3 Sending sparse data over the network . . . . . 105
6.4 Compression chain . . . . . **107**
6.5 Simulations . . . . . **109**
6.6 Results & discussions . . . . . **110**
6.6.1 Algorithms based on $\ell_0$ norm . . . . . 110
6.6.2 Alternatives & exotic algorithms . . . . . 112
6.6.3 A word about hardware implementation . . . . . 115
6.7 Conclusion . . . . . **116**


*Results and analysis in this chapter has been presented in [111] during EUSIPCO International Conference in August 2024.*

## 6.1 Presentation

### 6.1.1 Introduction to sparse coding

Sparse Coding is a powerful signal processing technique that aims to represent signals as a linear combination of a small number of elementary building blocks, also known as atoms, from a dictionary. Mathematically, given a signal $\boldsymbol{y} \in \mathbb{R}^M$ and a dictionary $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{M \times K}$, where $K \geq M$, the goal is to find a sparse representation $\boldsymbol{x} \in \mathbb{R}^K$ such that $\boldsymbol{y} \approx \boldsymbol{\mathcal{D}x}$. The sparsity constraint implies that only a few entries in $\boldsymbol{x}$ are nonzero, making it possible to efficiently represent and transmit the signal using a reduced number of parameters.

In the context of satellite communication systems, sparse coding could be used to find an efficient representation of the received signals using their specific spectral and temporal characteristics. By representing the incoming signals as a sparse combination of atoms in a dictionary $\boldsymbol{\mathcal{D}}$, we can efficiently compress the data transmitted between the antenna and the data center. The main question we will try to answer in this chapter is as follows:

> The main question is then: Given a signal $\boldsymbol{y}$ and a maximum degradation $\Delta$. How can we find the dictionary $\boldsymbol{\mathcal{D}}$ and the associated sparse representation $\boldsymbol{x}$ that guarantee a maximum degradation of $\Delta$?

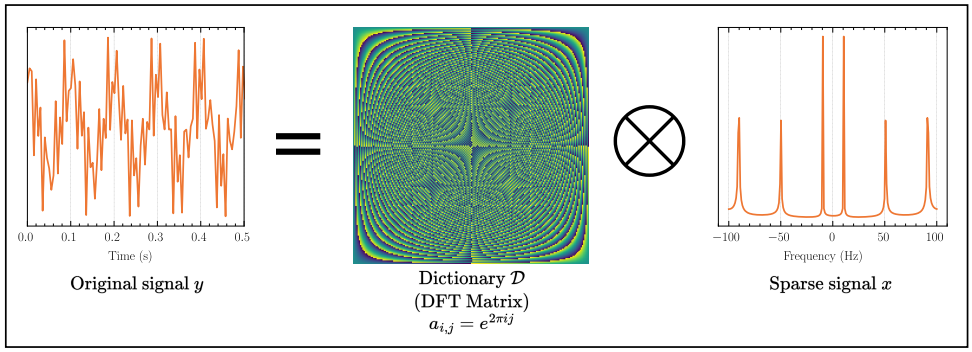



**Figure 6.1:** The simple case of a sinusoidal signal $\boldsymbol{y}$ represented as a sparse combination of atoms in a dictionary $\boldsymbol{\mathcal{D}}$ where $\boldsymbol{\mathcal{D}}$ is the Fourier basis.

## 6.2 Dictionary creation

### 6.2.1 Exposition & pertinence

In dictionary learning, our aim is to find a dictionary $\boldsymbol{\mathcal{D}}_{\tau_0} \in \mathbb{R}^{M \times K}$ and a sparse representation $\boldsymbol{X} \in \mathbb{R}^{N \times K}$ to approximate the input signal $\boldsymbol{Y} \in \mathbb{R}^{N \times M}$. We solve the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{\mathcal{D}}_{\tau_0}, \boldsymbol{X}}{\arg\min} \quad & \frac{1}{2}||\boldsymbol{Y} - \boldsymbol{\mathcal{D}}_{\tau_0}\boldsymbol{X}||_{\mathrm{F}}^2 \\ \text{subject to} \quad & ||\boldsymbol{X}^{[i]}||_0 \leq \tau_0, \\ & ||\boldsymbol{\mathcal{D}}_{\tau_0}^{[i]}||_2 = 1; \quad \forall i \in [\![1, K]\!], \end{aligned} \tag{6.1}$$

Where $||\cdot||_2$ is the Euclidean norm, $||\cdot||_0$ is the pseudo-norm counting nonzero elements in a vector, $\boldsymbol{\mathcal{D}}_{\tau_0}^{[i]}$ is the $i$-th column of $\boldsymbol{\mathcal{D}}_{\tau_0}$, and $\tau_0 > 0$ represents a sparsity constraint parameter. When $K = M$, $\boldsymbol{\mathcal{D}}_{\tau_0}$ is said to be *complete* and when $K > M$, $\boldsymbol{\mathcal{D}}_{\tau_0}$ is said to be *overcomplete*.

Familiar techniques such as K-SVD [112] or Online Dictionary Learning (ODL) [113] are applicable to learn the dictionary $\boldsymbol{\mathcal{D}}$. The choice of method, along with the amount of data at hand, determines whether the Dictionary Learning (DL) process is carried out *online* –direct training on $\boldsymbol{Y}$) or *offline* (using a dataset that simulates the properties of $\boldsymbol{Y}$–. Figure 6.3 presents some example of data reconstruction using sparse coding via a learned dictionary.

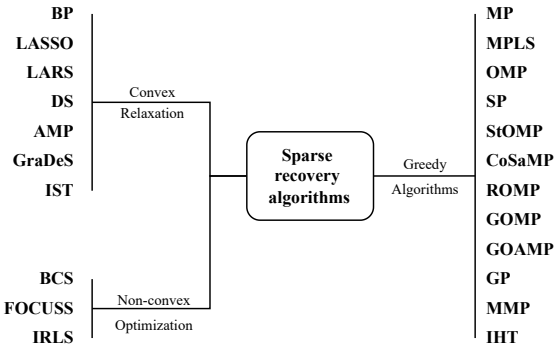



**Figure 6.2:** Categorization of recovery algorithms.
**Source:** *Adapted from [114]* .

Regarding the optimization problem of determining $\boldsymbol{X}$, it is an NP-hard problem [115] and obtaining the exact solution requires testing all feasible sets of values (which is not feasible). However, several sparse coding algorithms aim to find effective approximations of solutions [114]. Among these algorithms, the most famous is *Orthogonal Matching Pursuit (OMP)* [66], [116], while other approaches include *Iterative Hard Thresholding (IHT)* [67], *Generalized Orthogonal Matching Pursuit (gOMP)* [117], CoSaMP [118], Smoothed L0 (SL0) [119]... Some of these algorithms are classified in Figure 6.2.

Other methods such as LASSO [72], Elastic-Net [120] or Coordinate Descent (CD) [113] use $\ell_1$ regularization [69], [70] to achieve the same goal at

lower complexity. We will discuss them later in this chapter.

### 6.2.2 Data preprocessing

Data representation is a preliminary step that needs careful consideration before selecting any specific algorithm for sparse coding and dictionary learning. When dealing with a signal $y$, which is sampled at $\mathcal{F}$ samples per symbol, and has a total length of $\mathcal{L}$, it becomes imperative to define the dimensions of the data to be processed.

We define $M$ as the sparse channel dimension, which subsequently determines the number of channels, $N$, given by the ratio of the total signal length to the channel dimension, ie, $N = \lfloor \mathcal{L}/M \rfloor$.

The choice of $M$ is critical because it aligns with the dictionary atoms across the signal channels. Opting for a larger $M$ enhances the flexibility and compression capability of the system. Therefore, $M$ should ideally be chosen large and as an integer multiple of $\mathcal{F}$ to facilitate the optimal representation of the data and subsequent processing steps.

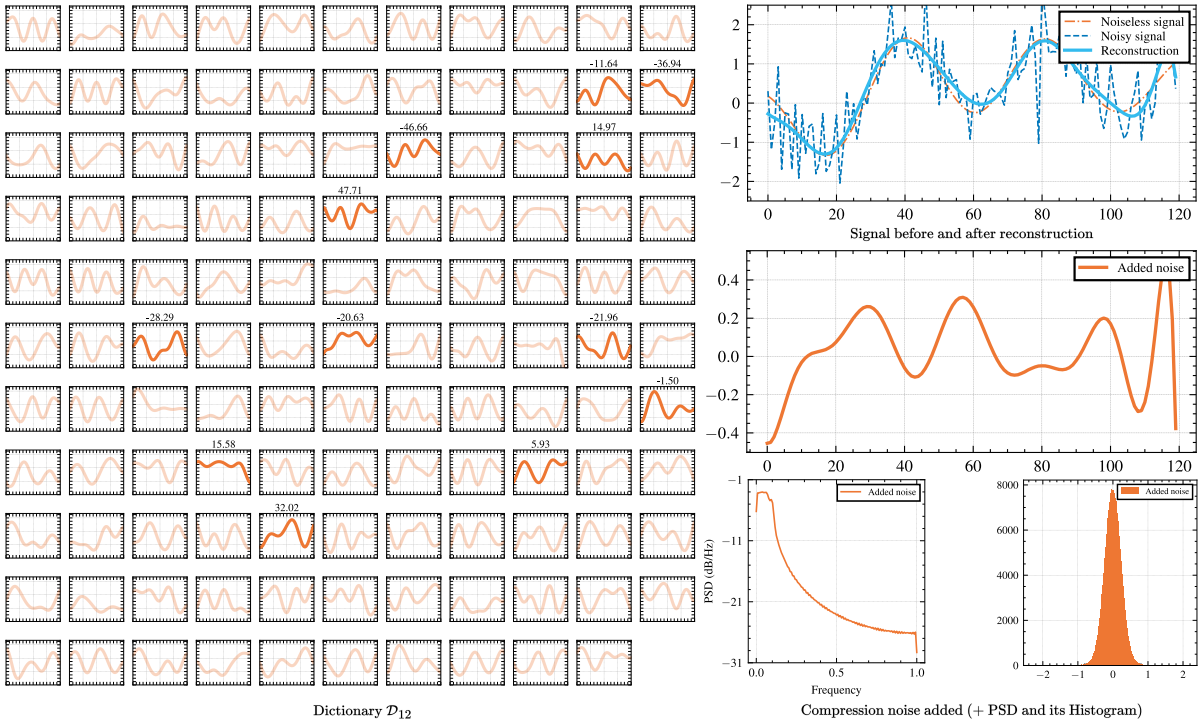



**Figure 6.3:** Example of signal approximation using ODL with OMP and a parameter $\tau_0 = 12$ on a 8-PSK signal sampled at 120 samples per symbols. The atoms of the dictionary and the associated coefficients are represented.

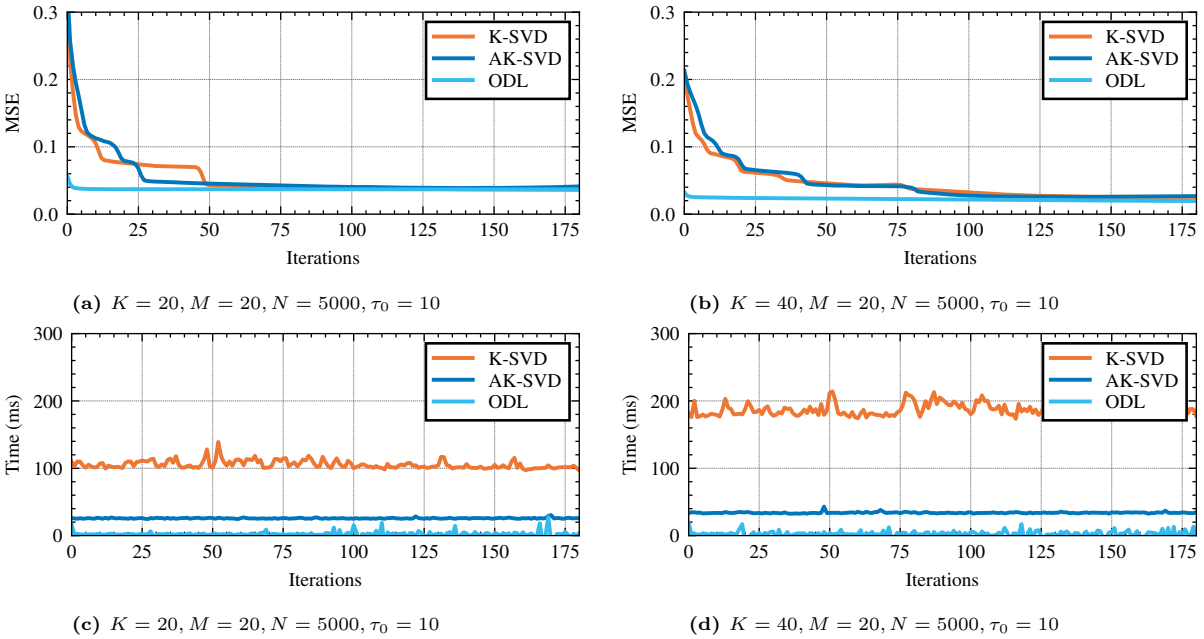
### 6.2.3 Choice of learning algorithm

Identifying an appropriate dictionary for a given dataset is a challenging task and constitutes a distinct research field within a sparse representation. Over the years, numerous algorithms have been proposed and explored to address this problem. The overarching goal of these algorithms is to discover iteratively the sparse representation that best fits the data.

Among the plethora of available algorithms, this thesis focuses on three specific methods. K-SVD [112] and its approximate variant AK-SVD [121], as well as the Online Dictionary Learning (ODL) method [113], [122] developed by Julien Mairal.

- The selection of these algorithms is strategic. ODL is renowned for its efficiency and has been thoroughly optimized and validated in the literature, making it a preferred choice for straightforward implementations using basic algorithms such as OMP or LASSO. However, for applications that require adaptation to more complex or less standard algorithms such as SOMP or gOMP [117], ODL may not always be suitable.
- In contrast, K-SVD, while generally slower and more computationally intensive due to its higher iteration requirement for convergence, offers significant customization flexibility. This makes it particularly valuable for handling complex signal samples [116] and accommodating exotic algorithms not supported by the standard ODL implementations, such as SOMP. This adaptability is critical for dictionary learning in environments with specialized requirements or in research scenarios exploring new sparse coding methodologies.

There is a noticeable variation in the rate of convergence between the different algorithms, as shown in Figure 6.4. We can notice that ODL provides superior results with faster convergence –requiring only 2 iterations to reach the optimal residual norm–. Furthermore, it is the most efficient to run on a CPU. **However, it consumes significantly more memory compared to K-SVD and AK-SVD.**




**(a)** $K = 20, M = 20, N = 5000, \tau_0 = 10$

**(b)** $K = 40, M = 20, N = 5000, \tau_0 = 10$

**(c)** $K = 20, M = 20, N = 5000, \tau_0 = 10$

**(d)** $K = 40, M = 20, N = 5000, \tau_0 = 10$

**Figure 6.4:** Convergence and calculation time of learning algorithms depending on problem conditions & parameters. Dictionaries are trained on a noiseless 8-PSK | $\text{RRC}_{0.25}$ signal sampled at 2 samples per symbol.

## 6.3 Sparse coefficients: analysis & compression

### 6.3.1 One problem for three equations

The task of identifying the sparse representation of a signal can be approached through multiple problem formulations, each suited to different objectives and constraints within the realm of sparse coding. These variations reflect various priorities, such as minimizing the reconstruction error or maximizing the sparsity, affecting both the algorithmic convergence and the quality of the results. The differences between these formulations are presented in Figure 6.5.

**Minimizing the residual norm** The first common formulation involves minimizing the residual norm. This approach aims to find a sparse representation $\boldsymbol{x} \in \mathbb{R}^K$ such that the difference between the original data $\boldsymbol{y} \in \mathbb{R}^M$ and the reconstructed data from the dictionary $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{K \times M}$ is minimized. The problem can be mathematically expressed as follows.

$$\min_{\boldsymbol{x}} ||\boldsymbol{y} - \boldsymbol{\mathcal{D}}\boldsymbol{x}||_2^2 \quad \text{subject to} \quad ||\boldsymbol{x}||_0 \leq \tau_0 \tag{6.2}$$

where $||\boldsymbol{y} - \boldsymbol{\mathcal{D}}\boldsymbol{x}||_2^2$ is the norm of the *residual* vector containing $M$ elements. Note that $\tau_0$ selection is greatly influenced by the SNR when aiming for a particular sparsity level.

**Minimizing the sparsity measure** Alternatively, the problem can be formulated to directly minimize the sparsity of the representation, typically measured by the $\ell_0$ pseudonorm, which counts the number of nonzero coefficients in $\boldsymbol{x} \in \mathbb{R}^K$. The formulation thus focuses on achieving the smallest possible number of nonzero coefficients:

$$\min_{\boldsymbol{x}} ||\boldsymbol{x}||_0 \quad \text{subject to} \quad ||\boldsymbol{y} - \boldsymbol{\mathcal{D}}\boldsymbol{x}||_2^2 \leq \sigma \tag{6.3}$$

providing direct control over the compression ratio while the exact reconstruction quality (i.e., residual norm) is bounded but not minimized.

**Lagrangian formulation** The third common approach uses a Lagrangian formulation, incorporating a regularization parameter $\lambda$ that balances the sparsity of the representation and the fidelity of the reconstruction. This formulation is expressed as follows.

$$\min_{\boldsymbol{x}} \frac{1}{2}||\boldsymbol{y} - \boldsymbol{\mathcal{D}}\boldsymbol{x}||_2^2 + \lambda_0||\boldsymbol{x}||_0, \tag{6.4}$$

where the trade-off between the reconstruction error and the sparsity level is adjusted by $\lambda$. **This approach offers a flexible compromise that allows tuning based on specific needs.** It should be noted that reaching a specific sparsity level depends on the choice of $\lambda$, which is affected by SNR. This is also the form used by $\ell_1$ approaches like LASSO:

$$\min_{\boldsymbol{x}} \frac{1}{2}||\boldsymbol{y} - \boldsymbol{\mathcal{D}}\boldsymbol{x}||_2^2 + \lambda_1||\boldsymbol{x}||_1, \tag{6.5}$$

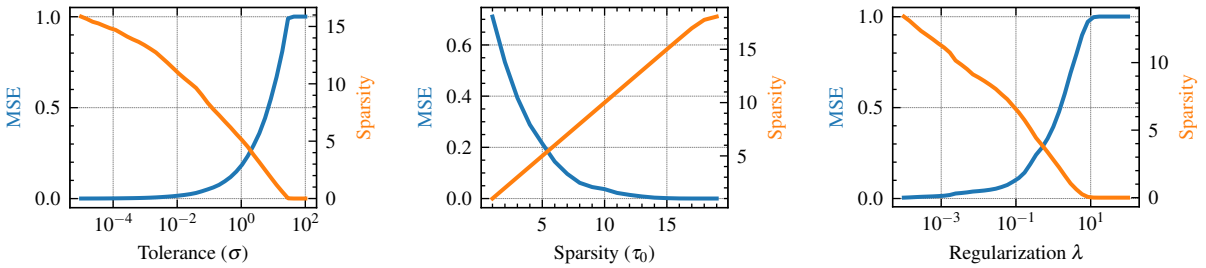



**Figure 6.5:** Changes in sparsity parameters based on the selected formulation and parameter. Dictionaries are trained on a noiseless 8-PSK | $\text{RRC}_{0.25}$ signal sampled at 2 samples per symbol.

**Link with MAP**

The last Lagrangian formulation with the $\ell_1$ norm can also be viewed as a *Maximum A Posteriori* (MAP) approach with a Laplacian prior. In this context, the term $\frac{1}{2}||\boldsymbol{y} - \boldsymbol{\mathcal{D}x}||_2^2$ corresponds to the negative log-likelihood of the data given the coefficients, and the regularization term $\lambda||\boldsymbol{x}||_1$ corresponds to the negative log of the Laplacian prior on the coefficients. This interpretation provides a probabilistic perspective on the trade-off between fidelity and sparsity.

Note that each formulation serves distinct priorities: minimizing the residual norm provides fine control over the reconstruction quality at the potential cost of higher computational complexity and less direct control over compression. Minimizing the sparsity directly influences the compression ratio, but may yield less precise reconstructions. The Lagrangian approach provides a balanced framework, adjustable through $\lambda$ to meet varying requirements between sparsity and reconstruction fidelity.

### 6.3.2 Managing loss with sparse coding

Since only a few entries in the sparse representation vector are nonzero values, the signal can be transmitted or stored using a reduced number of elements, resulting in efficient compression. However, the benefits of sparse representations come with a trade-off: The resulting compression is lossy in nature, which implies that some degree of information loss may occur during the compression/decompression process.

In contrast to image processing, where this compression scheme is already well known and used [123]–[127], digital communications require unbiased and meticulous considerations of this loss, which can then be considered as *compression noise*. Factors such as the noise statistical distribution law and power spectral density (PSD), whether it is white or colored noise, play crucial roles in evaluating the impact of noise on overall system performance. Furthermore, the intricate nature of receivers and the sequence of nonlinear stages, such as **synchronization, interleaving, and error correction codes**, contribute to the highly nonlinear decoding process. As a result, **traditional metrics such as the Mean Squared Error (MSE) or the Mean Absolute Error (MAE) are not enough to provide an accurate analysis of the loss**.

The main conclusion of this study is that the only accurate loss metric is the actual degradation in dB. To measure it, it is necessary to undergo the complete decoding process to accurately measure the Bit-error rate (BER).

To optimize the trade-off between compression efficiency, computation time and reconstruction quality, many parameters can be fine-tuned, such as the number of samples $M$, the dictionary overcompleteness $K$, the number of nonzero coefficients $\tau_0$, or the choice of the sparse coding algorithm. Our research presents results and ways to determine the best parameters depending on the requirements.

### 6.3.3 Sending sparse data over the network

#### 6.3.3.1 Conventional versus novel approaches

Sparse data, by nature, are... sparse, implying that they possess a significant number of zero values. Despite this, it doesn't means these data can be compressed easily, and we have yet to implement any compression techniques. To proceed, we must investigate the methods for compressing these data and the necessary preprocessing steps. In this section, we explore two primary strategies to improve the effectiveness of data compression in the context of sparse data.

1 · The conventional approach, historically adopted within the field, involves the transmission of data in its complete form, incorporating both nonzero and zero values. This method fundamentally relies on entropy coding, like Huffman's method –presented in Section 2.1.1–, utilizing the statistical distribution properties of sparse data to aid in compression. The effectiveness of this strategy primarily depends on the ability of Huffman coding to construct an optimally condensed tree, which effectively encapsulates the distribution between nonzero and zero elements. This is the primary method used in image processing when handling sparse data [123]–[127].

2 · On the contrary, this research introduces an optimized approach, deviating from conventional approaches. This innovative strategy is based on **selective transmission**, where only nonzero samples and their corresponding positions are communicated. Nonzero data elements are further processed through Huffman coding, optimizing transmission efficiency by exploiting their statistical distribution. Similarly, the positions of these nonzero elements are compressed using a dictionary compressor, significantly reducing the overhead associated with their transmission. By avoiding the transfer of zero values and using these two compression methods simultaneously, **it has been noticed that this method inherently reduces the volume of data that needs to be transmitted.**

#### 6.3.3.2 An optimized way to send nonzero values positions

In the pursuit of improving the efficiency of data transfer in sparse representations, a notable difficulty emerges in conveying the locations of nonzero values. Conventional approaches typically require the transfer of entire vectors, encompassing both nonzero and zero elements, leading to less than optimal bandwidth utilization and higher computational load. This section introduces an optimized approach to transmit only nonzero positions, significantly reducing the amount of data that needs to be communicated. By focusing on nonzero elements and their indices, we can leverage the sparse nature of the data more effectively. Figure 6.6 presents the different way we can use to send nonzero positions to the decompressor to reconstruct the data.

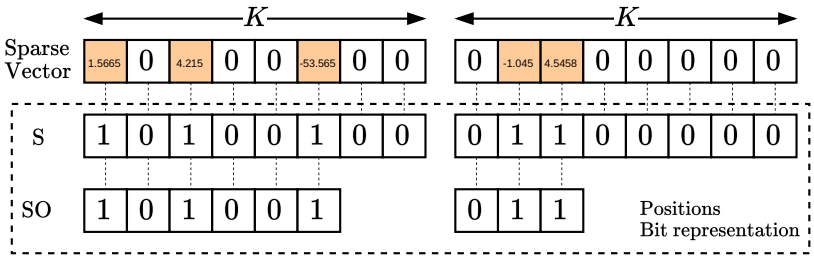



**Figure 6.6:** Two ways to send nonzeros positions when the number of nonzero values is known.

1 · **The S form** represents the most straightforward, albeit less efficient, method for transmitting the positions of nonzero coefficients. In this approach, each vector corresponding to a signal or data block is encoded using $K$ bits. Within this bit vector, a '1' indicates the presence of a nonzero value, and a '0' indicates a zero value. This method, while simple and easy to

implement, does not take advantage of the sparsity inherent in the data, as it encodes positions of both zero and nonzero values. Consequently, the S form **may lead to the transmission of a substantial number of redundant zero indicators**, especially in highly sparse scenarios, thus inefficiently using the communication bandwidth.

2 · **SO form** improves the transmission efficiency by capitalizing on the known sparsity level of the data, indicated by $\tau_0$. This method encodes only the positions of the first $\tau_0$ nonzero values in the bit vector, and the transmission stops as soon as this count is reached. By doing so, the SO form eliminates the need to send the remaining bits after the last nonzero entry, potentially reducing the length of the transmission vector significantly when compared to the S form. This optimization is particularly effective when the nonzero elements are clustered at the beginning of the vector, thus allowing for a much shorter encoded message. However, its efficiency can vary depending on the distribution of nonzero values within the vector.

3 · **SOC or SC forms** further refine the approach by incorporating a secondary compression stage using a dictionary-based lossless compression algorithm (see Chapter 5), such as zSTD [106]. After the positions of the nonzero values are encoded using either the S or SO forms, the resulting bit vector is compressed to remove additional redundancies, effectively reducing the transmission size. This method uses the entropy characteristics of the position data to achieve higher compression rates. The effectiveness of the SOC or SC forms depends on the ability of the dictionary compressor to capitalize on the patterns within the bit vectors, which can vary widely based on the sparsity pattern and the specific characteristics of the data being encoded.

Figure 6.7 demonstrates the reduction in transmission bits achieved through this method, highlighting its potential in scenarios where saving bandwidth is crucial. The figure illustrates the evolution of the number of bits required to transmit the positions of nonzero values for a given channel, the nonzeros elements are randomly distributed.

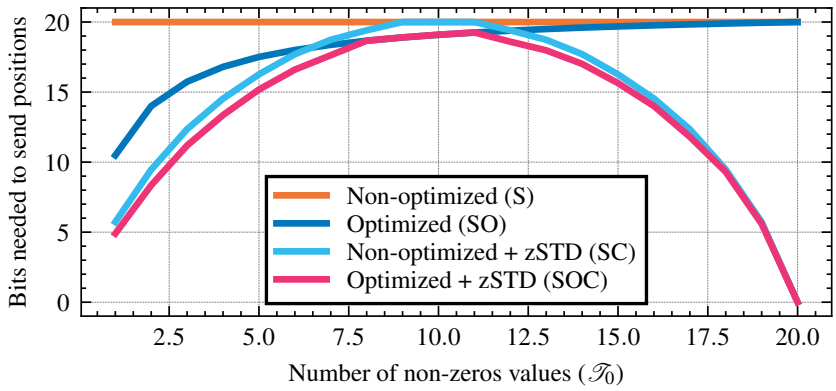



**Figure 6.7:** Evolution of the number of bits to send for each channel in order to transmit the position of nonzeros values to the decompressor ($K = 16$). zSTD could actually be replaced by another compression mechanism (gZIP, LZMA,...).

## 6.4 Compression chain

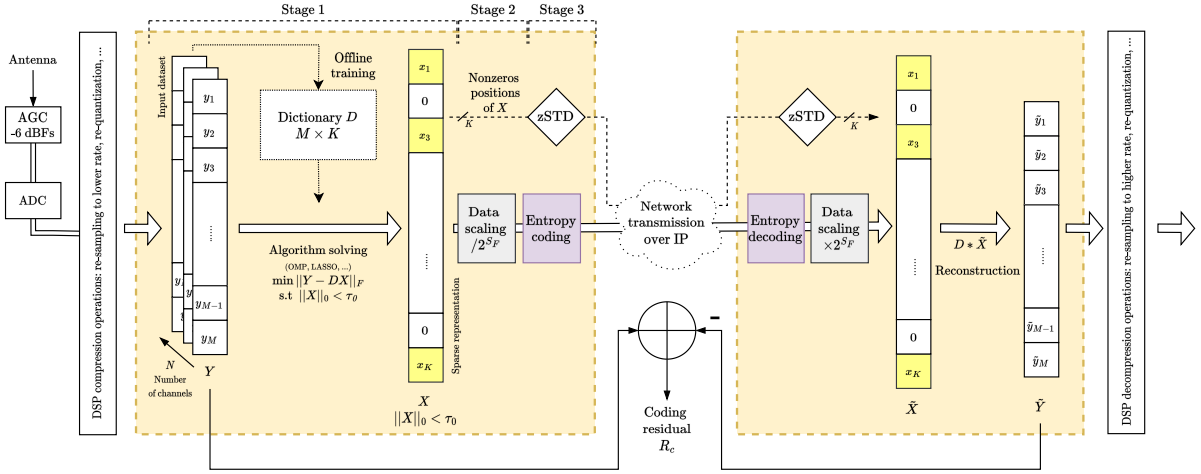



**Figure 6.8:** Proposed comprehensive compression and decompression chain for signal compression using sparse coding

The compression chain which has been proposed and illustrated by figure 6.8 is made up of three stages:

1 · The first stage performs actual sparse coding by applying a dictionary $\boldsymbol{\mathcal{D}}_{\tau_0}$ to the input samples.

2 · The second stage takes only nonzero values and applies a uniform quantization by dividing them by a scaling factor $\mathcal{S}_F$ (see below).

3 · The third stage applies an entropy coder to the remaining samples to perform actual compression.

The decision to apply a uniform scaling factor post-sparse coding rather than prior to it has been driven by various simulations conducted. Observations showed that, given the same factor, **the quantization noise introduced after the coding step was less than the noise introduced when scaling was performed before coding**. Furthermore, the **introduction of this factor essentially makes the compressor independent of the initial quantization of the signal**.

Finally, the compression steps are as follows:

1 · Based on the desired level of degradation and the target compression ratio, we select a pair of parameters $(\tau_0, \mathcal{S}_F)$. Given the infinite number of pairs for a particular degradation, it is necessary to identify the pair that, for a fixed degradation, maximizes the compression ratio. This involves examining all potential pairs and choosing the most optimal one. The preliminary investigation step, illustrated in Figure 6.10a, is carried out offline before compression occurs.

2 · The samples are divided into groups of $M$ adjacent samples, and a sparse coding operation is performed on them using a dictionary trained offline $\boldsymbol{\mathcal{D}}_{\tau_0}$.

3 · The sparse representation is cleansed of zeros, and **values that are not zero are processed through an entropy coder** –either Huffman or Rice–. We choose to include *Golomb-Rice* coder [38], which has demonstrated effective performance with sparse data [128], allowing for effortless adaptation to the diverse signals we come across. The positions of both zero and nonzero values are relayed via a $K$ bit header.

4 · To reconstruct the sparse vector, nonzero value positions are also sent over the network, but

are first packed and compressed using a dictionary algorithm according to Section 6.3.3.2. We chose *zSTD* (zStandard) [106] because of its trade-off between speed and performance.

Once reconstructed, the data are passed through the entire demodulation process, including the interleaver and Forward Error Correction (FEC) processes. These two processes are actually the most important part of the presented method. The interleaver will spread the compression error across the entire signal, while the FEC will ensure that the BER will reach the target value even when a strong degradation is applied.

## 6.5 Simulations

In order to evaluate the performance of this compressor, extensive simulations have been conducted to explore the interaction between the sparsity level $\tau_0$ and the scaling factor $\mathcal{S}_F$. **Our goal is to identify the optimal tuple that maximizes the compression ratio while adhering to a specified degradation threshold**. The complexity of these simulations arises from the need to meticulously evaluate each pair of parameters to discern the most effective combination under varying conditions.

Due to the rigorous computations and prolonged duration required for each simulation, the primary emphasis has been placed on an 8-PSK baseband signal that is processed through a 0.25 RRC filter. After the signal is generated, it is sampled by an Analog-to-Digital Converter (ADC), which paved the way for the subsequent implementation of the compression chain, as shown in Figure 6.8. The key parameters used in these simulations are described in Table 6.1, which facilitates a thorough and methodical investigation.

**Table 6.1:** Simulation parameters

| | |
|---|---|
| **AGC target** | -6 dBFs |
| **Interleaver size** | 324,000 |
| **FEC** | LDPC 5/6 (DVB-S2 mod15) |
| **BER goal** | $10^{-6}$ |
| **Quantization** | 8 bits |
| **Oversampling** | 2 samples per symbol |
| **M (sparsity parameter)** | 120 |
| **K (sparsity parameter)** | 120 |
| **N (sparsity parameter)** | 44,000 |

The parameters are based on typical settings and constraints for DVB-S2 signals. The AGC target of -6 dBFs accommodates the typical noise range for DVB-S2. A 324,000-size interleaver covers five DVB-S2 frames, providing a safety margin for uncertainty in data segmentation for sparse coding. The 5/6 LDPC FEC under modcod 15 is typical for moderate SNR regions. An 8-bit quantization is standard, and 2 samples per symbol for oversampling align with common practice. $M$ is set at 120, balancing the complexity of the algorithm and the effectiveness of compression against degradation, particularly since the efficient compression zones for $\tau_0$ are narrow (based on Figures 6.10 it can be observed that the effective range of values for $\tau_0$ is approximately between 20 and 60). $K$ equals $M$ since the use of overcomplete dictionaries did not show any improvement in the final results, simplifying our approach.

The dictionary for each value of $\tau_0$ is learned *offline* before simulation on a noiseless signal of the same kind according to Equation 6.1 using Mairal's ODL algorithm [113]. The actual implementation of the OMP used is the Order Magnitude Matching Pursuit (ORMP) algorithm [80], which has slightly better results than the classical OMP implementation [66]. All these algorithms implementations have been taken from the INRIA SPAMS library [27] and its revised version from Samuel St-Jean [129], optimized for the latest versions of Python.

## 6.6 Results & discussions

### 6.6.1 Algorithms based on $\ell_0$ norm

We performed our compression chain using the OMP algorithm with both Huffman and Golomb-Rice coders. Figure 6.9 presents the results obtained after simulation.

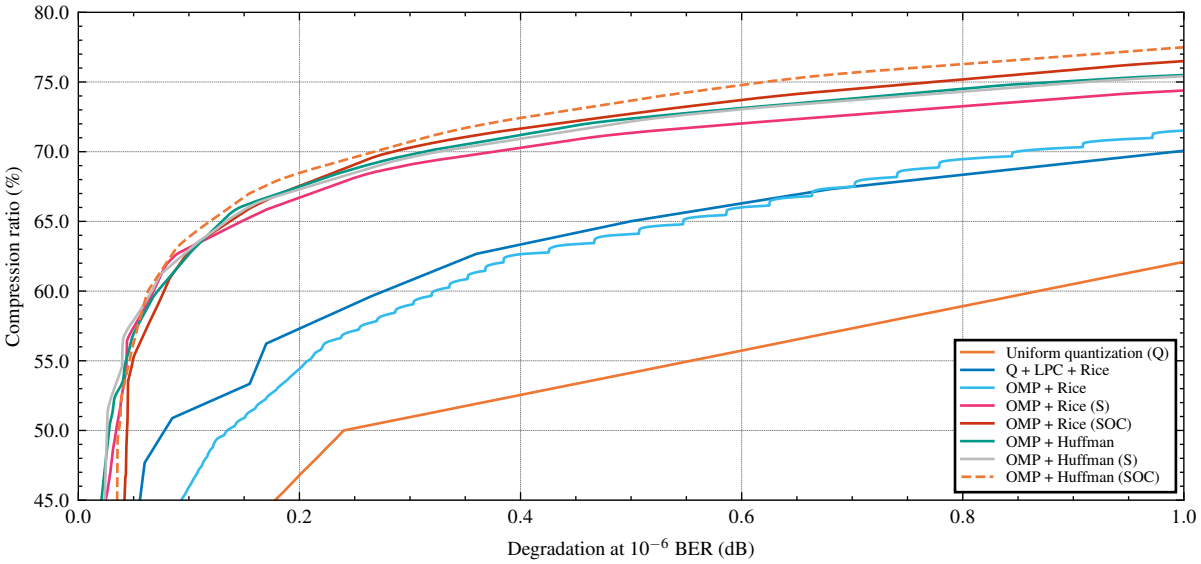
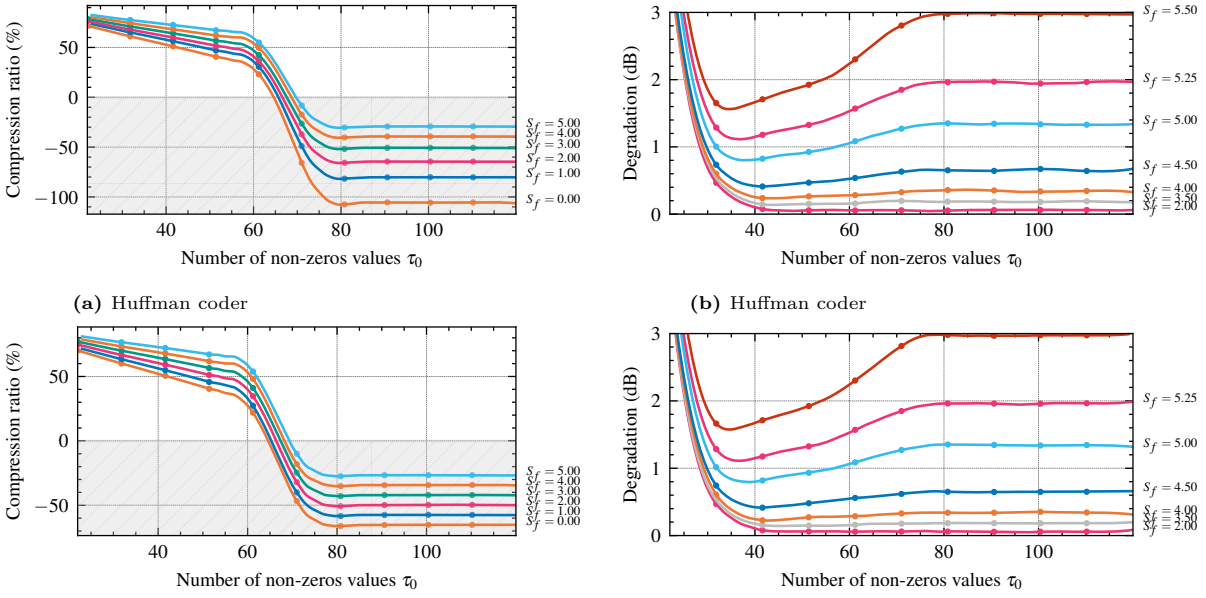



**Figure 6.9:** Expected compression ratio for each degradation. Simulations were performed for an 8-PSK signal with a 5/6 LDPC FEC.

To obtain these results, we must reduce our number of dimensions. Since we are working with two parameters, we must reduce it to a list of tuples of parameters. First, we determine the BER curve for all values of $\tau_0$ and $\mathcal{S}_F$. Determining the BER allows us to calculate the degradation $\Delta$, and $\tau_0$ gives us the associated compression ratio. This leads to the curves presented in Figure 6.10.

Next, we perform data interpolation to generate a scheme of values that covers the full spectrum of parameters. This allows us to plot the equi-degradation lines –symbolizing all parameter tuples that result in identical BER degradation–, which serve to identify the optimal parameter subspace, i.e., the subset that optimizes the compression ratio for each level of degradation. This process is depicted in Figure 6.11, where the optimal subspace is highlighted in orange. The final projection in this subspace gives us Figure 6.9



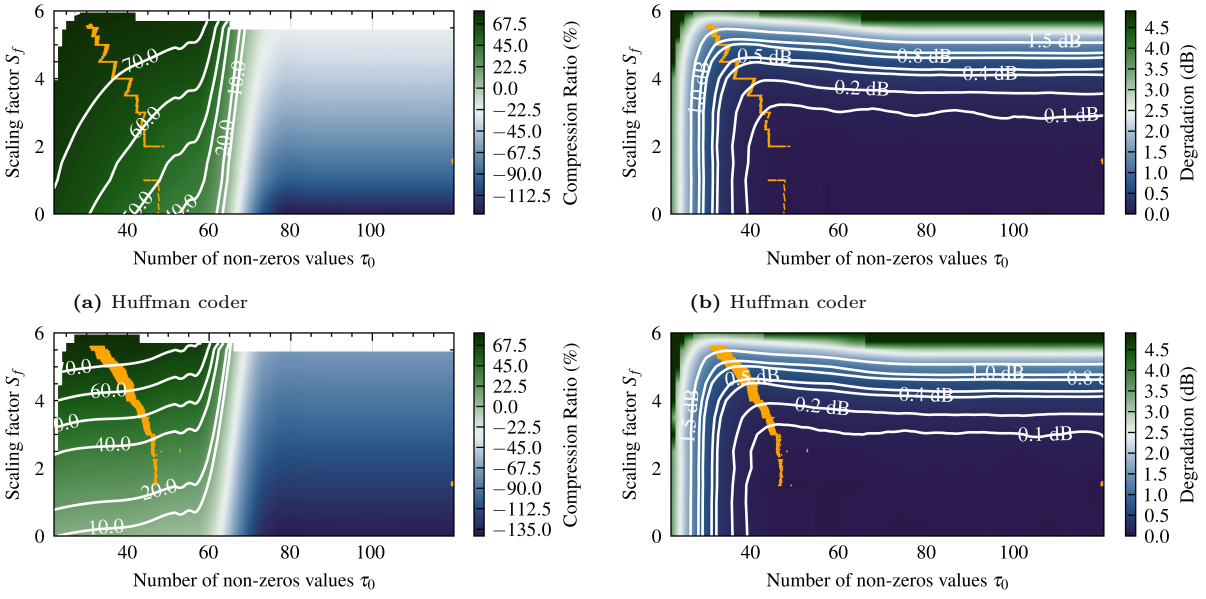

**(a)** Huffman coder

**(b)** Huffman coder

**(c)** Rice coder

**(d)** Rice coder

**Figure 6.10:** Effect of parameters on compression ratio & degradation for a 8-PSK signal using **SOC** optimization method




**(a)** Huffman coder

**(b)** Huffman coder

**(c)** Rice coder

**(d)** Rice coder

**Figure 6.11:** Determination of optimal subspace of parameters on compression ratio & degradation for a 8-PSK signal using **O** optimization method

### 6.6.2 Alternatives & exotic algorithms

#### 6.6.2.1 The ridge regression

**Presentation** Ridge regression, also known as *Tikhonov regularization*, is an alternative to traditional sparse coding techniques. Instead of minimizing the $\ell_0$ or $\ell_1$ norms, Ridge regression minimizes the $\ell_2$ norm. Given a signal $\boldsymbol{y} \in \mathbb{R}^M$ and a dictionary $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{M \times K}$, ridge regression seeks to find a coefficient matrix $\boldsymbol{x} \in \mathbb{R}^K$ such that the following objective function is minimized:

$$\min_{\boldsymbol{x}} \frac{1}{2}||\boldsymbol{y} - \boldsymbol{\mathcal{D}}\boldsymbol{x}||_2^2 + \frac{\lambda_2}{2}||\boldsymbol{x}||_2^2, \tag{6.6}$$

where $\lambda$ is a regularization parameter that controls the trade-off between the reconstruction error and the regularization term.

From a statistical perspective, the regularization $\ell_2$ imposes a penalty on the magnitude of the coefficients in $\boldsymbol{x}$, leading to smaller and more evenly distributed values. Unlike $\ell_0$ and $\ell_1$ norms, which promote sparsity by encouraging many coefficients to be exactly zero, the $\ell_2$ norm encourages a reduction in the overall magnitude of the coefficients. This results in a decrease in the energy of the representation and, in a specific way, in their entropy, as the values are more concentrated around zero without being strictly sparse.

**One of the main advantages of ridge regression is its simplicity in terms of computational complexity**. Unlike the $\ell_0$ and $\ell_1$ minimization problems, which are generally nonconvex and convex optimization problems, respectively, the ridge regression problem is strictly convex. This convexity ensures a unique solution that can be efficiently computed using an analytical expression. The solution to the ridge regression problem is given by:

$$\boldsymbol{x} = (\boldsymbol{\mathcal{D}}^\top \boldsymbol{\mathcal{D}} + \lambda_2 \mathbf{I})^{-1} \boldsymbol{\mathcal{D}}^\top \boldsymbol{y}, \tag{6.7}$$

where $\mathbf{I}$ is the identity matrix. This closed-form solution allows for rapid computation and straightforward implementation, making ridge regression a practical choice for scenarios where computational efficiency and reduced entropy are prioritized over strict sparsity.

**Usage for compression ?** We applied ridge regression in our experiments by varying the value of $K$ to investigate if reducing the number of coefficients could improve compression performance. Specifically, we tried to identify the optimal elements $\boldsymbol{x}$ with fewer coefficients than the data dimension $\boldsymbol{y}$. However, this approach proved to be ineffective. The results showed that the method was unstable, leading to significant variability in the quality of the reconstructed signals. Furthermore, the compression gain achieved through this technique was insufficient to justify its use –only 1 to 2% more than pure quantification–. Consequently, ridge regression was considered suboptimal for data recovery in our context. This finding led us to focus on the $\ell_0$ and $\ell_1$ norms, which are more suitable to achieve the desired balance between compression efficiency and signal reconstruction quality.

#### 6.6.2.2 About compressed sensing

In the state-of-the-art section, we introduced another widely-used data compression technique known as Compressive Sensing, which leverages the statistical properties of the data vector/matrix to reconstruct data from a lower-dimensional space.

Compressed sensing and sparse coding are concepts that are intimately linked in signal processing, involving the reconstruction of signals from limited information. Compressed sensing takes advantage of the sparsity of a signal in some domain, **allowing the signal to be reconstructed from far fewer samples than traditional methods would require**. This process hinges on the principle that many natural signals, when transformed into an appropriate basis, have a sparse representation. Sparse coding, on the other hand, focuses on finding such sparse representations of the data. It involves developing algorithms that, given a signal, can identify a sparse set of basis elements from a dictionary to represent that signal efficiently. The synergy between these two areas lies in their shared dependence on sparsity: **compressed sensing efficiently acquires signals assuming that they can be sparsely represented, while sparse coding provides the means to find and utilize these sparse representations for signal reconstruction and analysis.**

If we consider a data vector $\boldsymbol{y} \in \mathbb{R}^M$, and a *sensing matrix* (or *measurement matrix*) $\boldsymbol{\Phi} \in \mathbb{R}^{L \times M}$, the main step of compressed sensing process is to compute and transmit

$$\boldsymbol{y_s} = \boldsymbol{\Phi y} \tag{6.8}$$

Where $\boldsymbol{y_s} \in \mathbb{R}^L$ is called the *sensed vector* of data. When using compressed sensing, the actual compression ratio is determined by the number of measurements $L$, which corresponds to the actual number of data transmitted over the network (data of $\boldsymbol{y_s}$). Then, two conditions must be satisfied to actually compress the data.

- $L$ should be sufficiently low to achieve an acceptable compression ratio. When employing such advanced methods, we anticipate achieving *at minimum* the compression ratio that simpler techniques, such as resampling, provide.
- The amplitude of the coefficients in $\boldsymbol{y_s}$ must be comparable in size with the coefficients of $\boldsymbol{y}$ [130]. If we reduce the number of data by a factor 2, but the new data needs 32 bits to be encoded, we actually did not compress anything! Since coefficients of $\boldsymbol{y_s}$ are actually the results of multiple multiplication and addition, one direct consequence is that the norm of each **lines** of matrix $\boldsymbol{\Phi}$ must be equal to 1, and that the distribution of coefficients in $\boldsymbol{\Phi}$ will then follow a normal distribution with a standard deviation similar to that of data in $\boldsymbol{y}$, but without being bounded.

Since we will likely pass $\boldsymbol{y_s}$ through an entropy coder before actually sensing the data over the network, the total compression ratio $\mathcal{C}\{\boldsymbol{y}, \boldsymbol{y_s}\}$ can then be expressed as follows:

$$\mathcal{C}\{\boldsymbol{y}, \boldsymbol{y_s}\} = 1 - \frac{L * \text{Entropy of } \boldsymbol{y_s}}{M * N_q} \tag{6.9}$$

Let us now consider a proper example. An 8-PSK signal with a shaping filter of $\text{RRC}_{0.25}$ as used in the DVB-S2 standard. Suppose that the original signal stored in $\boldsymbol{y}$ is sampled at $\mathcal{F} = 2$ samples per symbol and $\boldsymbol{y} \in \mathbb{R}^{256}$. In order to use compressed sensing on this signal –the complete compression chain is available in Annex C–, we need three steps:

$\boxed{1 \cdot}$ **Sparse coding step:** First, we need to find a dictionary $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{K \times M}$ and a sparse vector $\boldsymbol{x} \in \mathbb{R}^K$ containing $k$ nonzeros elements ($||\boldsymbol{x}||_0 = k$), such as $\boldsymbol{y} = \boldsymbol{\mathcal{D}x}$. We are going to suppose

here that $K = M = 256$. As presented in the previous section, this sparse coding step will introduce a *coding noise* and deteriorate the BER.

2 · **Compressed sensing step:** Then, we need to find a *sensing matrix* $\boldsymbol{\Phi}$, multiple solutions exist: we can use a Gaussian random matrix [68], [131] which will do the job most of the time, or use our knowledge of $\boldsymbol{\mathcal{D}}$ to construct $\boldsymbol{\Phi}$ using for example the Elad algorithm [123], [132]. We then compute $\boldsymbol{y_s} = \boldsymbol{\Phi}\boldsymbol{\mathcal{D}}\boldsymbol{x} = \boldsymbol{\Phi}\boldsymbol{y}$.

3 · **Reconstruction step:** The goal is now to reconstruct $\boldsymbol{y}$ from $\boldsymbol{y_s}$ using only our knowledge of $\boldsymbol{\Phi}$, $\boldsymbol{\mathcal{D}}$ and $k$. This can be done with generalized algorithms like OMP, or more specialized algorithms like CoSaMP. One can also use $\ell_1$ relaxation algorithms –See [114] for more details–. This step introduce a new reconstruction error, since the probability of reconstruction is lower than 100%, some vectors of $\boldsymbol{y_s}$ can be badly reconstructed or not reconstructed at all ! If we suppose that an interleaver is used prior to transmission, **this phenomenon will result in a *recovery noise* spread across the received signal.**

> In compressed sensing theory, steps 1 and 2 are actually merged into one step, when we directly calculate $\boldsymbol{y_s} = \boldsymbol{\Phi}\boldsymbol{y}$, but the actual noise introduced at the end is the same, since the third step suppose to find $\boldsymbol{x}$ in a specific basis determined by a dictionary $\boldsymbol{\mathcal{D}}$.

Now, here is the catch: **What should be the value of $k$ to obtain a compression ratio of 30%, which is a ratio that we can easily have with classical resampling?**

Since $M = K = 256$, we need to have $L \leq 180$. According to [133, Figure 1], which provides a complete analysis of the compressed sensing application when the OMP algorithm is used, the value of $k$ required to achieve a data reconstruction success rate of over 90%, thereby minimizing *reconstruction noise*, **is less than 20**. But, as we saw previously, such a value of $k$ is likely to destroy our signal due to the *coding noise*.

> Under these conditions, **it seems quite impossible to have a low *reconstruction noise* and a low *coding noise* at the same time**, minimizing one of them increasing the other. Under this observation, the usage of compressed sensing for signal compression seems really difficult to achieve.

However, this does not mean that compressed sensing is impossible to achieve, but this will be at the cost of high resource usage. First, changing the OMP algorithm (which is already quite resource-consuming) to more accurate algorithms [85] will likely increase resource usage. For example, the usage of the *SL0* algorithm [119], [134], one of the best reconstruction algorithms so far, takes so much time to process –and resources to be implemented– that it is not usable for real-time applications such as signal processing.

Moreover, the research of a more efficient *sensing matrix* $\boldsymbol{\Phi}$ is difficult [135] as it must follow many mathematical properties to be valid [68], [78].

**Finally, we should understand what *exact reconstruction* means in compressed sensing**. In fact, many articles **actually suppose that signals are recovered when the support of the sparse vector $\boldsymbol{x}$ is recovered**, which is not enough for us since **we also need to reconstruct the amplitude of the signal**. This question has been the subject of numerous debates in the scientific community [136], and should be clearly stated.

### 6.6.3 A word about hardware implementation

Implementing sparse coding algorithms on hardware, particularly on Field-Programmable Gate Array (FPGA), presents unique challenges and advantages due to the inherent characteristics of these devices, such as parallel processing capabilities and reconfigurability. This section discusses various strategies and considerations derived from recent FPGA implementations of algorithms relevant to sparse coding and dictionary learning.

One of the main challenges in implementing these algorithms in FPGAs is **managing limited resources such as memory and computational units**. Techniques such as resource sharing, pipelining, and optimization of arithmetic units are critical to enhancing the performance and efficiency of FPGA implementations. Proper management of these resources ensures that the implementations can meet the real-time processing requirements of modern communication systems.

**Efficient implementation of Singular Value Decomposition (SVD)** FPGA-based implementations of SVD, as described by Shiri and Khosroshahi [137] and Tian et al. [138], highlight the use of iterative techniques and the COordinate Rotation DIgital Computer (CORDIC) algorithm to enhance the rotation operations necessary for SVD. The CORDIC algorithm is particularly advantageous for hardware implementations due to its straightforwardness and efficiency in executing vector rotations using only shift and add operations. This makes it highly suitable for FPGA implementations where resource utilization is crucial. These implementations facilitate the rapid and efficient execution of *online* dictionary learning algorithms such as K-SVD.

**Orthogonal Matching Pursuit (OMP) for sparse representation** Morales-Pérez et al. [139] provide an example of the implementation of the orthogonal matching pursuit algorithm in FPGA, highlighting its applicability in sparse signal recovery tasks. Their design leverages the parallel processing capabilities of FPGAs to speed up the atom selection phase of OMP, significantly reducing the time required to match dictionary atoms with the signal. The implementation details include the use of dedicated hardware blocks for inner product calculations, which are critical to determine the correlation between the signal and dictionary atoms.

Moreover, the work by Li et al. [140] on FPGA-based implementations for improved OMP algorithms points to modifications tailored to specific application needs, such as **adaptive thresholding** and **dynamic dictionary updates**, which are essential for real-time signal processing applications. These adaptations help maintain **high accuracy while coping with the constraints of hardware resources and processing time**.

In conclusion, the hardware implementation of sparse coding algorithms on FPGAs offers significant advantages in terms of speed and efficiency. However, it requires careful consideration of resource allocation, algorithmic adaptations, and hardware-specific optimizations. The reviewed studies provide valuable information on the approaches that can be taken to address these challenges, making FPGA a viable platform for deploying advanced signal processing techniques in real-world applications.

## 6.7 Conclusion

This chapter has presented an in-depth exploration of sparse coding and sparse dictionary learning techniques, emphasizing their applications and implications in the context of signal compression for satellite communications. Our investigation spanned from traditional sparse coding methods such as Orthogonal Matching Pursuit (OMP) to innovative approaches and the implementation of advanced algorithms on hardware platforms like FPGAs. The results demonstrated the efficacy of these methods in achieving significant data compression while maintaining acceptable levels of signal degradation.

One of the primary successes of our work lies in the efficient compression of satellite communication signals using sparse coding. By employing OMP in combination with Huffman (or Rice) coders, we achieved a notable enhancement in compression ratios, **reaching up to** 8% **higher than classical methods for equivalent degradation levels**. This improvement underscores the potential of sparse coding to significantly reduce the volume of data transmitted without compromising signal integrity. Moreover, the strategic selection of parameters, such as the number of nonzero coefficients ($\tau_0$) and the scaling factor ($\mathcal{S}_F$), was crucial in optimizing the trade-off between compression efficiency and signal quality. Despite these successes, our exploration also revealed several limitations and challenges inherent to sparse coding techniques. The computational complexity of solving the sparse representation problem remains a significant hurdle. While algorithms like OMP provide effective approximations, their performance can be hindered by the NP-hard nature of the underlying problem. Additionally, methods like ridge regression, which we investigated as potential alternatives, proved to be unstable and less effective for our specific application, highlighting the need for careful selection and adaptation of algorithms based on the signal characteristics and compression requirements.

Our research contributes to the state-of-the-art by introducing an optimized compression chain that leverages sparse coding principles. The proposed method, which includes the innovative SOC optimization for transmitting nonzero positions, demonstrated superior performance by minimizing the data overhead associated with sparse representations. This novel approach, combined with our strategic parameter selection and the implementation of efficient entropy coding techniques, sets a new benchmark for signal compression in satellite communications.

Furthermore, the exploration of hardware implementations, particularly the use of FPGAs to process sparse coding algorithms, highlights the practical feasibility and potential for real-time applications. The studies reviewed on FPGA implementations provide a blueprint for optimizing resource allocation and enhancing computational efficiency, paving the way for the deployment of these advanced signal processing techniques in real-world scenarios.

In conclusion, while the journey through sparse coding and dictionary learning techniques has revealed both potentials and limitations, our contributions have significantly advanced the field. The proposed methods not only offer improved compression ratios and signal quality, but also provide a framework for future research and practical applications in satellite communications and beyond. By addressing the challenges and leveraging the strengths of sparse coding, we have laid the groundwork for more efficient and effective data compression strategies.

Part III

# Compression by partial matching & predictive coding

---

This part explores sophisticated predictive coding methods and their use in data compression within the field of digital communications. Chapter 7 focuses on Linear Predictive Coding (LPC) and its implementation, evaluating its efficiency in compressing baseband signals, while Chapter 8 explores nonlinear methods, particularly using neural networks, to enhance compression performance for complex real-world signals. The goal of these chapters is to enhance compression efficiency by utilizing both linear and nonlinear prediction techniques, which are tailored to the complex characteristics of communication signals.

Chapter 7

# Predictive coding & PPM

***To compress better than yesterday, but less than tomorrow***


7.1 Related work . . . . . **118**
7.2 Linear prediction . . . . . **119**
7.2.1 Mathematical model . . . . . 119
7.2.2 LPC based predictors . . . . . 120
7.3 Tests & usage . . . . . **124**
7.3.1 Usage of FLAC codec . . . . . 124
7.3.2 LPC vs FLAC . . . . . 126
7.4 Limitations . . . . . **127**
7.5 Conclusion . . . . . **129**


## 7.1 Related work

In the following chapter, we explore a particular compression technique referred to as *predictive compression*, also known as Prediction by Partial Matching (PPM). PPM stands out as an adaptive statistical data compression technique, primarily leveraging context modeling and predictive analytics to optimize compression efficacy. This method uses a series of previous samples from the uncompressed data stream, using these to **forecast subsequent samples**, and then to compress **the difference between the predicted samples and the original ones**. This difference, usually called *residual*, is designed to have statistical properties which are beneficial to entropy coding compression algorithms. This technique finds significant application in the domain of audio compression, with FLAC and SHORTEN [58] representing notable examples where PPM is used, generally employing **linear prediction methods** for this purpose [49].

**The application of FLAC-like algorithms in Centralized Radio Access Network (C-RAN) and virtualized architectures is relatively new** but has already shown significant improvements in data compression [10], [19], [21]. These algorithms offer a fast, reliable and lossless method for **blind waveform compression**, particularly useful when the **waveform characteristics are unknown**.

The considered PPM compression system (Figure 7.1) is based on a prediction of the signal to compress [141]. It offers a really interesting architecture based on a direct and fast compression step, followed by a slower and iterative decompression step.

## 7.2 Linear prediction

### 7.2.1 Mathematical model

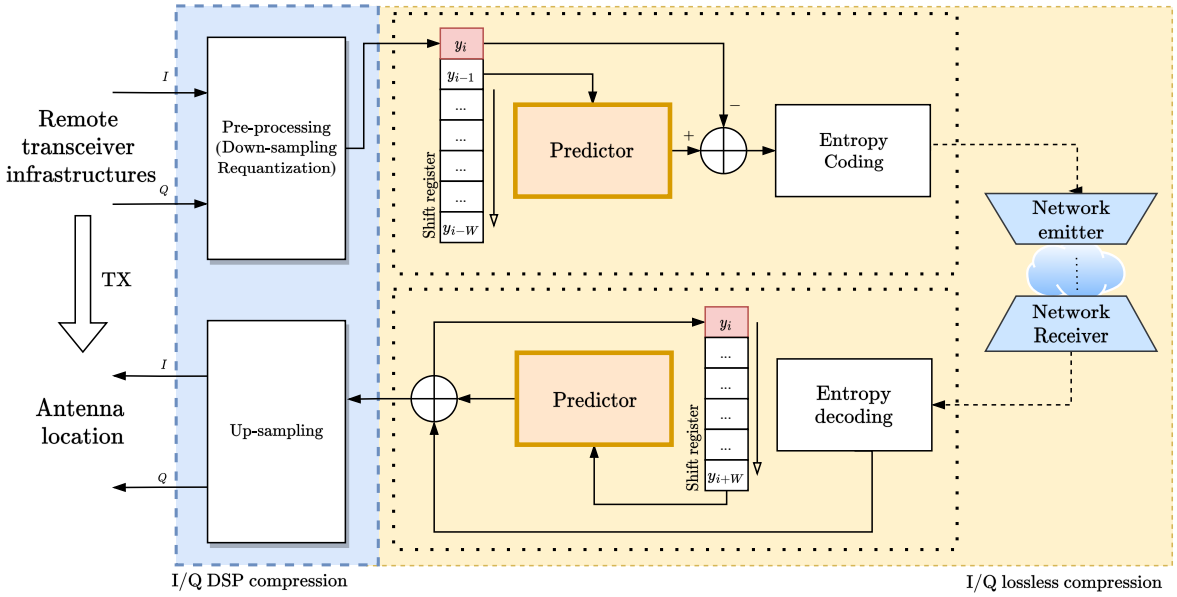



**Figure 7.1:** Classical architecture for compression/decompression schema from a remote modulation unit to an antenna using predictors. Entropy coding is usually performed by a Golomb-Rice encoder.

Consider a sampled signal $(y_k)_{k\in\mathbb{N}}$, the goal of a linear predictor is to find a linear relationship between past and current samples.

$$\forall k, \quad y_k = \underbrace{\sum_{i=1}^{\mathrm{W}} \alpha_i(k) y_{k-i}}_{\tilde{y}_k} + \varepsilon_k \tag{7.1}$$

$\boldsymbol{\alpha}_k = (\alpha_1(k), \ldots, \alpha_{\mathrm{W}}(k)) \in \mathbb{C}^{\mathrm{W}}$ represents the linear coefficients of the relation. In this case, the number of coefficients W is usually chosen when the amplitude of the coefficients becomes too low for their contribution to be significant. **Since we are actually compressing by sending the residual and not the signal**, the main difficulty of this prediction method is to find a set of $(\boldsymbol{\alpha}_k)_{k\in\mathbb{N}}$ that maximizes the global compression ratio, *i.e.,* minimizes $\varepsilon$ entropy. However, minimizing entropy directly sounds like a really complex issue.

To optimally choose the number of coefficients W, several theoretical criteria can be used, such as the *Akaike Information Criterion (AIC)*, the *Bayesian Information Criterion (BIC)* and the *Minimum Description Length (MDL)*.

- **Akaike Information Criterion (AIC) [142]**: AIC aims to balance the quality of the fit of the model with its simplicity, penalizing models with a large number of parameters.
- **Bayesian Information Criterion (BIC) [143]**: Similarly to AIC, BIC imposes a stronger penalty on complex models, favoring simpler models for a given dataset size.
- **Minimum Description Length (MDL) [144]**: Based on information theory, MDL seeks to minimize the total length required to describe the model and the associated data.

In practice, due to the small size of W and the fact that the value is stable –its value remains largely unchanged, regardless of modifications to the parameters or the waveform–, we can dynamically verify performance by testing multiple values of W. This approach is much more practical and faster, allowing us to empirically determine the optimal number of coefficients.

Depending on the signal's properties, various scenarios are possible. **The simplest case appears when the process is autoregressive**. An autoregressive process is a kind of stochastic process where $\forall i, k\ \alpha_i(k) = \alpha_i$ and $(\varepsilon)_{k\in\mathbb{N}}$ is white noise (not necessarily Gaussian). In this case, the $(\alpha_i)_{i\in[\![1,\mathrm{W}]\!]}$ coefficients are called the AR coefficients. **The study of AR (autoregressive) and MA (moving average) processes, generally referred to as the ARMA process, is a vast and well-studied area of parametric spectral analysis, which allows the development of complex linear estimators**.

In communication theory, while signals often do not strictly adhere to AR models, AR coefficient resolution methods can still be beneficial by leveraging the short-term predictability of the signals. This approach, assuming that the signal temporarily behaves like a AR process, forms the basis for Linear Prediction Coding (LPC). LPC is an integral part of numerous contemporary compression algorithms, both in audio and in 5G technologies [10], [21].

## 7.2.2 LPC based predictors

### 7.2.2.1 Presentation & mathematical background

Linear Prediction Coding (LPC) is a method used in signal processing to predict a signal's future samples using an **unique linear combination** of its past samples.

$$\forall k,\quad y_k = \underbrace{\sum_{i=1}^{\mathrm{W}} \alpha_i y_{k-i}}_{\tilde{y}_k} + \varepsilon_k \tag{7.2}$$

The strength of LPC resides in its proficiency at accurately modeling a signal's spectral features, thus establishing it as a favored approach for predictive tasks in compression systems. The primary challenge lies in determining the coefficients $\alpha$. Given a signal $y(t)$, represented by a sequence of samples $(y_k)_{k\in\mathbb{N}}$, the objective is to identify the set of coefficients $\boldsymbol{\alpha} \in \mathbb{C}^{\mathrm{W}}$ by minimizing the entropy of $\varepsilon$. A common approach is then to minimize the $\ell_2$ norm of $\varepsilon$ such as:

$$\boldsymbol{\alpha} = \underset{(\alpha_1,\cdots,\alpha_{\mathrm{W}})\in\mathbb{C}^{\mathrm{W}}}{\operatorname{argmin}}\ \mathbb{E}\left\{\left|y_k - \sum_{i=1}^{\mathrm{W}} \alpha_i y_{k-i}\right|^2\right\} \tag{7.3}$$

where $\mathbb{E}\{.\}$ denotes the mathematical expectation operator.

### 7.2.2.2 Coefficients determination

Equation (7.3) leads to a minimization problem **that can be solved using the least squares method**. The resolution process reveals the biased autocorrelation coefficients of $y$ in a Toeplitz matrix, which makes the system efficiently solvable using a Levinson-Durbin recursion [145], [146]. This is, for example, the method used in the actual implementation of LPC provided by MATLAB®. It is worth mentioning that a detailed explanation of the technique, along with a discussion on restrictions, edge effects, etc., can be found in the excellent lecture by Gilles Fleury [147].

**However, these coefficients can also be extracted from the calculated weights of a simple neural network that contains a dense hidden layer without activation function [148]. This is the method presented below.**

By definition, a linear neural network has no activation function (and thus no nonlinearity), the relation between the input and the output of the network follows a linear relation. Consider, for example, a neural network with an input layer that takes the previous W samples, a prediction / hidden layer that contains $q$ neuron and an output layer made of only one neuron that was used to get the prediction. Figure 7.2 presents such a network.

In this case, the relationship between inputs and output can be expressed as follows:

$$\tilde{y_k} = b_0 + \sum_{l=1}^{q} \omega_{l,1}^{[1]} \left( b_l + \sum_{i=1}^{\mathrm{W}} \omega_{i,l}^{[0]} y_{k-i} \right) \tag{7.4}$$

After rearranging the terms of the equation, we get :

$$\tilde{y_k} = b_0 + \sum_{l=1}^{q} \omega_{l,1}^{[1]} b_l + \sum_{i=1}^{\mathrm{W}} y_{k-i} \left( \sum_{l=1}^{q} \omega_{l,1}^{[1]} \omega_{i,l}^{[0]} \right) \tag{7.5}$$

We can then draw a parallel with an LPC/AR decomposition by assigning each AR coefficient as follows:

$$\begin{cases} b = b_0 + \sum_{l=1}^{q} \omega_{l,1}^{[1]} b_l \\ \alpha_i = \sum_{l=1}^{q} \omega_{l,1}^{[1]} \omega_{i,l}^{[0]}, \forall i \in [\![1, \mathrm{W}]\!] \end{cases} \tag{7.6}$$

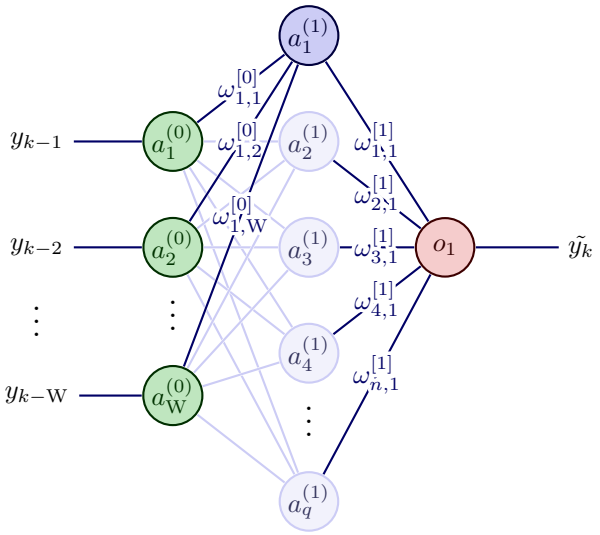



**Figure 7.2:** Linear 1-layer neural network without activation function

Which leads to:

$$\tilde{y_k} = b + \sum_{i=1}^{\mathrm{W}} \alpha_i y_{k-i} \tag{7.7}$$

We notice that if the neural network is correctly trained, then the combination of these weights and its biases theoretically lead to the LPC coefficients found thanks to the least squares method. Moreover, in the case of a signal with zero mean, we can expect to find $b = 0$. Considering that the weights are nonzero, this condition should lead to the following condition on the biases:

$$\forall l \in [\![1, q]\!], b_l = 0 \tag{7.8}$$

To verify these claims, we trained a single-layer linear network on a two-sample-per-symbol GMSK data set containing 50 millions samples. We chose $\mathrm{W} = 5$ and $q = 16$ and ran the training over 5 periods using a batch size of 512. The table below lists the weights of the trained network:

$$\omega^{[0]} = 10^{-3} \begin{bmatrix} 82 & 420 & -3 & 46 & 229 & 136 & -30 & -4 & 403 & 394 & -299 & -418 & 488 & -104 & -548 & -390 \\ -324 & 18 & -30 & -395 & 327 & -209 & -526 & 618 & 0 & 280 & 87 & -147 & 109 & 392 & -34 & 85 \\ 436 & 465 & 7 & 161 & 26 & -172 & 477 & 301 & -439 & -68 & 252 & -281 & 101 & 565 & -521 & -139 \\ -507 & 89 & -20 & -246 & 540 & -33 & -356 & 334 & -283 & -91 & -213 & 265 & 466 & -454 & 243 & -49 \\ 575 & 99 & 17 & 23 & -729 & -390 & -44 & -540 & 366 & 49 & 497 & 371 & 229 & 378 & 297 & 404 \end{bmatrix}^T \tag{7.9}$$

$$\omega_{,}^{[1]} = 10^{-3}\left[\begin{array}{cccccccccccccccc} 800 & 561 & -35 & 42 & -762 & -531 & 369 & -549 & 93 & 309 & 459 & -266 & -348 & 494 & -352 & 16 \end{array}\right] \tag{7.10}$$

Which leads to

$$\tilde{\alpha} = \begin{bmatrix} 1.746 & -1.770 & 1.252 & -0.602 & 0.147 \end{bmatrix} \tag{7.11}$$

Actual LPC coefficients calculated over the same set of samples using least-square method are:

$$\alpha = \begin{bmatrix} 1.743 & -1.770 & 1.248 & -0.594 & 0.151 \end{bmatrix} \tag{7.12}$$

A neural network, when used with no activation function, is strictly identical to an LPC predictor.

#### 7.2.2.3 Effective lossless compression using LPC

**Compression stage** Knowing the coefficients, we can then calculate the values of the residual sequence $(\varepsilon_k)_{k\in\mathbb{N}}$. As mentioned above, the residual is the actual value passed to the entropy encoder and then transmitted over the network. Depending on the implementation and the characteristics of the signals (such as stationarity or ergodicity), the coefficients $(\alpha_k)_{k\in\mathbb{N}}$ can be calculated once and stored permanently on the compressor / decompressor side if they do not change over time. They can also be recalculated and transmitted with each data packet, which reduces the global compression ratio, but is still acceptable when the number of coefficients (W parameter) is low.

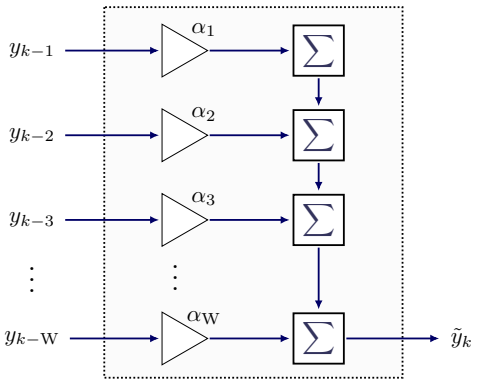



**Figure 7.3:** Predictor (compressor) using LPC coefficients. The implementation of this kind of predictor actually has the architecture of a FIR filter

In this study, the entropy encoder used is a classical Golomb-Rice encoder [38], which is a particular case of the Golomb encoder. As seen before, this encoder offers really good results with residuals that have an exponential or seminormal statistical distribution [149]. Moreover, its implementation on both software and hardware platforms has also been studied and is actually used on multiple hardware implementations of audio codecs using LPC (including FLAC).

To compress data using Rice coding, the residuals are first transformed to positive integers. This is achieved by mapping each value $\varepsilon_k$ of the residual $(\varepsilon_k)_{k\in\mathbb{N}}$ –which is the difference between the LPC prediction and the actual value of the signal– to $2\varepsilon_k$ if $\varepsilon_k$ is not negative and to $1-2\varepsilon_k$ if $r_k$ is negative. After this transformation, the Rice parameter $k$ is determined. This parameter can be found either by brute force, –testing multiple values and selecting the one that results in the smallest encoded size–, or analytically if the probability distribution of the residuals is known. Appendix A provides some details about the analytical approach.

**Decompression stage** Note that in the decompression stage, the data can be recovered step by step by using previously reconstructed samples to reconstruct the current one. However, this operation requires transmitting the first W samples to the decompressor –or define them to 0–.

$$\forall k \in \mathbb{N}, y_k = \underbrace{\varepsilon_k}_{\substack{\text{Transmitted from}\\ \text{the compressor}}} + \sum_{i=1}^{\mathrm{W}} \underbrace{\alpha_i y_{k-i}}_{\substack{\text{Known by}\\ \text{the decompressor}}} \tag{7.13}$$

We can observe that the real count of multiplications, following the determination of the coefficients, does not exceed W. However, if we had directly used the original neural network, the multiplication count would have increased significantly for the exact same result –According to Equation 7.6, each coefficient necessitate $q$ multiplications. So a total of $q \times \mathrm{W}$ multiplications–. **Understanding and examination of the underlying process can result in substantial enhancements in the final implementation**.

## 7.3 Tests & usage

### 7.3.1 Usage of FLAC codec

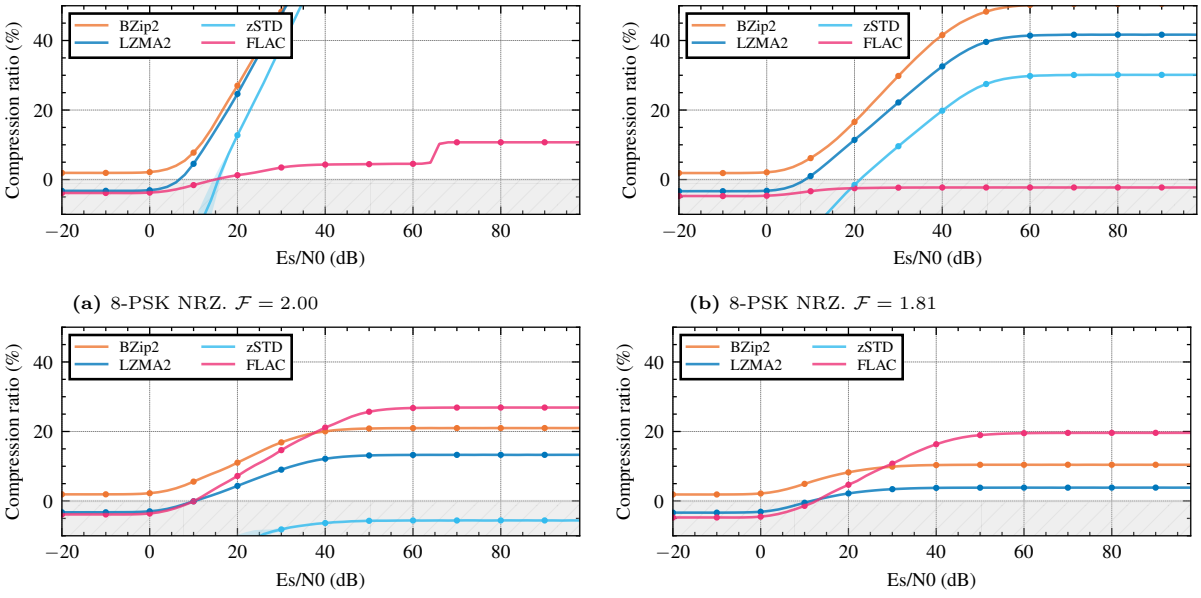
A simple and quick method to determine whether LPC-based methods are effective in compressing baseband signals is to apply the FLAC codec directly to our data. FLAC employs LPC in the background and adds optimizations such as block separation to partition the audio signal into temporal frames that share the same LPC coefficients. In order to maximize the compression ratio and have a good overview of the FLAC capabilities, we can use the following command parameters.

```
flac.exe -l 16 --best --exhaustive-model-search \
    --qlp-coeff-precision-search --lax
```

Parameters used in this command are chosen for optimal performance:

- `-l 16`: Specifies the LPC order, with 16 being a good trade-off to capture more detail.
- `-best`: Ensures the highest compression quality by using the best compression algorithm.
- `-exhaustive-model-search`: Conducts a comprehensive search for the best model parameters.
- `-qlp-coeff-precision-search`: Searches for the best precision for quantized linear prediction coefficients.
- `-lax`: Allows settings that might not be compatible with all FLAC implementations but can improve performance while relaxing the conditions on input parameters.

Since FLAC is designed for audio formats, a simple way to use is to save our samples in a *.wav* file, using each channel in stereo mode to map each channel of our samples (I & Q). This solution is the one used in the original article that introduced this method [19].




**(a)** 8-PSK NRZ. $\mathcal{F} = 2.00$

**(b)** 8-PSK NRZ. $\mathcal{F} = 1.81$

**(c)** 8-PSK $\text{RRC}_{0.25}$. $\mathcal{F} = 2.00$

**(d)** 8-PSK $\text{RRC}_{0.25}$. $\mathcal{F} = 1.81$

**Figure 7.4:** Evolution of compression ratio $\mathcal{C}$ for various algorithms, waveforms, and SNR conditions

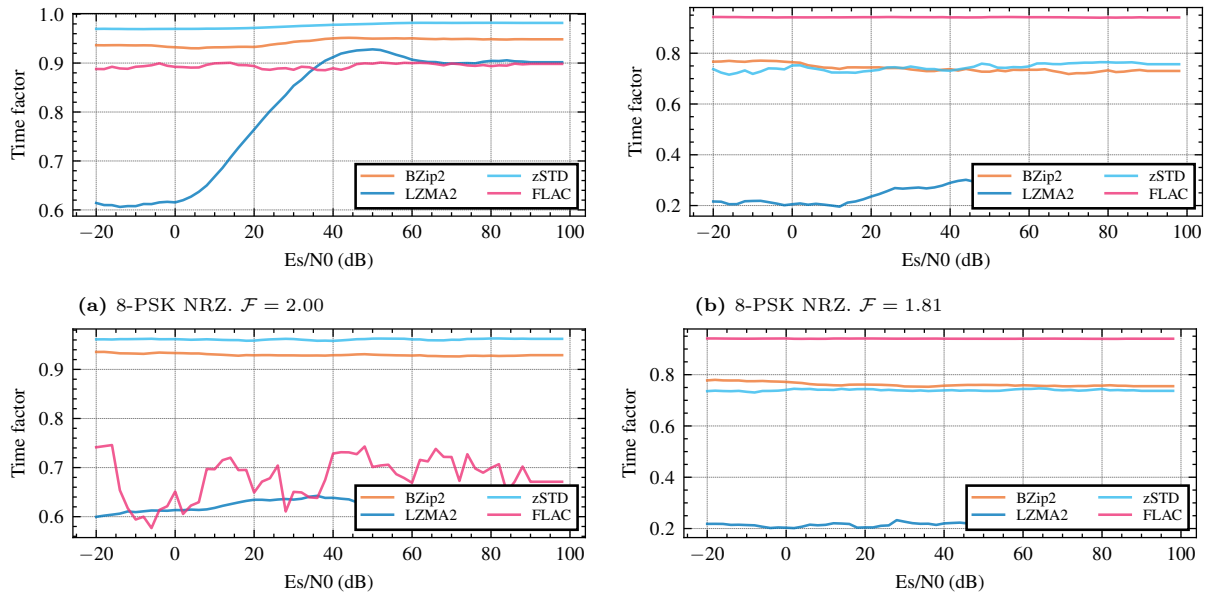
Figure 7.4 presents the results of the compression ratio $\mathcal{C}$ for each modulation compared to zSTD, BZIP2, and LZMA for two different sampling rates. The results indicate:

- First, FLAC is not effective for NRZ-shaped signals. This was attented since there is no predictable information in the signal waveform.
- For RRC signals, FLAC performs better than traditional dictionary-based compressors such as BZIP2 or zSTD under high SNR conditions. However, its performance rapidly degrades with lower SNR.

One could conclude here that the information that makes LPC compression work is linked to Inter-Symbol Interference (ISI). The higher the ISI, the better the prediction.

The most notable observation from these curves is that **LPC is not highly sensitive to resampling**. If the number of samples per symbol is noninteger, it does not significantly impact the result, making this solution particularly suitable for environments where controlling the samples per symbol is challenging. **Another consequence is that LPC can be used after a down-sampling block as originally suggested by [17]**

Another important metric is the time factor $\mathcal{T}_F$. As shown in Figure 7.5, **the FLAC decompression step is much more time consuming and resource intensive than compression**. This discrepancy is due to the fact that, while the compression step can be easily and quickly parallelized, the decompression step is iterative and cannot be easily parallelized.




**(a)** 8-PSK NRZ. $\mathcal{F} = 2.00$

**(b)** 8-PSK NRZ. $\mathcal{F} = 1.81$

**(c)** 8-PSK $\text{RRC}_{0.25}$. $\mathcal{F} = 2.00$

**(d)** 8-PSK $\text{RRC}_{0.25}$. $\mathcal{F} = 1.81$

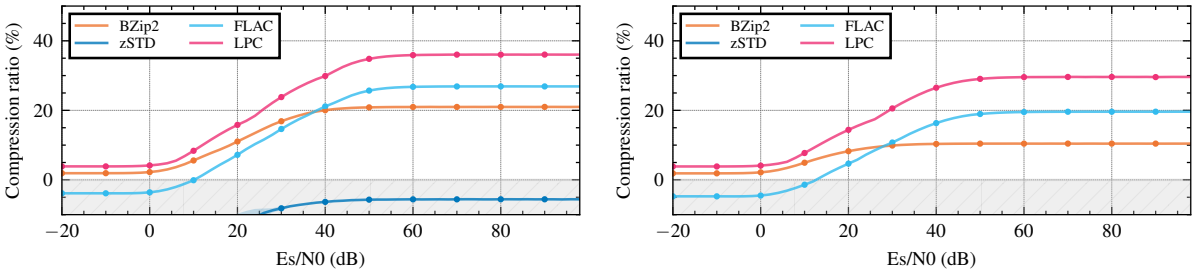
**Figure 7.5:** Evolution of time factor $\mathcal{T}_F$ for various algorithms, waveforms, and SNR conditions

### 7.3.2 LPC vs FLAC

Although FLAC provides a quick method to evaluate the potential of LPC compression for a specific waveform, the actual compression ratios achieved with it are suboptimal. This can be attributed to several issues inherent to FLAC's design, which is primarily tailored for audio data.

- First, FLAC divides the original signals into blocks and calculates LPC coefficients for each block. This not only increases computation time but also makes FLAC highly sensitive to noise, as it cannot leverage the averaging effect of using an high quantity of samples to find the best LPC coefficients possible. In our case, since all samples share the same temporal characteristics, **we can calculate the optimal LPC coefficients offline and apply the same coefficients to both noisy and nonnoisy data.**
- Secondly, FLAC includes numerous headers and metadata necessary for audio reconstruction. By removing these headers, we can significantly enhance the compression ratio.
- Finally, FLAC incorporates the first W samples of each block. In our implementation, we set the first W samples to zero, and since we do not use blocks, there is no need to store and transmit uncompressed samples over the network.

These optimizations substantially improve our compression ratio, as demonstrated in Figure 7.6. **Making LPC compression the best compression so far for RRC shaped signals, allowing us to compress –a little– even at low SNR.**




**(a)** 8-PSK $\text{RRC}_{0.25}$. $\mathcal{F} = 2.00$

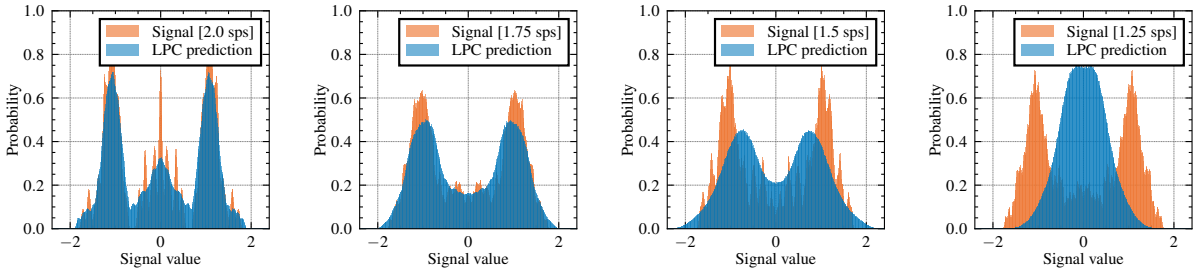
**(b)** 8-PSK $\text{RRC}_{0.25}$. $\mathcal{F} = 1.81$

**Figure 7.6:** Evolution of compression ratio $\mathcal{C}$ for various algorithms, waveforms, and SNR conditions. –If a curve is missing, it means that the algorithm compression ratio is too low to be displayed.–

## 7.4 Limitations

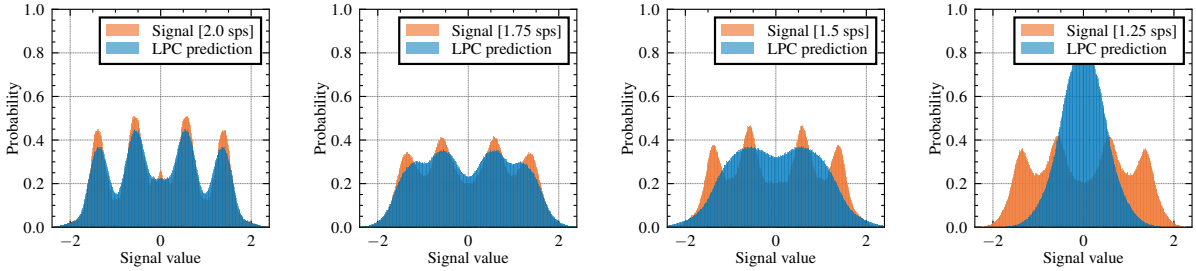
While LPC encoding has significant advantages, including fast coefficient calculation and low hardware complexity for residual computation via a Finite Impulse Response filter (FIR) filter [150], it is also well suited for nonstationary signals by segmenting them into stationary intervals for coefficient computation, an approach employed by audio codecs such as FLAC and Shorten. Note that for an ergodic and stationary signal, LPC coefficients strictly equate to AR coefficients [151].

However, LPC has limitations. The LPC method only considers the temporal distribution, which makes the prediction less accurate if the autocorrelation resembles a Dirac function. This can lead to a residual with higher entropy and non-Gaussian statistics, adversely affecting the performance of the Rice entropy coder (Figure 7.7).




**(a)** $\mathcal{F} = 2.00$ **(b)** $\mathcal{F} = 1.75$ **(c)** $\mathcal{F} = 1.50$ **(d)** $\mathcal{F} = 1.25$

**Figure 7.7:** Histograms of noiseless QPSK signals and their prediction using LPC predictors. $\mathcal{F}$ indicate the number of samples per symbol (also called the oversampling factor). The signals are shaped with a 0.25 roll-off resampled from a unique source using polyphase resampling.




**(a)** $\mathcal{F} = 2.00$ **(b)** $\mathcal{F} = 1.75$ **(c)** $\mathcal{F} = 1.50$ **(d)** $\mathcal{F} = 1.25$

**Figure 7.8:** Histograms of noiseless 8-PSK signals and their prediction using LPC predictors.

This degradation in performance becomes even more pronounced when dealing with signals employing higher modulation schemes. Despite these drawbacks, resampling remains an essential step in achieving the best compression ratio possible.

Figure 7.9 shows that the relative efficiency of LPC **drops when the number of samples per symbol is lowered**, going from an additional compression gain of 10% at $\mathcal{F} = 75$ to a gain of 0% at $\mathcal{F} = 1.25$ that highlights the limitations discussed earlier. **In the end, the entire compression is actually supported by the resampling stage, and we have no way to actually benefit from both methods**.

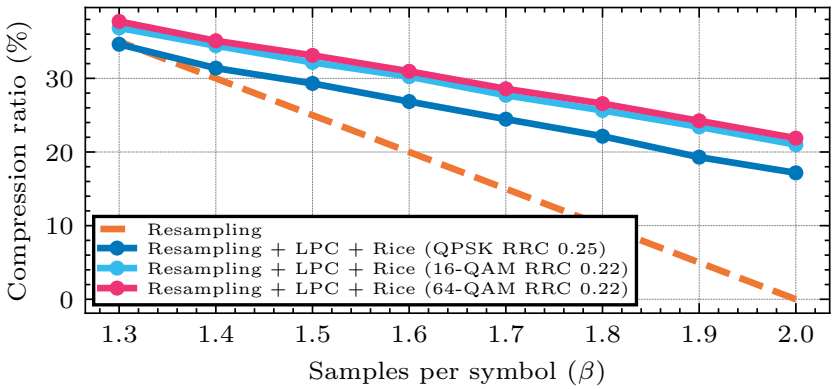



**Figure 7.9:** Evolution of global compression ratio when a re-sampling followed by a compression using LPC is performed. The original signal oversampling is $\mathcal{F} = 2$, it is a 8-PSK signal generated by a *Safran NuRon Engine* modulator, shaped using a 0.25 roll-off RRC that allows a theoretical minimum for $\mathcal{F}$ of 1.25 samples per symbol. We note that the compression efficacy of LPC decreases as the number of samples per symbol is reduced.

Note that alternative approaches such as ARMA were evaluated, but ultimately discarded due to concerns related to filter stability and limited performance improvement [152]. In addition, there have been investigations into LPC enhancements to improve prediction accuracy [153], but the gains achieved did not meet the expected standards.

## 7.5 Conclusion

The exploration of Linear Prediction Coding (LPC) within the domain of predictive compression has shown several advantages and limitations. **LPC offers a robust method for compressing data with its fast coefficient calculation and low complexity of software (or hardware)**, making it a suitable candidate for implementation on FPGA and other embedded systems. This suitability stems from its reliance on a simple linear model to predict future samples based on past samples, effectively reducing the entropy of the residual signal and thus improving compression efficiency.

One of the primary strengths of LPC is its ability to **accurately model the spectral characteristics of a signal**, which has been effectively used in various audio codecs such as FLAC. The empirical results from applying LPC to baseband signals have shown promising compression ratios but are highly dependent on the shaping applied. However, LPC's effectiveness diminishes as the predictability of the signal decreases, as observed with signals having autocorrelation functions resembling Dirac functions.

Practical implementations of LPC-based compression, such as the FLAC codec, highlight the robustness of the method under varying signal conditions, including **different levels of noise and resampling rates**. Despite FLAC's suboptimal performance due to its design for audio signals and inclusion of unnecessary metadata, **custom LPC implementations can achieve better compression ratios by optimizing coefficient calculation and reducing overhead.** However, LPC is not without limitations. The method's reliance on linear prediction means that it may struggle with nonlinear signal characteristics, which can result in higher residual entropy and less efficient compression. Furthermore, while resampling can improve compression efficacy, it is not always feasible to control the samples per symbol in practical applications, which limits the overall benefit of LPC.

Given these findings, the future of predictive compression could benefit significantly from exploring nonlinear predictors. Nonlinear models have the potential to capture more complex signal dependencies and improve prediction accuracy. **Incorporating nonlinear predictors into the compression framework could address the limitations of LPC, offering a more adaptable and efficient solution for various types of signal.**

While LPC remains a valuable tool for signal compression due to its simplicity and low computational requirements, **advancing predictive compression techniques through nonlinear predictors represents a promising direction for future research**. This approach could unlock new levels of compression efficiency and adaptability, meeting the evolving demands of modern telecommunications and digital communications systems.

Chapter 8

# Non-linear PPM compression

***When one NuRoN is not enough***


8.1 Motivations . . . . . **131**
8.2 Choice of the model . . . . . **132**
8.2.1 Taylor series model . . . . . 132
8.2.2 Volterra model . . . . . 132
8.2.3 Deep neural models to the rescue . . . . . 135
8.3 Neural Network-based predictors . . . . . **136**
8.3.1 General approach . . . . . 136
8.3.2 Simulations setup & training methodology . . . . . 138
8.3.3 A word about resources usage . . . . . 139
8.4 Compression on noiseless signals: TX channel . . . . . **140**
8.4.1 Standalone modulations . . . . . 140
8.4.2 DVB-S2 and multi-modulation Standards . . . . . 142
8.4.3 Discussion . . . . . 144
8.4.4 Selection of hyper-parameters . . . . . 145
8.5 Compression in presence of noise: RX channel . . . . . **147**
8.5.1 Experimental setup . . . . . 147
8.5.2 Results and discussion . . . . . 147
8.6 Implementation & optimization . . . . . **150**
8.6.1 Deploying neural networks for high throughput applications . . . . . 150
8.6.2 Pruning & quantization . . . . . 151
8.6.3 Implementing neural networks on FPGA . . . . . 153
8.6.4 Results for an FPGA implementation of the shallow network . . . . . 153
8.7 Conclusion and future perspectives . . . . . **156**


## 8.1 Motivations

In the context of *RF over IP*, we saw that linear compression methods such as LPC have been extensively explored and have been shown to be effective. However, the complexity of real-world signals often necessitates the exploration of nonlinear prediction methods to achieve higher compression efficiency.

Signals used for satellite communications have several interesting properties, which are not encountered with audio or medical signals, when there is no noise:

- The signals are all stationary or cyclo-stationary: This means that the autocorrelation coefficients vary periodically over time.
- The statistical properties of the signals are perfectly defined and calculable, and therefore these signals can be considered to come from an ergodic process.
- The envelope of the signals is perfectly known and predictable. Sometimes even constant (with waveforms like GMSK or SOQPSK).
- The signals maintain constant power over time (in TX).
- The dynamic range used by the signals is also constant and the quantization bias is perfectly defined.

The exceptional statistical and temporal regularity of the signals studied **makes nonlinear predictors worth considering**, despite their typical drawbacks, such as implementation complexity and unstable performance. In the following section, we will study some well-known models used for nonlinear approximation of function to choose the one that best fits our needs.

## 8.2 Choice of the model

### 8.2.1 Taylor series model

One classical approach to extending linear methodologies to nonlinear contexts is the utilization of Taylor series expansions. Taylor series provide a means to approximate nonlinear functions as infinite sums of polynomial terms, based on derivatives of a function at a single point.

$$y(x) = \sum_{k=0}^{\infty} \frac{1}{k!} \frac{\partial^k f}{\partial x^k}(a)(x-a)^k \tag{8.1}$$

While Taylor series are well-known and can theoretically approximate any smooth function given a sufficient number of terms, their application in predictive compression presents several limitations. Firstly, the effectiveness of a Taylor series approximation **is heavily based on the choice of the expansion point**; a poor choice can lead to significant approximation errors, particularly for functions with high nonlinearity or in the presence of noise.

Furthermore, the computational **overhead associated with permanently calculating higher-order derivatives** and evaluating polynomial terms, although beneficial for improving compression, **introduces significant computational burden and potential latency**. This added complexity may not be compatible with the stringent constraints of communication systems. This is particularly relevant in the context of telecommunications and embedded electronics, where computational resources are limited, and maximizing efficiency is crucial.

In light of these considerations, while Taylor series offer a bridge between linear and nonlinear methods, they fall short of providing an adequate solution for predictive data compression.

Limitations in approximation accuracy, adaptability, and computational efficiency lead to the exploration of alternative nonlinear approaches, such as Volterra series, which can offer better performance by catering to the specific characteristics of the signal being approximated.

### 8.2.2 Volterra model

#### 8.2.2.1 Presentation

The Volterra expansion, also called Volterra's series [154], is a nonlinear model used to approximate any function with a sequence of coefficients called "kernels". It is similar to Taylor series decomposition but, unlike the latter, it does not use the notion of derivative to design its model but is based on the usage of different orders of the signal self-correlation. This characteristic allows the Volterra series to obtain a "memory" effect. This effect is particularly interesting in the context of the design of nonlinear predictors. Moreover, the ability of Volterra series to exploit the self-correlation of signals makes them a candidate of choice in the context of a signal processing application.

The $P^{\text{th}}$ order Volterra decomposition is defined by the formula :

$$y(k) = h_0 + \sum_{p=1}^{P} \sum_{\tau_1=a}^{b} \cdots \sum_{\tau_p=a}^{b} h_p(\tau_1, .., \tau_p) \prod_{j=1}^{p} x(k-\tau_j) \tag{8.2}$$

In the case of a predictor, we want the prediction to be made on the last W samples of the signal. This leads to $a = 1$ and $b = \mathrm{W}$ and the following formula:

$$y(k) = h_0 + \sum_{p=1}^{P} \sum_{\tau_1=1}^{\mathrm{W}} \cdots \sum_{\tau_p=1}^{\mathrm{W}} h_p(\tau_1, .., \tau_p) \prod_{j=1}^{p} x(k - \tau_j) \tag{8.3}$$

| Order | Expression |
|---|---|
| 0 | $h_0$ |
| 1 | $h_0 + \sum_{\tau_1=1}^{\mathrm{W}} h_1(\tau_1) y(k - \tau_1)$ |
| 2 | $h_0 + \sum_{\tau_1=1}^{\mathrm{W}} h_1(\tau_1) y(k - \tau_1) + \sum_{\tau_1=1}^{\mathrm{W}} \sum_{\tau_2=1}^{\mathrm{W}} h_2(\tau_1, \tau_2) y(k - \tau_1) y(k - \tau_2)$ |

**Table 8.1:** Volterra decomposition for different orders.

We observe that the first order of a Volterra decomposition corresponds exactly to an Autoregressive model (AR) decomposition in the case of autoregressive signals. More generally, it can be interpreted as a Linear Prediction Coding (LPC) decomposition. At the second order, the decomposition involves the double product of the sum of the signal samples, introducing the nonlinearity characteristic of the Volterra series. Finding the second-order kernels requires using the cross-correlation of the signal's self-correlation coefficients, a concept referred to as Higher Order Coherence.

Moreover, we can observe that if we have 10 coefficients in the first-order decomposition then we will have 100 in the second, then 1,000 in the third... Thus, increasing the order of the Volterra decomposition exponentially increases both the resources and the time needed for the computation, as well as the computational load needed to find the Volterra kernels.

#### 8.2.2.2 Volterra decomposition as ideal predictor

**Fréchet's approximation theorem [155], [156]**

A time-invariant functional relation (satisfying certain very general conditions) can be approximated uniformly and to an arbitrary degree of precision by sufficiently high finite-order Volterra series.

Note that the theorem does not give any indication as to how many terms are needed for a good approximation, which is an essential question in applications. However, this theorem indicates that for certain categories of signals, an infinite-order Volterra decomposition will lead to a perfect prediction. In practice, it is clear that such a degree of prediction is unattainable.

#### 8.2.2.3 Limitations

Even if the Volterra decomposition theoretically offers a robust method to build predictors of infinite precision, it actually poses many problems:

1 · First of all, the number of coefficients required to realize a Volterra decomposition follows an exponential evolution: if we choose to use 10 coefficients for the first-order kernel, we will need 100 for the second-order kernel, then 1000 for the third order... This explosion in the number of coefficients quickly leads to the impossibility of hardware implementation and an explosion of computation time in software architectures. We can then try to see if truncating the number of coefficients (by keeping only some of them) can limit this effect. In practice, however, it adds a significant additional difficulty, since it would be necessary to determine which coefficients are "interesting" and which are not. This represents a large investment in time and resources for each class of signals studied.

2 · Second, the supposed (cyclo-)stationarity of our signals only ensures stability of the first-order coefficients. Thus, the stability of the second-order and even the third-order cannot be considered as guaranteed. This would mean that the coefficients would need to be recalculated periodically and shared between the compressor and the decompressor, which, in addition to requiring a heavy computational load (much greater than for calculating LPC coefficients), would necessitate a network load that would quickly become unacceptable, as it also grows exponentially with the degree of decomposition.

3 · Finally, the difficulties in calculating the kernels make any attempt at implementation particularly lengthy and discouraging, employing highly complex mathematical models [157]–[159] that pose significant challenges in their application for prediction. This aspect goes against our search for a systematic, adaptable, and reproducible method for the compression of communication signals.

### 8.2.3 Deep neural models to the rescue

Wray & Green [160] demonstrated in 1994 that **a properly trained 3-layer perceptron, a basic neural network structure, can be equivalent to a Volterra approximation when paired with a suitable activation function.** Given these findings, it appears that neural networks hold considerable promise for predictive tasks. Naturally, our interest is piqued by investigating their precision. However, we must consider the practicality of implementation: If the network structure is too complex or deep, it may not be feasible for real-world compression applications in actual systems.

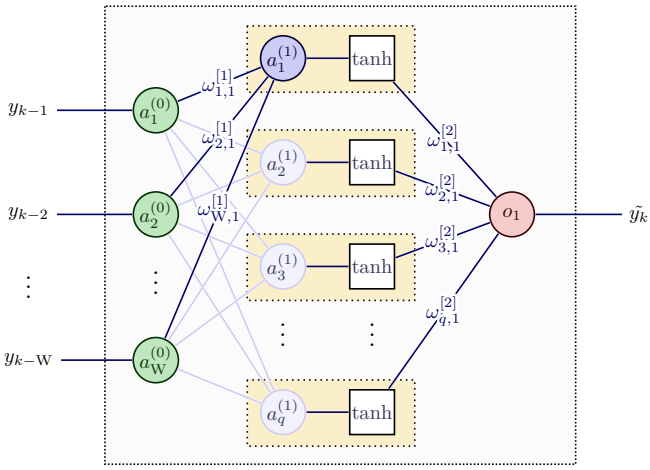



**Figure 8.1:** Simple 3-Layer neural network with a tanh activation function (NAR-tanh). This network is equivalent to an approximated infinite Volterra decomposition according to Wray & Green.

The enhancement of predictive compression through neural networks is not a novel concept. Such architectures have been explored since the early 2000s [161], with applications extending to various fields:

- The use of neural networks for lossless compression of EEG data has been a topic of research [162], using the predictive capabilities of these networks.
- Recurrent neural networks (like ones using LSTM layers) have been used for lossless compression of telemetry data [163], demonstrating their utility in handling sequential data.

Finally, the integration of neural networks within the physical layer of wireless communication systems has generated increasing interest. Wang et al. [164] provided a comprehensive report on the use of deep learning in the physical layer, with their current limitations.

## 8.3 Neural Network-based predictors

### 8.3.1 General approach

Using the previous results, the main idea of our research was to see **if some well-designed neural network architectures could be used as efficient and agnostic predictors**. Although the concept itself has been used for a long time, its use in this scenario introduces a novel approach. In order

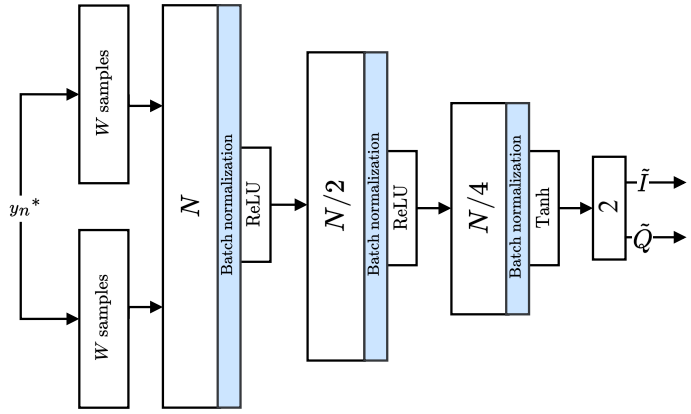



**Figure 8.2:** Shallow network architecture: A simple and robust network that outperforms LPC [165], which can be easily trained on a conventional CPU and has a very high inference rate. This allows compression at low latency and high throughput.

.

to build robust and adaptable predictors that can be efficiently implemented across various platforms, such as CPUs, GPUs, and FPGAs, we conducted a comprehensive study of multiple neural network architectures. Our selection criteria centered on diversity in complexity and functionality. The following architectures were chosen for their distinctive characteristics:

- **Shallow Network**: One of our contributions, the Shallow Network, is designed for simplicity and efficiency, characterized by its dense, fully connected structure. It includes two layers with ReLU activation, followed by a layer with tanh activation. Figure 8.2 presents its architecture.
- **Bi-GRU Network**: This architecture employs a well-known bidirectional Gated Recurrent Unit (GRU) approach, which consists of a ReLU layer followed by two GRU layers with tanh activation functions [166]. This network leverages the sequential nature of data for improved temporal pattern recognition.
- **T-Conv Network**: Utilizing a transposed convolutional architecture [167], this network features a series of transposed convolutional layers combined with batch normalization and ReLU activations to up-sample the input. It concludes with adaptive max-pooling to standardize the output dimensions, followed by a set of Fully Connected (FC) layers that include further batch normalization, ReLU activation, and dropout for regularization. This architecture is based on established convolutional network designs.

The goal of the first network is to mix the results observed on the sets of linear coefficients with the Volterra decomposition. The objective is to obtain sets of Volterra kernels that can be interchanged on the fly. Note that this whole description is a purely theoretical explanation of the internal functioning of these networks. **At this stage of complexity, actually extracting the kernels over time appears to be a very difficult task.**

Regarding the second network, the GRU [166] layers are a type of recurrent neural network layer that can selectively remember or forget information over time, allowing them to model long-term dependencies in sequential data. Since they are commonly used for time series forecasting, it appeared like a natural choice as a comparison for our use case. From a theoretical point of view, it is much more difficult to explain how such layers work, and they cannot be linked with Volterra decomposition.

Finally, the T-Conv network uses deconvolution layers to extract a good approximation of the transmission chain, leading to very accurate results and a good prediction.

To benchmark our methodologies against current state-of-the-art practices and previous studies, we trained two additional networks based on the Non-linear Autoregressive model (NAR) architecture. This particular architecture is essentially a one-layer perceptron with an activation function. The specific NAR networks we examined are:

- **Volt Network**: This NAR network uses a tanh activation function and represents a fixed infinite Volterra series decomposition, as explained in [160].
- **Linear Network**: Operating without an activation function, this NAR network essentially parallels a Linear Prediction Coding (LPC) decomposition. The expected results should align with the findings of [148]. Once trained, the LPC coefficients can actually be extracted from these network widths (see Section 7.2.2.2).

All these networks will also be compared to the actual LPC compression scheme used in the state-of-the-art, using fragmentation methods similar to the one used in the FLAC algorithm. Note that all the networks have been trained on complex data, represented as vectors of length 2 (one dimension for I, one for Q). This means that each network receives as input a matrix of size $(W, 2)$.

### 8.3.2 Simulations setup & training methodology

This section briefly outlines the training methodology for neural networks used in processing noiseless space communication signals, specifically for the TX link –when the signal is sent from the ground station to the satellite–.

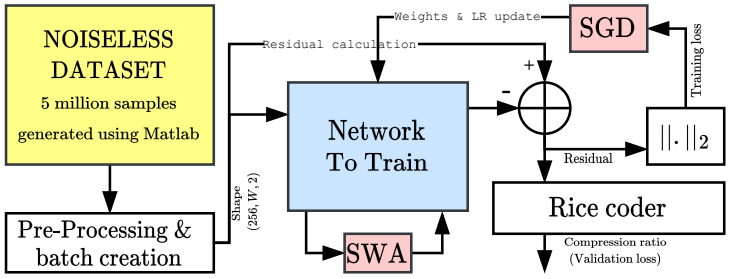



**Figure 8.3:** Schematic representation of the neural network training pipeline, from dataset preparation to model evaluation using a Rice coder.

The training is conducted using *Pytorch* and *Pytorch Lightning*, which span more than 250 epochs. The networks are optimized using Stochastic Gradient Descent (SGD) with Nesterov momentum of 0.2 and a dynamic learning rate controlled by a *Cosine Annealing Warm Restarts* Learning Rate Scheduler (LRS) with a base period of 4 epochs. The key steps of the training process include the following:

1 · Utilization of 5 million samples per dataset, generated with MATLAB®, split in 85% training and 15% validation. The samples were originally sampled at 2 samples per symbol.

2 · Pre-processing steps involving FFT resampling to achieve the target number of samples per symbol and data quantization to 8 bits. Data are then normalized to have unit power.

3 · Batch processing with a size of 256, balancing computational efficiency and noise averaging in noisy data scenarios [165].

4 · During processing, application of a Stochastic Weight Averaging (SWA) with an LRS of $10^{-2}$.

5 · Validation using the compression ratio metric and using a Rice entropy coder for residual analysis.

6 · Selection and export of the top five models according to the compression ratio, converted to ONNX format for CPU-based simulation on a test dataset.

### 8.3.3 A word about resources usage

| Network | Number of parameters |
|---|---|
| LPC | 48 (complex data, so $2 \times 24$) |
| Volt ($N = 128$) | $\approx$ 2,200 |
| Shallow | $\approx$ 15,000 |
| Bi-GRU | $\approx$ 75,000 |
| T-Conv | $\approx$ 350,000 |

**Table 8.2:** Number of parameters for each network, without any optimization step (weight quantization, node pruning or network compression). The training window is W = 24. The given number includes the bias and the weights.

The data presented in Table 8.2 provides an illustration of the parameter count for each neural network architecture explored in this study. The LPC network, serving as our baseline, demonstrates the simplicity and minimal computational overhead typical of linear predictive coding. In contrast, more sophisticated neural networks, such as the Shallow, Bi-GRU, and T-Conv architectures—exhibit a significantly higher number of parameters, reflecting their enhanced capacity to capture complex patterns in data.

These differences in parameter count have substantial real-world implications. Networks with a larger number of parameters generally offer better predictive performance, which is essential for high-precision applications in telecommunications and data compression. However, this increased complexity comes at the cost of more computational and memory resources.

In practical terms, the deployment of these networks requires careful consideration of the available hardware capabilities. For instance, while the Shallow Network can be efficiently implemented on conventional CPUs, the Bi-GRU and T-Conv networks might necessitate more powerful GPUs or specialized hardware like FPGAs to achieve real-time performance.

Moreover, the high parameter count in networks such as T-Conv, with approximately 350,000 parameters, demands substantial memory bandwidth and computational power, potentially limiting their deployment to environments without robust hardware infrastructure. This trade-off between model complexity and resource consumption must be balanced to ensure the feasibility of these advanced neural network models in practical, real-world applications.

Ultimately, the choice of network architecture for predictive tasks in telecommunications should consider not only the predictive accuracy, but also the resource constraints of the deployment environment. This approach ensures that the benefits of advanced neural network models can be realized without compromising on performance due to hardware limitations. This topic will be explored further in Section 8.6.

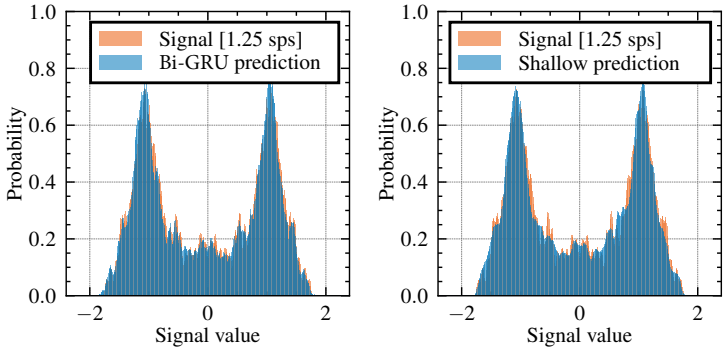
## 8.4 Compression on noiseless signals: TX channel

### 8.4.1 Standalone modulations

To evaluate the performance of the proposed predictors, we trained several networks on modulations commonly used in satellite communications, focusing on QPSK and 8-PSK, both shaped with a RRC filter of 0.25 roll-off and generated with an oversampling of 2.0 samples per symbol. The signals were synthesized using MATLAB®. The results obtained are presented in Figures 8.5a and 8.5b.

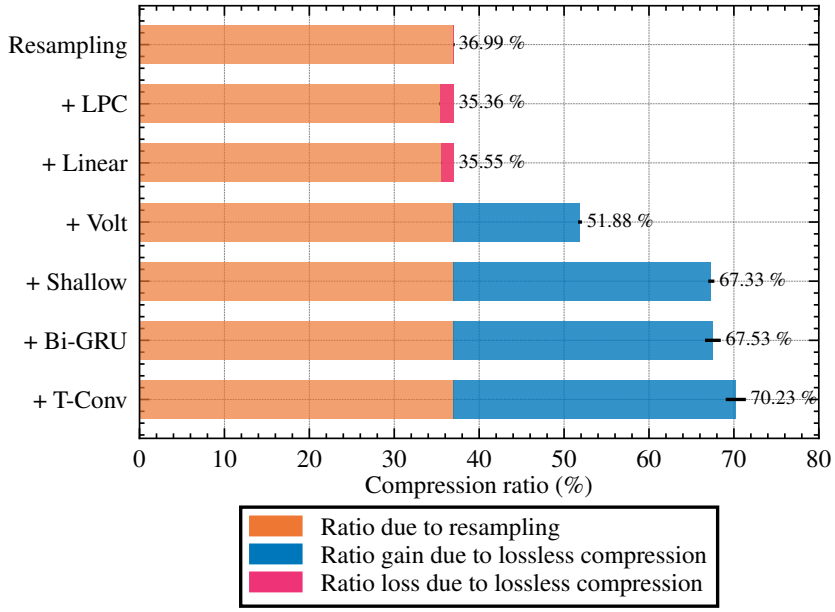
Regarding performance, neural network-based compressors exhibit up to a 30% higher compression ratio compared to the LPC-based compressor. Despite its simplicity, the *shallow* architecture achieves results that exceed 90% of those of the most advanced networks such as Bi-GRU and T-Conv, but are significantly larger, consuming more time to train and more resources to infer. Regarding the Bi-GRU, we noticed that enabling the bidirectional effect of the GRUs used in this network has shown substantial improvements in the compression ratio. Furthermore, the usage of GRU instead of LSTM as in [163] has shown a convergence improvement during training.

Figure 8.4 presents an observation of the predictive histograms of the networks studied. It shows a clear improvement in prediction for TX signals compared to Figure 7.7.

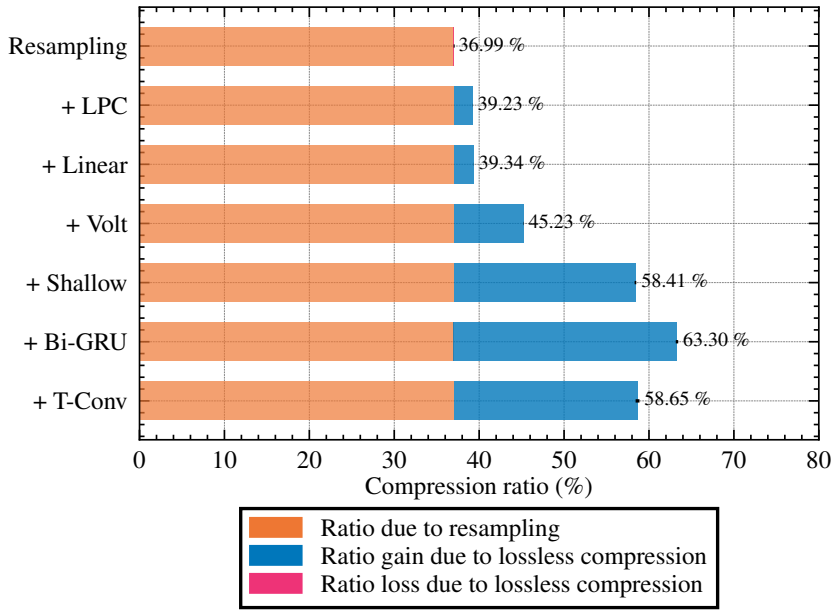



**(a)** Bi-GRU network **(b)** Shallow network

**Figure 8.4:** Histograms of QPSK signal and its prediction using Neural predictors. $\mathcal{F} = 1.25$ for both figures. Predictions still follow the signal histogram for low values of oversampling.




**(a)** QPSK




**(b)** 8-PSK

**Figure 8.5:** Compression performance of the combination of resampling with lossless techniques applied on QPSK and 8-PSK (RRC 0.25) signals.

### 8.4.2 DVB-S2 and multi-modulation Standards

The development of new markets, driven by new space activities, has led to the emergence of interest in TX upstream channels (data or telemetry) using telecommunications standards that were previously reserved for RX downstream transmissions. This is especially true for DVB-S2 and DVB-S2X, which have recently seen a renewed surge in interest.

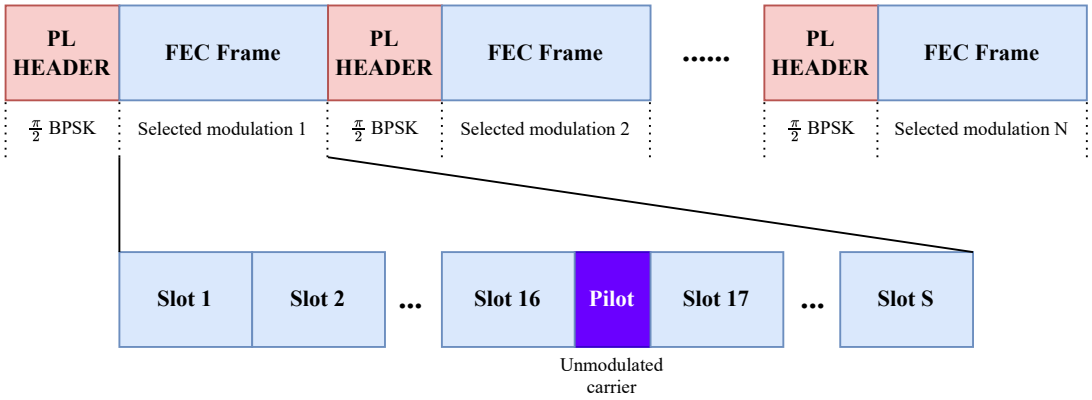



**Figure 8.6:** Structure of a DVB-S2 frame [168]–[170].

Digital Video Broadcasting - Satellite - Second Generation (DVB-S2) is a communication standard developed specifically for the transmission of satellite broadcasts. The signal is sent in the form of frames, modulated with $\pi/2$-BPSK in the PL HEADER, followed by modulations ranging from QPSK to 32-APSK normalized in power. This last point means that the dynamic range of the signal is constantly changing. Consequently, in LPC applications, frequent recalibration is necessary to adapt to the signal's evolving nature and maintain an optimal compression ratio. The DVB-S2 signal also incorporates pilots and error correction codes and is segmented into slots. Furthermore, these signals are band-limited with significant cosine shaping and a roll-off factor of 0.25. As discussed above, resampling at such a rate can substantially impair the performance of LPCs. The objective is therefore to see if a well-thought neural network can offer an interesting compression of the DVB-S2 signal.

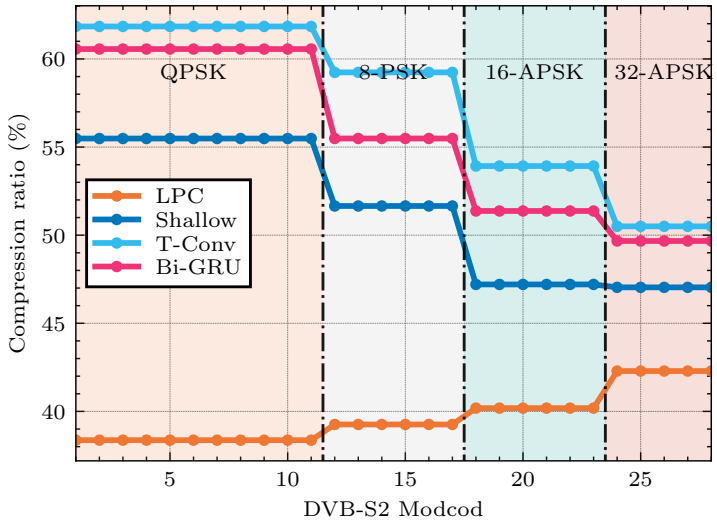



**Figure 8.7:** Evolution of global compression ratio for each DVB-S2 modcod. We observe that the compression ratio for lossless compression does not depend on the FEC used but only on the modulation index. Note that 37.5% of the compression ratio is due to the decimation / resampling step, the rest comes from our compressor.

Figure 8.7 presents the results obtained for classical modulations. We tested three different networks: An RNN network preceded by a dense layer (Bi-GRU), a deconvolution network (T-Conv) and a 3-layer network (Shallow). The window uses 24 complex samples to compute the prediction.

The training dataset consisted of a set of thousands of DVB-S2 frames using random modulation, but distributed so that the total number of modulated samples for each constellation was identical. The 28 test datasets (one for each modcod) were composed of 20,000 frames of a unique modcod.

The findings indicate that, across all modulation schemes, every network consistently exceeds the adaptive prediction of the LPC. Specifically, these networks deliver an improvement in compression ratio ranging from 15% to 20% over LPC at lower modulation indices. Moreover, they continue to provide an improvement of up to 5% in compression efficiency even at the highest modulation index (32-APSK). This also applies when multiple frames with different modulations are sent consecutively. We can notice that our shallow network still offers performance close to those offered by more complex networks.

In this scenario, we find that LPC behaves in the opposite way: It performs better in compressing signals with the highest modulation index. This occurs because DVB-S2 frames are standardized based on power rather than amplitude. As a result –and as presented by Figure 8.8–, there are significant variations when transitioning from high-order modulations, such as 16-APSK and 32-APSK, to lower-order ones, such as BPSK, QPSK, and 8-PSK. This variation hinders the signal's ability to utilize the full dynamic range assigned to it when a lower-order modulation is in use. **Essentially, it is the entropy coder that effectively compresses the unused dynamic range rather than compressing the signal itself.**

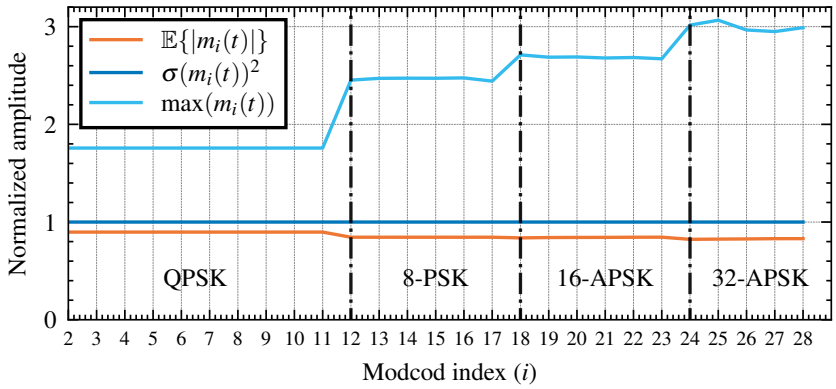



**Figure 8.8:** Each signal $m_i(t)$ of each modcod $i$ has been normalized to the same power ($\sigma^2 = 1$). But their amplitude range is different. When transmitting a DVB-S2 signal, a lower modulation index leads to a large non-used amplitude range that is easily compressed by the entropy coder

### 8.4.3 Discussion

We observe that the increase in modulation reduces the performance of our predictor. As the modulation order increases, more bits are involved in state changes, especially the more significant bits, which can undergo more abrupt transitions and thus become less predictable. With a 32-APSK modulation, the actual gain is only 5% higher than that of an LPC predictor. We can easily imagine that higher modulations, such as 128-APSK or 256-APSK, will not gain any benefit from this method. For standards such as DVB-S2X, a lossy compression scheme should be considered to obtain further compression improvements.

Figure 8.8 illustrates that each signal $m_i(t)$ of each modcod $i$ has been normalized to the same power ($\sigma^2 = 1$). However, their amplitude ranges differ. When transmitting a DVB-S2 signal, a lower modulation index leads to a large non-used amplitude range **that is easily compressed by the entropy coder**.

> Note that all the signals used to make the previous comparisons have been generated and refined to address the limits of LPC-based predictors. In real implementations, the **performance of a predictor is highly dependent on the characteristics of the signal**. These include signal dynamic range, quantization depth, quantization uniformity, sampling rate, modulator clock precision, resampling method, etc. For these reasons, **it is highly unlikely that two systems with the same performance will be found in the literature, regardless of the fact that they would use the same modulation and the same types of predictor**.

### 8.4.4 Selection of hyper-parameters

This section discusses the different values that training and simulation parameters must have to obtain optimal results. Note that the final choice will always be driven by the particularities of each platform and signal studied.

#### 8.4.4.1 Size and number of layers

As presented in Figure 8.9, increasing the number of neurons in the hidden layer will actually increase the precision of Volterra kernels. Due to Frechet's approximation theorem, this increment would likely lead to an improvement of the prediction.

However, this increment is not linear. Simulations have shown that more than 128 neurons in a single layer do not lead to any interesting improvement. After that, it will be more interesting to increase the number of hidden layers.

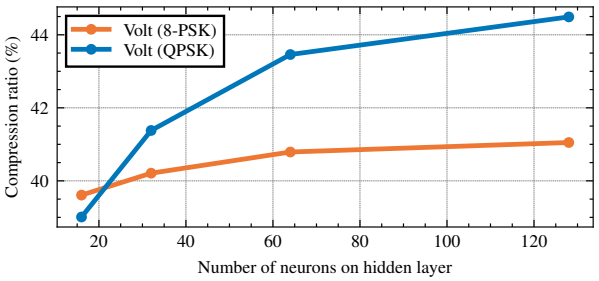



**Figure 8.9:** Evolution of global compression ratio depending on neurons on hidden layer for QPSK and 8-PSK modulated signals. Studied for Shallow network.

#### 8.4.4.2 Data complexity

Working with baseband signal usually means working with complex data. Performance presented in section 8.4.3 is directly trained on complex data that are represented as a tensor of shape $(\mathrm{W}, 2)$ at the network input. This method accelerates the entire training process (no need to train one network per channel), but has the drawback of increasing the number of parameters on the first and last layers by a factor of two, as well as increasing the number of epochs necessary to converge. However, it allows us to benefit from a possible cross-correlation between the $I$ (real) and $Q$ (imaginary) channels, which would not be possible with two separately trained networks.

The use of a network using complex weights [171] has also been studied, without any real improvement in prediction.

#### 8.4.4.3 Window width W

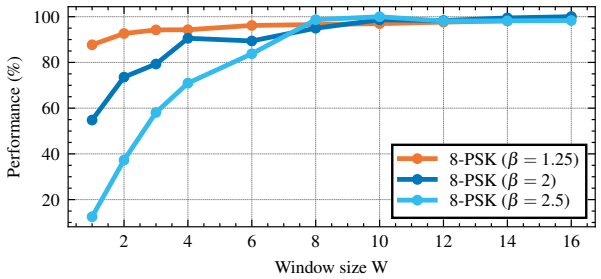



**Figure 8.10:** Evolution of global compression ratio depending on window size W for Shallow network.

In order to increase the computation speed and decrease the usage of resources, it is important to choose the correct amount of samples to perform the prediction, *that is,*, the correct value of W.

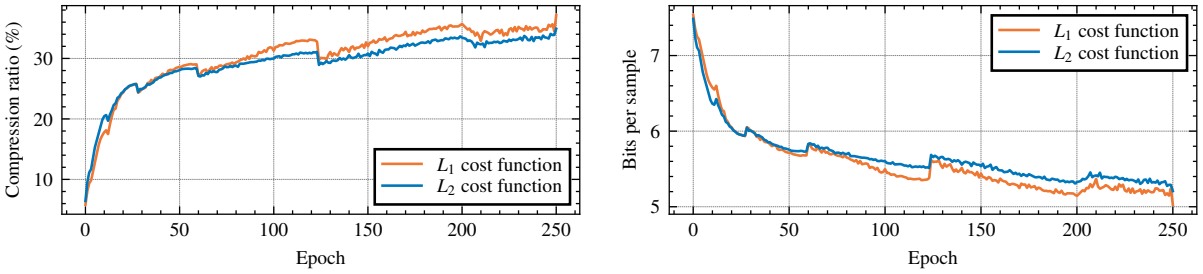
For classical modulations with a number $\mathcal{F}$ of samples per symbol between 1.25 and 2.5, Figure 8.10 shows that a window size of 12 is sufficient to obtain the best compression ratio most of the time.

#### 8.4.4.4 Cost function

The choice of the cost function is a crucial step in the training of a neural network. It is the function that will be minimized during the training process. In the case of the prediction of Volterra kernels, the choice of the cost function is not obvious since **the goal of this function is to find a**

**solution that minimizes the output residual entropy**, or, in a more practical way, produces residual values that have the best statistics before feeding the Rice encoder in order to minimize the number of bits used to encode the signal. The original LPC uses a $\ell_2$ norm minimization, but $\ell_2$ minimization actually minimizes energy and the output residual will then follow a Gaussian-like statistic. On the other hand, using a $\ell_1$ will likely produce a geometric Laplacian statistic at the output, which has better performance with Rice coding, but possibly a higher entropy...

In order to compare the different cost functions, we trained a network with the same parameters as those presented in Section 8.4.3 but with different cost functions. The results are presented in Figures 8.11 a and b. We can see that the $\ell_1$ norm is the best choice for this application.




**(a)** **(b)**

**Figure 8.11:** Impact of the cost function on the compression ratio.

The relative performance on $\ell_1$ can be determined by formula $1-\mathcal{C}_{\ell_2}/\mathcal{C}_{\ell_1}$. As illustrated in Figure 8.12, employing $\ell_1$ as the cost function enhances the final compression ratio by relative 10% compared to using $\ell_2$.

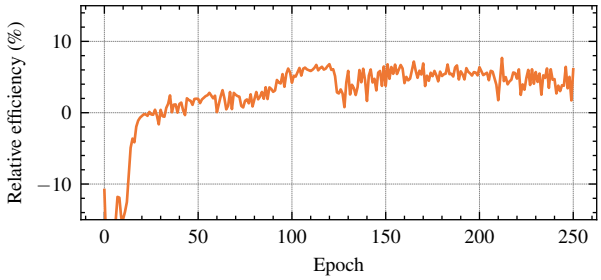



**Figure 8.12:** Relative efficiency over $\ell_1$

Based on these findings, an optimal range for the number of neurons is between 128 and 256. The window size W should exceed 10, with 12 or 16 being suitable choices, and the MAE $\ell_1$ is recommended as the cost function.

## 8.5 Compression in presence of noise: RX channel

*Results and analysis in this section have been presented in [165] during ICETC International Conference in December 2023. Part of the text and figures are reproduced here for the sake of completeness.*

### 8.5.1 Experimental setup

Figure 8.2 presents the neural network used in our study, what has been called the *Shallow* architecture. It is a multilayer feedforward network with four layers. **The chosen architecture is relatively simple, yet powerful, and careful consideration is given to maintaining a balance between complexity and performance.** The usage of *batch normalization* is highly beneficial for noisy data. **By normalizing inputs across each mini-batch, it stabilizes learning, reduces training epochs required, and enhances robustness**. It makes the neural network less sensitive to noise, mitigating issues such as vanishing or exploding gradients. Thus, simulations showed that it improves the stability and accuracy of the predictions and the compression performance in noisy conditions.

Moreover, the architecture was designed with a deliberate intention to keep the model light (low number of multiplications, no recursive layer), using a limited number of layers and neurons. The intent was to enable easy hardware implementation and parallelization while delivering strong performance. The network was trained on various sets of baseband I/Q signals, striving to predict the most likely next sequence in the data, thus reducing the residual error and improving the compression ratio.

In our study, we trained the networks so that the parameters are specifically learned for successive $E_S/N0$ ranges of 7.5dB. Our simulations show that a network trained across a wide noise range tends to underperform compared to a network tuned to a specific noise range. In practice, this is not a significant hurdle. The noise range in which a signal resides is generally known, even if the exact signal-to-noise ratio is uncertain. This knowledge helps to effectively assign the appropriate trained network. Moreover, the simplicity of training and implementing networks and the ease of switching between sets of parameters, facilitated by indicating the set used in a header, contribute to the feasibility and efficiency of this approach in real-world applications when multiple parameter sets are needed. However, this training approach can potentially induce threshold effects at the borders of various noise ranges, leading to the observed "wavelike" patterns in Figure 8.13.

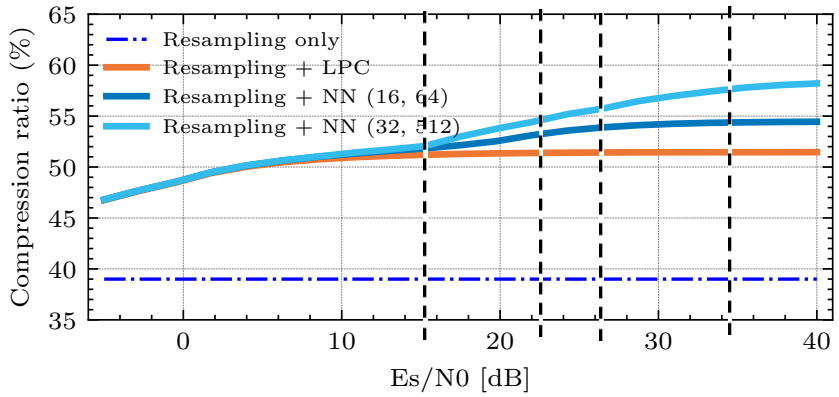
If the following examples are taken, we take $(\mathrm{W}, N)$ equal to $(16, 64)$ and $(32, 512)$, which correspond to two architectures of 4,900 and 180,000 parameters.

### 8.5.2 Results and discussion

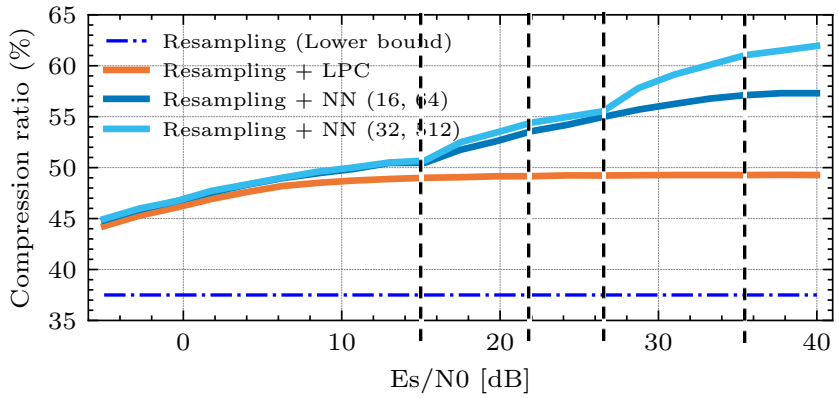
Our experiments show a clear advantage for the NN-based compression systems over LPC in high-SNR environments ($E_S/N0 > 10$ dB). Compared to LPC, our NN solutions offer compression ratios that increase by up to 12 points. This advantage is displayed primarily in scenarios that require high-accuracy data transmission and environments that are characterized by low noise. The NN-based system is proving to be a valuable asset in these contexts.

In contrast, under low SNR conditions, the performance of the NN-based system converges to that of the LPC-based system. Note that LPC can be seen as a really simple case of neural network since training a network with no activation function (*i.e.*, without nonlinearities) will actually find the LPC coefficients [148]. The inherent randomness of noise introduces some unpredictability that challenges the predictive abilities of the neural network. The network performance is thus reduced to a level similar to that of LPC in these noisy situations. However, this particular behavior showcases the resilience of the system as it continues to maintain reliable performance even in highly noisy conditions.

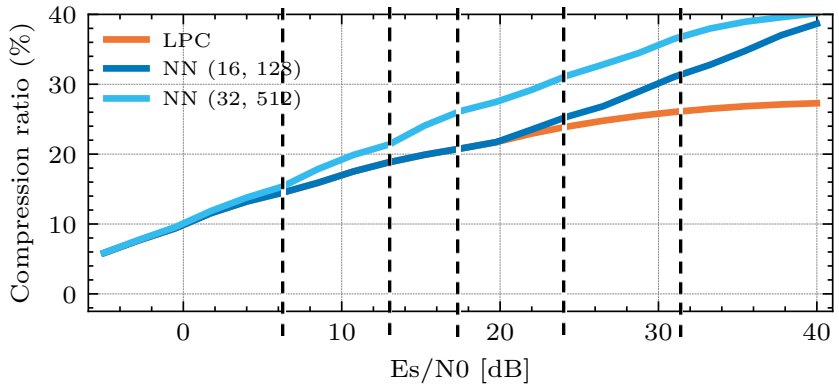
Furthermore, an interesting trend was observed regarding the performance of neural networks




**(a)** 16-QAM signal shaped with $\text{RRC}_{0.22}$. Resampling is 1.25 samples per symbol




**(b)** 8-PSK signal shaped with $\text{RRC}_{0.25}$. Resampling is 1.22 samples per symbol




**(c)** GMSK signal with $BT = 0.3$

**Figure 8.13:** Compression ratio obtained for various signals and SNRs. All signals have an original sampling rate of 2 samples per symbol. Simulations showed that performing a resampling before applying lossless compression increases the global compression ratio. NNs were specifically trained for successive SNR ranges. The parameter mentioned for each network is the tuple $(\text{W}, N)$.

in relation to the number of their parameters. Our results demonstrate that even with a relatively low number of parameters –4,900–, the neural networks still deliver commendable performance. Switching from NN(16, 64) to NN(32, 512) allows increasing the compression ratio by up to 5 points, but also requires increasing the number of parameters from 4,900 to 180,000 . Pruning NN(32, 512) would certainly

reduce the number of parameters without significant performance loss. Such strategies offer an interesting perspective on balancing performance and computational efficiency.

Finally, **in practical use cases**, such as those using the DVB-S2 standard, our method would be particularly beneficial. **Measured noise levels [172] indicate that the SNR of the received signal in clear weather can sometimes exceed 35 dB of $E_S/N0$, and can even approach 50 dB for high elevations (near 90°)**. These values are significantly higher than the operating values for all DVB-S2 modulation and coding states. In the case of using Constant Coding and Modulation (CCM) or Variable Coding and Modulation (VCM), there are time intervals where the use of neural predictors would be very useful, as they would be operating at their full potential. In Adaptive Coding and Modulation (ACM), the measured margin is so high that even with the highest modulation it remains large enough for effective compression. **This underscores the potential of deploying our neural network-based system in practical scenarios, where its ability to excel in environments with high Signal-to-Noise Ratio (SNR) can significantly improve satellite signal compression, thus increasing the efficiency of ground segment operations.**

## 8.6 Implementation & optimization

### 8.6.1 Deploying neural networks for high throughput applications

Deploying such neural networks on traditional Central Processing Units (CPUs) is impractical due to the extreme data rates, often several gigabauds, required in applications like space communications. Therefore, it is essential to use specialized hardware like GPU, FPGAs or ASICs. However, the deployment of neural networks on these architectures for industrial usage **poses several significant challenges**, particularly in terms of **timing, throughput, resources, and power consumption**. These constraints are more pronounced compared to CPUs, especially in high-demand environments such as space applications.

- Firstly, **timing** constraints on FPGAs are stringent. FPGAs operate on a fixed clock cycle and the timing of each operation must be **meticulously managed to avoid timing violations**. Neural networks, especially those with a large number of layers and parameters, require precise timing coordination. On CPUs, while there are timing constraints, **the general-purpose architecture allows more flexibility and dynamic management of processing cycles**, which is less stringent compared to the fixed logic of FPGAs.
- Secondly, **throughput** is a critical factor. FPGAs excel in their parallel processing capabilities, which makes them theoretically suitable for the high-throughput demands of neural networks. However, in practice, achieving high throughput requires careful balancing of data flow and processing units. Unlike CPUs, which can handle sequential processing efficiently, FPGAs require extensive parallelism, which can be complex to implement due to dependencies between layers in neural networks [173], [174].
- In terms of **resources**, FPGAs offer limited on-chip memory and logic resources compared to CPUs. Neural networks, particularly those with **large numbers of parameters (e.g. millions of floating point weights), can quickly exhaust these resources**. CPUs, with their abundant memory hierarchies and larger caches, handle large models more gracefully. However, the limited on-chip resources of FPGAs necessitate efficient management, weigth quantization –and then, possible precision loss– and often reduction of model size.
- **Power consumption** is another critical aspect. FPGAs are generally more power efficient for specific tasks because of their customized architectures. However, the power efficiency advantage diminishes when dealing with large neural networks, as the power required to maintain high throughput and extensive parallelism can offset the inherent efficiency of FPGAs. CPUs, despite being less power efficient on a per-operation basis, manage power dynamically across a wide range of tasks, benefiting from mature power management technologies.

In the context of space communications, where data rates are extremely high and resource constraints are stringent, deploying neural networks as-is (with floating-point weights and without pruning) is impractical. The limited on-chip resources and stringent timing requirements make it challenging to meet the high throughput and power efficiency demands of these applications.

Therefore, to address these challenges, various methods such as pruning and quantization have been developed. **These techniques aim to optimize neural networks to fit the constraints of FPGA deployments without significantly compromising performance [173], [175], [176].**

### 8.6.2 Pruning & quantization

Pruning and quantization are critical techniques in the optimization of neural networks for efficient deployment, especially in embedded systems and FPGA implementations, which are common in telecommunications environments. This section discusses the importance of these methods in the context of the implementation of neural network-based predictors for signal compression.

#### 8.6.2.1 Understanding pruning

Pruning involves **reducing the complexity of a neural network by removing redundant or less significant weights**. This is done to decrease the computational requirements and memory usage without significantly impacting the network's performance. **The main idea behind pruning is that not all neurons or structures are equally important for the network's function**, allowing for selective reduction. Several pruning techniques are available; here are two frequently used ones.

**Magnitude-based pruning** Magnitude-based pruning [177]–[179] is one of the simplest and most widely used techniques. It involves removing weights from the neural network based on their magnitude. Here is how it is typically done:

1. **Train the network:** Begin with a fully trained three-layer perceptron with ReLU activation.
2. **Calculate weight magnitudes:** For each weight $w_{ij}$ in the network, calculate its magnitude $|w_{ij}|$.
3. **Threshold determination:** Choose a threshold value. Commonly, a global threshold is used where a certain percentage of the smallest weights are pruned.
4. **Prune weights:** Set weights with magnitudes below the threshold to zero.
5. **Fine-tuning (optional):** Retrain the network with the pruned weights to recover any potential performance loss. This usually involves a few training epochs with a reduced learning rate. (Optional)

For example, in a three-layer perceptron with layers $L_1$, $L_2$, and $L_3$, suppose that $L_2$ has weights $W_2$. After training, we might find that several weights in $W_2$ have very small magnitudes. We set those weights to zero on the basis of our threshold and then fine-tuned the network to maintain accuracy.

**Structured pruning** Structured pruning [180]–[182] extends the concept of magnitude-based pruning by removing entire neurons, filters, or channels, rather than individual weights. This method can significantly reduce model size and inference time, making it more suitable for hardware implementations. Here's how structured pruning can be applied:

1. **Train the network:** Start with a fully trained three-layer perceptron with ReLU activation.
2. **Identify structures to prune:** Evaluate the importance of neurons or filters. This can be done using criteria such as the average activation of neurons.
3. **Rank neurons:** Rank the neurons in each layer based on their importance metrics.
4. **Prune neurons:** Remove the least important neurons. This involves removing the corresponding weights and biases.

5 · **Retrain the Network (optional):** Fine-tune the pruned network to regain performance. As with magnitude-based pruning, this usually requires a few epochs of training with a reduced learning rate.

In our three-layer perceptron example, if we determine that certain neurons in layer $L_2$ consistently have low activations, we might decide to remove those neurons entirely. This involves not only removing the corresponding weights in $W_2$ but also adjusting the weights that connect $L_2$ to $L_3$ accordingly.

In summary, while magnitude-based pruning is the least complex and easiest to implement, **structured pruning provides more practical benefits, especially in resource-constrained environments**. In the following simulations, we will focus on structured pruning.

#### 8.6.2.2 Quantization overview

Quantization involves **reducing the precision of neural network parameters and activations from floating-point representation to lower-bit representations, such as integers**. This reduction in data-type precision leads to **less computational intensity and less memory usage**. Importantly, quantization can also reduce the latency of operations –by reducing critical path–, a critical factor in real-time signal processing applications such as satellite communications. In our case, both weights and activations are quantized, typically to 8, 4, or even 2 bits, depending on the required precision and resource constraints. –Note that the input signal is generally quantized to 8 to 12 bits and can be noisy or noise-free depending on the specific application requirements.–

#### 8.6.2.3 Importance in real implementations

Pruning and quantization are particularly relevant when deploying models to hardware with limited resources or where power consumption is a concern, such as in satellite or aerospace applications. The benefits include:

- **Resource Efficiency:** Reduced model size translates to lower memory requirements, allowing deployment on devices with limited RAM and storage.
- **Faster Computation:** Fewer computations per inference lead to faster response times, which is crucial for time-sensitive applications.
- **Energy Conservation:** Less computational work per inference means lower energy consumption, which is essential for battery-operated devices in remote locations.

Pruning and quantization not only make the deployment of neural network-based predictors feasible in resource-constrained environments, but also maintain the throughput necessary for real-time signal processing. These optimizations are particularly pertinent given the upcoming discussion on FPGA implementations, where hardware constraints often dictate the feasibility of deploying advanced machine learning models.

### 8.6.3 Implementing neural networks on FPGA

Traditionally, writing FPGA designs in Register Transfer Level (RTL) code is a time-consuming and complex process, requiring deep hardware expertise. However, modern tools and frameworks have emerged to streamline this process and make it more accessible to developers without strong skills in hardware design.

One of the modern key solutions is VITIS AI, a comprehensive development platform provided by Xilinx. VITIS AI integrates with popular machine learning frameworks like TensorFlow and PyTorch, allowing users to deploy optimized neural network models on Xilinx FPGAs. It supports advanced pruning and quantization techniques to optimize models for FPGA deployment, ensuring efficient utilization of resources and power consumption. Similarly, FINN [183] is another Xilinx framework that focuses on the efficient inference of quantized neural networks. By emphasizing the use of extremely low bit-width representations, often down to binary or ternary weights, FINN allows for significant reductions in the computational requirements and memory footprint of neural networks. This approach directly ties into the pruning and quantization strategies discussed previously, as it aims to maintain performance while minimizing resource usage.

Other notable tools include HLS4ML [184] and OpenVINO. HLS4ML, an open source tool for high-level synthesis, translates high-level algorithmic descriptions into FPGA firmware, reducing the barrier to entry for FPGA programming by using languages such as C++ or Python. OpenVINO, developed by Intel, offers a similar capability for FPGAs and other hardware accelerators, including model optimization techniques that perform pruning and quantization to convert neural network models into an optimized intermediate representation. Both tools simplify the deployment process and ensure efficient hardware implementations without requiring developers to write RTL code.

These tools collectively address the complexity of implementing neural networks on FPGAs by abstracting low-level hardware details and enabling high-level design flows. By integrating pruning and quantization techniques, they ensure that neural network models can be efficiently mapped onto FPGA architectures, balancing the trade-offs between performance, resource utilization, and power consumption. This advancement significantly simplifies the development process, allowing rapid prototyping and deployment of high-performance neural networks in resource-constrained environments such as aerospace applications.

### 8.6.4 Results for an FPGA implementation of the shallow network

This section presents a detailed overview of FPGA resource consumption under varying optimization strategies, specifically quantization and pruning rates. The simulations use the **Shallow** model with an initial layer of 128 neurons. **The presented neural networks have been implemented using Xilinx FINN –which appeared as the simplest library to use for prototyping–.** and the parameters have been chosen **in order to perform one inference per clock cycle**. This can seem a bit optimistic, but the goal is to show the potential of the method for compression purpose. Since the target frequency was set to 100 MHz the network could potentially compress at a rate of 100 Msamples/s.

Figure 8.14 illustrates the compression ratio as a function of the pruning factor for two networks trained on QPSK signals with $\text{RRC}_{0.25}$ shaping: **one network utilizes floating-point layers, while the other weights are completely quantized at 8 bits.** The graph shows that the compression ratio remains relatively stable up to a pruning factor of approximately 30%. Beyond this point, the compression ratio begins to decrease more significantly, indicating a trade-off between the pruning factor and the compression ratio. **In particular, 8-bit quantization exhibits a slightly lower compression ratio compared to the floating-point model across most pruning factors, reflecting the subtle**

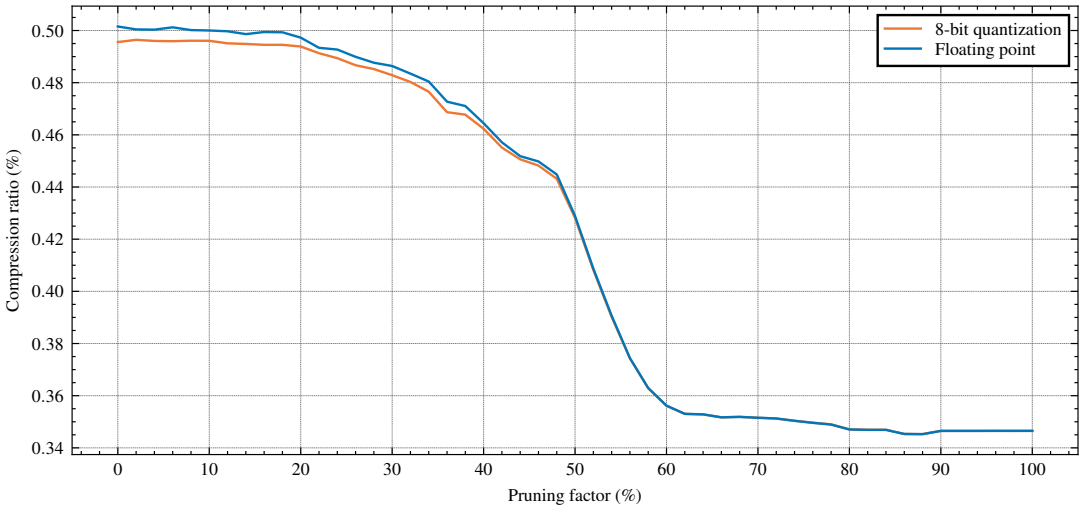



**Figure 8.14:** Evolution of compression ratio depending on *Pruning factor* for two networks trained on QPSK signals with $\text{RRC}_{0.25}$ shaping on **noiseless data**. **Compression ratio includes resampling step from** $\mathcal{F} = 2.00$ **to** $\mathcal{F} = 1.25$

**impact of reduced precision on compression performance.**

The impact of these optimizations on the utilization of different FPGA resources is cataloged in Table 8.3, allowing for a comparative analysis of the efficiency and trade-offs associated with each optimization approach.

The target FPGA for our implementation was the *Xilinx xcku060ffva1156-2*. This choice was made only for simulation purposes. The notation used to distinguish different optimized models is (number of quantization bits, pruning ratio). The target frequency was targeted to 100 MHz in order to compress at a rate of 100 Msamples/s (which is a common rate for satellite communications).

**Table 8.3:** Resources usage on FPGA depending on pruning ratio parameters. **Quantization of network weights is 8 bits** Pruning and implementation has been comped with *Xilinx FINN*.

| **Site Type** | **Used resources** | | | | **Available resources** |
|---|---|---|---|---|---|
| | 0% **pruning** | 20% **pruning** | 50% **pruning** | 80% **pruning** | |
| CLB LUTs* | 202,796 | 171,102 | 138,909 | 106,061 | 331,680 |
| LUT as Logic | 153,075 | 121,417 | 89,188 | 84,278 | 331,680 |
| LUT as Memory | 49,721 | 49,721 | 49,721 | 21,783 | 146,880 |
| LUT as Distributed RAM | 128 | 128 | 128 | 64 | |
| LUT as Shift Register | 49,593 | 49,593 | 49,593 | 21,719 | |
| CLB Registers | 189,053 | 169,821 | 151,139 | 93,359 | 663,360 |
| DSP48E2 | 220 | 220 | 220 | 100 | 2,760 |
| Register as Flip Flop | 189,053 | 169,821 | 151,139 | 93,359 | 663,360 |
| Register as Latch | 0 | 0 | 0 | 0 | 663,360 |
| CARRY8 | 842 | 843 | 846 | 288 | 41,460 |

As observed in Table 8.3, usage of different strategies influence the resource usage in the FPGA. Quantization reduces the number of CLB LUTs and registers required, while higher pruning ratios further reduce these numbers. For example, with 8-bit quantization, moving to the 50% pruning ratio reduces the usage of CLB LUTs from 202,796 to 138,909, demonstrating the effectiveness of pruning in resource optimization.

These findings highlight the balance required between resource utilization and performance when optimizing neural network models for FPGA implementations. The combination of quantization and pruning can lead to substantial resource savings, but careful consideration must be taken to maintain the desired performance levels.

## 8.7 Conclusion and future perspectives

The method described in this manuscript offers a straightforward, reproducible, and agnostic approach to compress waveforms for use in space, aeronautical, and terrestrial communications, without the need for prior knowledge or analysis of their statistical characteristics. It also eliminates the need for costly implementation of a parameter estimation stage, as is often required when using LPC, albeit at the expense of a significant increase in the number of parameters.

In addition to the examples provided in this document, the wide-ranging applications of this method can be further extended to include the compression of multiple streams simultaneously, potentially taking advantage of any cross-correlation between streams that might be automatically computed by the network.

Regarding the usage of noisy signals, the high sensitivity of neural networks to noise seems to limit their use in RX signals (Section 8.5), where the SNR can typically go below 0dB. However, the application of well-thought-out cutting methods in the noise range can actually achieve a good compression ratio even in noisy environments [165].

Implementing neural networks on hardware devices is typically challenging, and ways to implement them efficiently (considering the effects of weight quantization and node pruning) are still the subject of many investigations. However, the network architectures presented in this chapter remain simple compared to modern prediction networks, which use transformer cells [185].

For these reasons, further investigations could be conducted to identify how the number of parameters in the neural network could be reduced while maintaining a satisfactory compression ratio.

Chapter 9

# Conclusion

***Beyond Journey's End***


9.1 Restatement of research objectives . . . . . **157**
9.1.1 Advancements in lossless compression . . . . . 157
9.1.2 Advancements in lossy compression . . . . . 158
9.1.3 Results on practical case . . . . . 159
9.2 Study limitations . . . . . **160**
9.3 Recommendations for future research . . . . . **161**
9.3.1 Broader signal types . . . . . 161
9.3.2 Hardware implementations . . . . . 161
9.3.3 Algorithm optimization . . . . . 161
9.3.4 Cross-layer optimization . . . . . 161
9.3.5 Adaptive and intelligent compression techniques . . . . . 162
9.3.6 Extending techniques beyond compression . . . . . 162
9.4 Going further: new and emerging methods . . . . . **163**
9.4.1 Compression based on autoencoder networks . . . . . 163
9.4.2 Generative model-based compression . . . . . 164
9.4.3 Vector quantization via Neural Networks . . . . . 164
9.5 Overall conclusion & final thoughts . . . . . **169**


This chapter encapsulates the essential findings and insights gleaned throughout this investigation. It synthesizes the key contributions and highlights the broader implications of the work conducted. By reflecting on the achievements and limitations of the study, this conclusion aims to provide a comprehensive understanding of the advancements made and their importance in the context of satellite communication systems. In addition, it outlines potential avenues for future research, emphasizing the dynamic nature of this field and the continuous pursuit of innovation.

## 9.1 Restatement of research objectives

In this manuscript, our objective was to investigate efficient methods to compress **baseband telecommunication signals within digital and satellite communication systems**. The primary research questions were as follows:

- How can signal compression improve bandwidth efficiency in satellite communications?
- What are the most effective techniques for compressing baseband signals?

The findings of our research indicate that substantial bandwidth savings can be realized using signal compression methods. Important results encompass the examination of both *lossy* and *lossless* compression techniques, with varying complexity levels.

### 9.1.1 Advancements in lossless compression

Regarding lossless compression, the thesis began with a survey of existing techniques, particularly focusing on Linear Prediction Coding (LPC) and algorithms derived from the *Lempel-Ziv (LZ)*

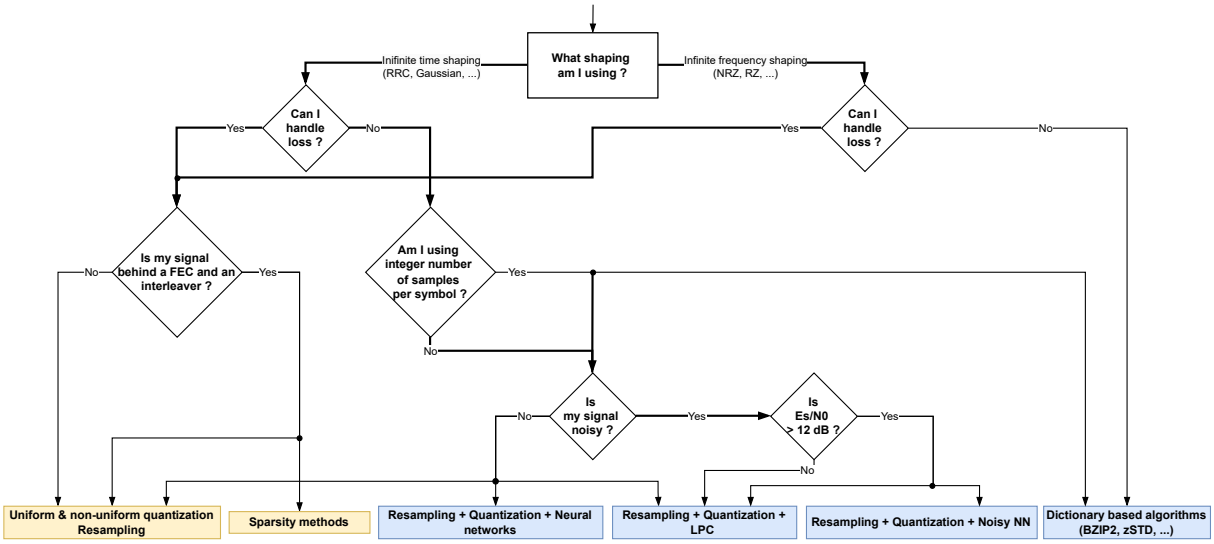



**Figure 9.1:** A streamlined decision tree employed to select the appropriate algorithm family for signal compression. Larger version of this schematic is available in Appendix B.

*family*, such as *BZip2* and *zSTD*. Our research made several key contributions to the field.

**Categorization of algorithms:** We systematically categorized the usage and persistence of various classes of lossless compression algorithms depending on waveform characteristics. This categorization provides a clear guide for selecting the most appropriate compression method based on the signal properties.

**Limitations of LPC in space communications:** A thorough investigation into the limitations of LPC was conducted, especially in scenarios where the signal is downsampled before compression. This analysis is crucial for space communications, where bandwidth and processing power are limited.

**Integration of neural networks:** To enhance the prediction capabilities of LPC, neural networks were incorporated to perform nonlinear prediction. This hybrid approach was evaluated in both noisy and noiseless environments, resulting in significant improvements in prediction accuracy. Quantitative evaluations showed that neural network-enhanced predictive coding outperforms traditional LPC in terms of compression efficiency and noise resistance. The performance and training methods of these networks were rigorously analyzed, providing insight into their practical implementation in noisy communication channels.

### 9.1.2 Advancements in lossy compression

The study on lossy compression addressed several basic and advanced aspects to optimize signal degradation and compression efficiency:

**Metric definition and prerequisites:** We defined a comprehensive set of metrics to quantify signal degradation and established the prerequisites for effective lossy compression. **This included highlighting the importance of Forward Error Correction (FEC) and interleaving in maintaining the integrity of the signal after compression.**

**Quantization and BER degradation:** The range of usage for traditional uniform quantization was categorized and the associated degradation of the Bit-error rate (BER) was meticulously calculated for

all DVB-S2 modcods. This helped in understanding the trade-offs between compression efficiency and signal quality.

**Resampling techniques:** A detailed examination of resampling methods was conducted, providing a clear analysis of their impact on BER and overall signal integrity. The study offered guidelines on selecting the appropriate resampling techniques based on the specific requirements of the communication system.

**Sparse coding for controlled degradation:** A novel method based on sparse coding was introduced, which allows the selection of desired levels of signal degradation while achieving efficient compression. This approach takes advantage of the sparsity of signal representation to minimize data loss and optimize compression ratios. Simulations and practical implementations demonstrated the method's effectiveness, showing significant improvements over traditional lossy compression techniques.

These advances in lossless and lossy compression contribute to the field of telecommunications by providing robust, efficient, and adaptable methods suitable for a variety of signal processing scenarios, particularly in challenging environments like space communications.

### 9.1.3 Results on practical case

Table 9.1 summarizes the results obtained for the various algorithms presented in this manuscript. The input signal used in the experiments is an 8-PSK 3/4 modulation (DVB-S2) with 2 samples per symbol, quantized to 8 bits. This results in an expansion factor of 14.22 bits without any compression applied. The table provides the expansion factor, compression ratio, degradation in dB, and the main bottleneck encountered for each algorithm.

| | **Expansion factor** | **Compression ratio** | **Degradation (dB)** | **Bottleneck** |
|---|---|---|---|---|
| **Lossless (TX)** | | | | |
| **Raw** | 14.22 | 0.00% | 0 | |
| **LPC** | 8.64 | 39.23% | 0 | Decompressor |
| **Shallow prediction** | 5.91 | 58.41% | 0 | Decompressor |
| **Lossless (RX, Es/N0 = 15dB)** | | | | |
| **Raw** | 14.22 | 0.00% | 0 | |
| **LPC** | 7.11 | 50.00% | 0 | Decompressor |
| **Shallow prediction** | 6.54 | 54.00% | 0 | Decompressor |
| **Lossy** | | | | |
| **Quantization** | 5.26 | 63.00% | 1 | Both |
| **Resampling** | 7.11 | 50.00% | 1 | Decompressor |
| **Resampling + LPC + Rice** | 4.27 | 70.00% | 1 | Decompressor |
| **Sparse coding (Huffman)** | 3.13 | 78.00% | 1 | Compressor |
| **Sparse coding (Rice)** | 3.41 | 76.00% | 1 | Compressor |

**Table 9.1:** Results of compression for the different compression algorithms presented, implemented and tested in this manuscript.

## 9.2 Study limitations

Despite the promising results, this research has certain limitations that must be acknowledged. These limitations may have influenced the generalizability of the findings across different communication systems.

First, hardware constraints were not fully addressed in the current implementation. The algorithms were tested in simulated environments on CPUs, but real-world hardware limitations, such as those encountered on FPGAs and GPUs, were not considered. The implementation of algorithms proposed on these platforms can present significant challenges, as all algorithms cannot be easily adapted to these devices. This gap highlights the need for further research to explore hardware-specific optimizations and adaptations.

Furthermore, the complexity of some advanced compression algorithms requires substantial computational resources, which can limit their practical applicability. This computational demand may hinder the implementation of these algorithms in real-time systems, where processing speed and efficiency are critical.

In addition, the communication chain modeled in this research is not complete. The most advanced simulations included Forward Error Correction (FEC), interleaving, phase error, and the addition of Additive White Gaussian Noise (AWGN). However, several essential elements were not considered:

- **Synchronization:** The synchronization stages, primarily composed of Phase-Locked Loops (PLL) and other loops, introduce induced noise into the signal. These loops must converge for the system to function correctly, and the degradation induced by compression must be evaluated considering these loops to ensure reliable convergence.
- **Time-selective and frequency-selective channels:** The study considered channels as constant over time. In reality, channels vary over time and frequency, altering the parameters of the communication environment. Future work should reflect these dynamic changes in algorithmic parameters to improve performance and reliability.
- **Channel types:** Only direct-view channels were considered in the simulations. However, real-world channels often include reflections (Ricean channels) or may lack a direct line of sight (Rayleigh channels). These variations can significantly affect the performance of the compression algorithms and should be investigated in future studies.

Addressing these limitations in future research will improve the practical applicability and robustness of the proposed methods, ensuring that they can be effectively implemented in a wider range of real-world communication systems.

## 9.3 Recommendations for future research

Based on the findings and limitations identified in this thesis, several avenues for future research are recommended to further advance the field of signal compression in telecommunications. These recommendations aim to address current challenges and explore new frontiers that could enhance the efficiency and applicability of compression techniques in both terrestrial and aerospace environments.

### 9.3.1 Broader signal types

One significant direction for future research is the expansion of the study to include a wider variety of signal types. Although this thesis primarily focused on baseband signals, incorporating different types of signal, such as wideband and narrowband signals, would provide a more comprehensive understanding of compression techniques across diverse communication environments. This expansion could involve testing in various scenarios, including different frequency bands, modulation schemes, and environmental conditions (e.g., urban, rural, aerospace settings, Rice or Rayleigh channel, ...). Furthermore, investigating the performance of compression algorithms on signals with varying levels of noise and interference would be beneficial in understanding their robustness and reliability.

### 9.3.2 Hardware implementations

Another critical area for future research is the development and testing of hardware-based implementations of the proposed compression algorithms. Although software simulations provide valuable information, real-world applications often face unique challenges that can only be identified through hardware testing. Future studies should focus on designing and prototyping hardware architectures, such as Field-Programmable Gate Array (FPGA) or Application-Specific Integrated Circuit (ASIC) implementations, to evaluate the practical performance, power consumption and feasibility of these compression techniques. This research could also explore the integration of these hardware solutions into existing communication systems and assess their impact on overall system performance.

### 9.3.3 Algorithm optimization

Optimizing the computational efficiency of advanced compression algorithms is essential for their practical application in real-time systems. Future research should aim to enhance the processing speed and reduce the computational complexity of these algorithms without compromising their compression performance. Techniques such as parallel processing, algorithmic simplifications, and the use of machine learning to tune adaptive parameters could be explored. **Furthermore, investigating the trade-offs between compression ratio, latency, and computational resources would provide a clearer understanding of how to balance these factors in different application scenarios.**

### 9.3.4 Cross-layer optimization

Investigating cross-layer optimization strategies could significantly enhance the performance of compression algorithms in practical communication systems. This approach involves considering the interactions between different layers of the communication protocol stack, such as the physical layer, data link layer, and network layer, to optimize the overall performance of the system. Future research could explore how compression techniques can be adapted or integrated with synchronization stages, modulation schemes, and network protocols to achieve better efficiency and reliability. **This holistic view could lead to more robust and adaptive communication systems capable of handling a wide range of operational conditions.**

### 9.3.5 Adaptive and intelligent compression techniques

The development of intelligent and adaptive compression techniques is a promising area for future research. These techniques would leverage advances in artificial intelligence and machine learning to **dynamically adjust compression parameters based on real-time analysis of signal characteristics and the communication environment**. For example, neural network-based predictors and adaptive quantization schemes could be employed to continuously optimize compression performance. Research in this area could focus on designing algorithms that learn and evolve over time, improving their efficiency and adaptability to varying conditions.

### 9.3.6 Extending techniques beyond compression

Although the primary focus of this thesis has been signal compression, the techniques and tools developed during this research have potential applications beyond their original purpose. One promising direction for future research is to explore how these methods can be adapted for use in other stages of the communication process, such as modulation/demodulation and synchronization.

For instance, the optimization algorithms and data processing techniques designed for compression could be repurposed to enhance modulation schemes. By applying these methods, it may be possible to achieve more efficient modulation processes, reducing bandwidth requirements, and improving signal clarity. **Similarly, the advanced signal analysis tools used in compression could be used to refine demodulation algorithms, potentially increasing their accuracy and resilience in noisy environments.**

Another area worth investigating is the **application of these techniques in synchronization processes**. Whether for blind synchronization or data-aided synchronization, compression methods could help streamline the synchronization stages, reducing overhead, and improving the speed and accuracy of synchronization. This could be particularly beneficial in challenging conditions with high interference or varying signal strengths.

By extending the use of these compression techniques and tools to other aspects of communication systems, future research could uncover new efficiencies and performance improvements, demonstrating the versatility and broader impact of these innovations.

---

In conclusion, these recommendations for future research highlight the potential areas where significant advances can be made to further enhance the field of signal compression in telecommunications. **By addressing these directions, future studies can contribute to the development of more efficient, robust, and versatile compression techniques that meet the evolving demands of modern communication systems.**

## 9.4 Going further: new and emerging methods

In this section, we explore the frontier of advancement that lies beyond the scope of this thesis. Although the primary focus of our research has been the establishment of methodologies within the realm of telecommunications, embedded electronics, and digital communications, it is essential to acknowledge the continuous evolution of this field. **Due to time constraints and the rapid pace of technological development, several innovative techniques and approaches have emerged or gained prominence during the course of this study.**

We will discuss the potential of these cutting-edge methods to address existing challenges and enhance the capabilities of current technologies. This exploration aims to provide a broader perspective on future research directions and the continuous improvement of systems in both terrestrial and aerospace environments. By examining these new and emerging methods, we can appreciate the dynamic nature of our field and the importance of staying abreast of the latest advances to foster ongoing innovation and development.

### 9.4.1 Compression based on autoencoder networks

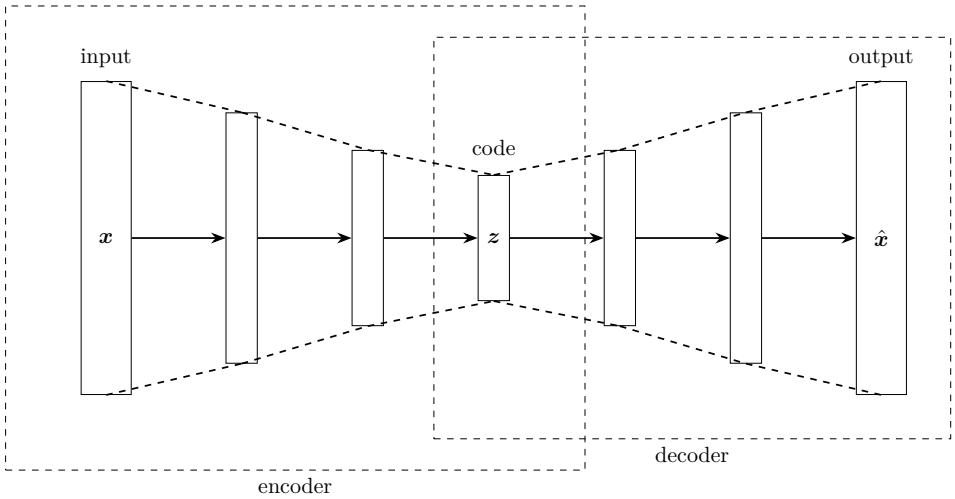



**Figure 9.2:** Structure of an autoencoder network.

An autoencoder is a type of Neural Network (NN) that is trained to compress or (*encode*) the input into a lower-dimensional representation and then reconstruct (*decode*) it back to the original input as closely as possible. The basic structure of an autoencoder can be represented by the following equations:

$$\mathbf{z} = f_{\theta_e}(\mathbf{x}) \tag{9.1}$$

$$\hat{\mathbf{x}} = f_{\theta_d}(\mathbf{z}) \tag{9.2}$$

where $\mathbf{x}$ is the input signal, $\mathbf{z}$ is the compressed representation, $\hat{\mathbf{x}}$ is the reconstructed signal, $f_{\theta_e}$ is the encoding function with parameters $\theta_e$, and $f_{\theta_d}$ is the decoding function with parameters $\theta_d$.

In practical applications, this architecture can easily describe a compression system by itself: the output of the encoder $\mathbf{z}$, which represents the compressed data, would be transmitted over the network. Once received, the compressed data would be fed into the decoder to reconstruct the original signal.

The performance of an autoencoder **is often measured by its ability to minimize the reconstruction error**, which is typically a function of the difference between the input and output signals. The Mean Squared Error (MSE) is a common choice for this:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\mathbf{x}}_i)^2 \tag{9.3}$$

### 9.4.2 Generative model-based compression

Generative models such as Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) have also been used for signal compression. GANs use a two-network architecture comprising a generator and discriminator to produce high-fidelity signal representations. VAEs, on the other hand, use a probabilistic approach to generate a distribution that models the data, which can then be sampled to compress and decompress the signals. The VAE can be expressed mathematically as

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} \tag{9.4}$$

where $p_\theta(\mathbf{x}|\mathbf{z})$ is the probability of the signal $\mathbf{x}$ given the latent representation $\mathbf{z}$, and $p(\mathbf{z})$ is the prior distribution over the latent variables.

### 9.4.3 Vector quantization via Neural Networks

#### 9.4.3.1 Overview & emerging methods

As described in Chapter 3, Vector Quantization (VQ) is a classical signal compression technique which works by mapping vectors from a high-dimensional space to a finite set of vectors in the same space, known as codevectors, which form a codebook. VQ has since been adapted to neural network architectures for improved performance. The mapping is done in such a way that the average distortion between the input vectors and their corresponding codevectors is minimized. Mathematically, this is expressed as follows:

$$\hat{\mathbf{x}} = \underset{\mathbf{c}\in\mathbf{C}}{\text{argmin}} \|\mathbf{x} - \mathbf{c}\|^2 \tag{9.5}$$

where $\mathbf{C}$ represents the codebook and $\mathbf{c}$ represents a codevector in $\mathbf{C}$.

In the context of neural networks, **VQ can be implemented using a trained encoder that performs the mapping to the codebook**. The encoder is often coupled with a decoder that attempts to reconstruct the input from the quantized output, thus forming an end-to-end trainable system.

In today's literature, multiple quantization methods have been developed. Here are the main ones:

**Residual Vector Quantization** RVQ [186] extends VQ by iteratively quantizing the residual error after the initial quantization step. This method can be represented in a neural network framework as a series of auto-encoders where each autoencoder encodes the residual errors of the previous one.

$$\mathbf{e}_0 = \mathbf{x} \tag{9.6}$$
$$\mathbf{z}_i = \mathrm{VQ}(\mathbf{e}_{i-1}; \mathbf{C}_i), \quad i = 1, 2, \ldots, k \tag{9.7}$$
$$\mathbf{e}_i = \mathbf{e}_{i-1} - \mathrm{Dec}(\mathbf{z}_i), \quad i = 1, 2, \ldots, k \tag{9.8}$$
$$\hat{\mathbf{x}} = \sum_{i=1}^{k} \mathrm{Dec}(\mathbf{z}_i) \tag{9.9}$$

where $\mathbf{e}_i$ is the residual error after the $i$-th quantization, $\mathrm{VQ}(\cdot; \mathbf{C}_i)$ is the vector quantization operation with codebook $\mathbf{C}_i$, $\mathbf{z}_i$ is the quantized output at stage $i$, $k$ is the number of quantization stages and $\mathrm{Dec}(\cdot)$ is the corresponding decoding operation. RVQ improves compression by exploiting signal structure and allowing trade-off control between complexity and performance through stage adjustment. However, it has increased computational complexity and challenges in multistage training.

**Random Projection Quantizer** (RPQ) [188] utilizes random projections to reduce the dimension of the signal prior to quantization. This technique takes advantage of the properties of high-dimensional geometry and random matrices. The process can be described as follows:

$$\mathbf{u} = \mathbf{R}\mathbf{x} \tag{9.12}$$
$$\mathbf{z} = \mathrm{Quantize}(\mathbf{u}; \mathbf{C}) \tag{9.13}$$

where $\mathbf{R}$ is a random projection matrix, $\mathbf{u}$ is the projected vector, and $\mathbf{z}$ is the quantized output.
RPQ is beneficial for its dimensionality reduction properties and the ability to preserve the distances between data points, making it suitable for applications in compressive sensing and machine learning. However, the randomness of the projection can sometimes lead to variability in performance.

**Finite Scalar Quantization** Developed by Google DeepMind, Finite Scalar Quantization (FSQ) [187] involves quantizing a signal by mapping it to the nearest value in a finite set of scalar levels. This method is characterized by its simplicity and efficiency in implementation. The process can be expressed mathematically as follows.

$$\mathbf{y} = \mathrm{FSQ}(\mathbf{x}; \mathbf{L}) \tag{9.10}$$
$$\mathbf{q}_i = \min_{l \in \mathbf{L}} |x_i - l|, \quad i = 1, 2, \ldots, n \tag{9.11}$$

where $\mathbf{y}$ is the quantized output, $\mathbf{L}$ is the set of scalar levels, and $\mathbf{q}_i$ is the quantization index for the $i$-th element of the input vector $\mathbf{x}$.
FSQ is particularly advantageous for its low computational requirements and ease of hardware implementation. However, it might not take advantage of the full structure of the signal, leading to suboptimal compression performance in some cases.

**Latent Quantization** (LQ) [189] involves learning a latent representation of the data that is then quantized. This approach is typically used in neural network frameworks, where the latent representation is derived from an encoder. The process is given by:

$$\mathbf{h} = \mathrm{Enc}(\mathbf{x}) \tag{9.14}$$
$$\mathbf{z} = \mathrm{Quantize}(\mathbf{h}; \mathbf{C}) \tag{9.15}$$
$$\hat{\mathbf{x}} = \mathrm{Dec}(\mathbf{z}) \tag{9.16}$$

where $\mathrm{Enc}(\cdot)$ and $\mathrm{Dec}(\cdot)$ are the encoder and decoder functions, respectively, $\mathbf{h}$ is the latent representation and $\mathbf{z}$ is the quantized latent code.
LQ benefits from the ability to learn compact and informative representations, leading to efficient compression. However, the training process can be complex and computationally intensive, especially for deep neural networks.

#### 9.4.3.2 Specific compression methods using neural networks

This section introduces several renowned architectures utilizing Neural Vector Quantization, which would serve as a solid foundation for those looking to delve deeper into this topic.

**SoundStream** SoundStream [186], proposed by Zeghidour et al. in 2021, is a novel end-to-end neural audio codec capable of effectively compressing speech, music, and general audio at bitrates ranging from 3 kbps to 18 kbps. The codec consists of a fully convolutional encoder-decoder network with a novel residual vector quantizer. The encoder processes time-domain waveforms into embeddings that are quantized and then reconstructed by the decoder. This setup allows SoundStream to operate in real time on resource-limited devices such as smartphones, supporting streamable inference without compromising the latency.

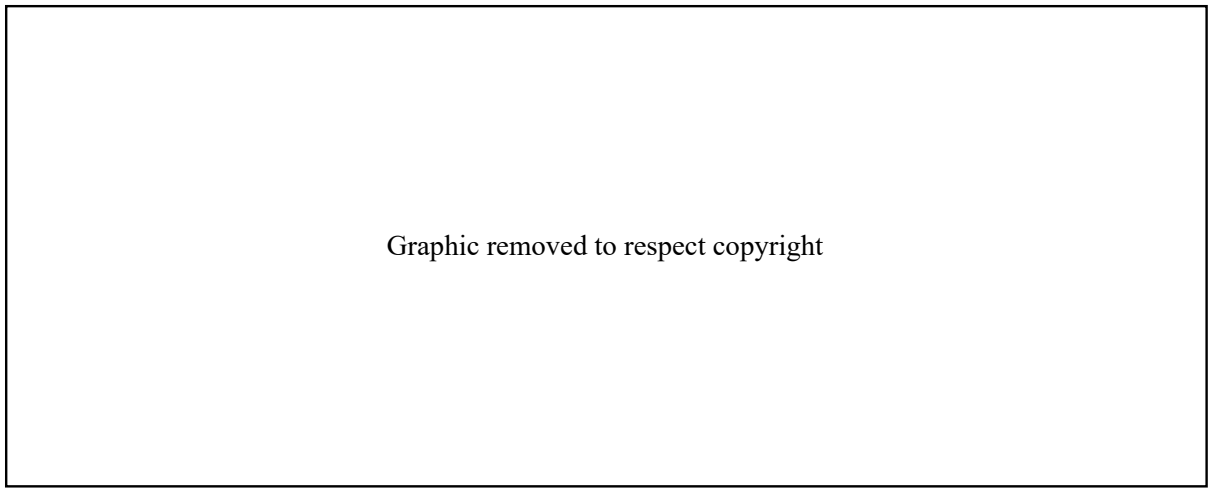



**Figure 9.3:** Architecture of SoundStream
**Source:** *SoundStream Github Project [190]*

Training incorporates both adversarial and reconstruction losses to improve the perceptual quality of audio. The system can also perform joint compression and enhancement, which proves beneficial for applications such as denoising without additional latency. This makes SoundStream particularly adept at handling a variety of audio inputs and maintaining high fidelity under various conditions.

SoundStream introduces several advancements in the domain of quantization. The primary innovation is the use of a residual vector quantizer (RVQ). Traditional quantization methods can struggle with the efficient representation of high-dimensional data, often leading to significant quality losses. However, RVQ in **SoundStream sequentially quantizes the residuals of the signal**, effectively reducing the error at each step and leading to higher fidelity in the reconstructed audio. This process enables finer quantization granularity without a substantial increase in bitrate. Moreover, the hierarchical structure of the RVQ allows SoundStream to **dynamically allocate bits according to the complexity of the audio signal**. Simpler signals require fewer bits, thus optimizing the overall bitrate usage. This dynamic allocation is particularly beneficial for maintaining consistent audio quality across different types of content, from simple speech to complex musical compositions.

Another critical aspect of SoundStream's quantization approach is its ability to leverage neural network-based learning for optimization. By training the quantizer end-to-end along with the encoder and decoder, SoundStream achieves a more integrated and efficient compression pipeline. **This contrasts with traditional methods, where quantization is often a separate, less flexible component.**

**Encodec** Following the development of SoundStream, Encodec by Défossez et al. [191] the technology specifically for high-fidelity real-time audio compression. Encodec builds on the architectural principles of SoundStream, but introduces enhancements tailored to streaming environments. It utilizes a convolutional encoder-decoder system with a novel approach to vector quantization that manages the quantized latent space dynamically during live audio streaming.

The significant improvement of Encodec lies in its adaptive handling of different audio signals and its optimized performance under varying network conditions, which is crucial for mobile and streaming applications. The model architecture supports low-latency operation essential for real-time applications and incorporates a unique loss balancer in training to stabilize the learning of its network, ensuring efficient and high-quality audio compression across a wide range of bitrates.

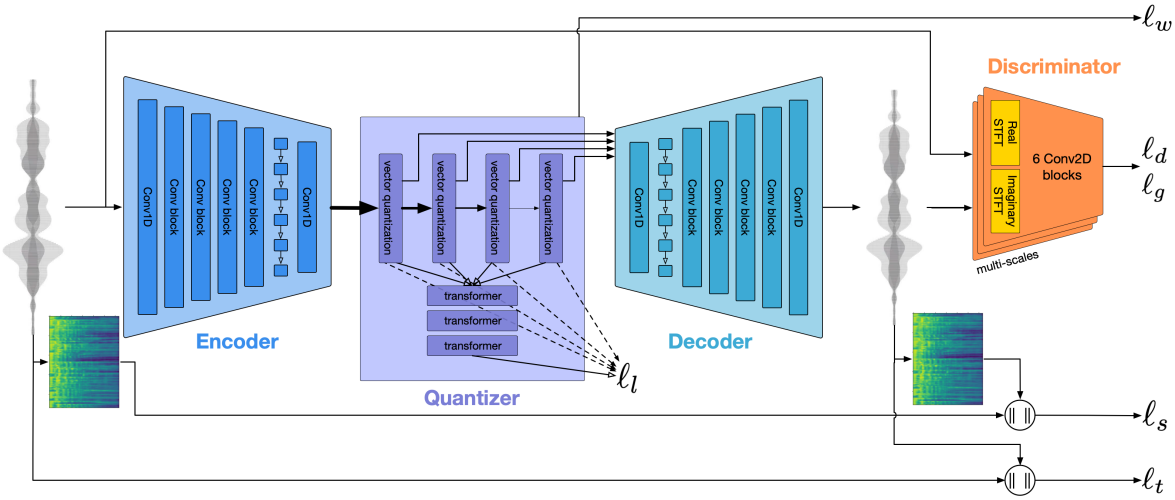



**Figure 9.4:** Architecture of Encodec
**Source:** *Encodec Github Project [191], MIT licensed*

The Encodec and SoundStream architectures provide a robust framework. These neural network-based models, which efficiently handle various audio signals at different bitrates and conditions, offer valuable information for developing advanced audio compression techniques. By analyzing the fully convolutional encoder-decoder structure and the innovative use of residual vector quantizers in SoundStream, alongside Encodec's dynamic adaptation strategies for real-time streaming, significant guidance can be drawn for crafting a compressor that not only optimizes data rate reduction but also maintains high fidelity and operational efficiency in diverse environments.

## 9.5 Overall conclusion & final thoughts

This thesis has contributed to the field of communication systems and *RF over IP* by demonstrating the importance of advanced signal compression techniques, both lossless and lossy, in improving bandwidth efficiency and maintaining data integrity. The exploration and application of these techniques have been meticulously documented, offering **the missing comprehensive overview** of current methods, from well-established approaches to cutting-edge innovations. Although the research presented in this thesis covers a variety of promising technologies, it is important to note that **not all potential solutions have been investigated**. The inherent biases in this work stem from the necessary choices made during the research process, including the selection of algorithms, the design of the testing environment, the simulator used, the simulation parameters and the waveforms tested. **Such biases are an integral part of scientific research, reflecting the practical constraints and decisions involved in the implementation of complex systems.**

Despite the promising results, the algorithms and techniques discussed in this thesis require further fine-tuning and testing under real-world conditions. The rapid pace of technological advancement in this field means that new methods and frameworks are continually being developed, leaving ample scope for ongoing and future research. The findings of this thesis not only provide a solid foundation for current applications but also highlight the potential for multiple future theses to build upon this work.

> In my sense, this thesis addresses the primary research question by providing a detailed list of state-of-the-art techniques and algorithms, guidelines for selecting the most appropriate method under specific conditions, and a thorough examination of their limitations.

However, it is essential to acknowledge that signal compression is not the sole method to optimize bandwidth in ground systems. **Other strategies, such as network optimization, should also be considered for a more holistic approach.**

In summary, this thesis has advanced our understanding and application of signal compression techniques in satellite communication systems. The contributions include a thorough evaluation of various compression methods, an analysis of their performance under different conditions, and a clear articulation of their respective strengths and limitations. The work underscores the importance of continuous research and development in this dynamic field, paving the way for future innovations and improvements.

"*Do not go where the path may lead, go instead where there is no path and leave a trail.*"

Ralph Waldo Emerson

# BIBLIOGRAPHY


[1] C. Shannon, « Communication in the Presence of Noise », *Proceedings of the IRE*, vol. 37, *1*, pp. 10–21, Jan. 1949, ISSN: 2162-6634. DOI: 10.1109/JRPROC.1949.232969.

[2] J. Proakis, *Digital Communications* (McGraw-Hill Series in Electrical and Computer Engineering), 4th ed. McGraw-Hill, 2000, ISBN: 978-0-07-232111-1.

[3] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing* (Always Learning), Fourth edition, Pearson new international edition. Harlow: Pearson Education Limited, 2014, 1013 pp., ISBN: 978-1-292-02573-5.

[4] M. Rice, *Digital Communications: A Discrete-Time Approach*, 1st ed. Prentice Hall, 2008, ISBN: 978-0-13-030497-1.

[5] *1914.3-2018 - IEEE Standard for Radio over Ethernet Encapsulations and Mappings*. S.l.: IEEE, 2018, 1 p., ISBN: 978-1-5044-5038-6.

[6] *VITA Radio Transport (VRT) Standard*, ANSI/VITA 49.0-2015, Approved ANSI Standard, version ANSI/VITA 49.0-2015, American National Standards Institute, Inc., Aug. 2015, 184 pp.

[7] CPRI, *Common Public Radio Interface (CPRI); Interface Specification*, Interface Specification, version 6.0, Aug. 30, 2013, 128 pp.

[8] CPRI, *Common Public Radio Interface: eCPRI Interface Specification*, Interface Specification, version 2.0, May 10, 2019, 109 pp.

[9] Chih-Lin I and Sachin Katti. « O-RAN ALLIANCE Introduces 48 New Specifications Released Since July 2021 », O-RAN Alliance. (Aug. 2022), [Online]. Available: https://www.o-ran.org/blog/o-ran-alliance-introduces-48-new-specifications-released-since-july-2021 (visited on 08/22/2022).

[10] O-RAN ALLIANCE, « O-RAN Working Group 4 (Open Fronthaul Interfaces WG) Control, User and Synchronization Plane Specification », O-RAN ALLIANCE, Alfter, Technical specification 4, 2021, p. 241.

[11] J. Iyengar and M. Thomson, « QUIC: A UDP-Based Multiplexed and Secure Transport », Internet Engineering Task Force, Request for Comments RFC 9000, May 2021, 151 pp. DOI: 10.17487/RFC9000.

[12] « The Road to QUIC », The Cloudflare Blog. (Jul. 26, 2018), [Online]. Available: https://blog.cloudflare.com/the-road-to-quic (visited on 02/14/2024).

[13] S. N. Io, « Cortex NuRoN - Virtualized Software Modem for satellite Command & Control », Marketing flyer (France).

[14] A. Thomas, J.-M. Leveau, and S. Augarde, « Study of a full software CCSDS receiver », *in 2018 SpaceOps Conference*, ser. SpaceOps Conferences, American Institute of Aeronautics and Astronautics, May 25, 2018. DOI: 10.2514/6.2018-2454.

[15] J. Bartelt, P. Rost, D. Wubben, J. Lessmann, B. Melis, and G. Fettweis, « Fronthaul and backhaul requirements of flexibly centralized radio access networks », *IEEE Wireless Communications*, vol. 22, *5*, pp. 105–111, Oct. 2015, ISSN: 1536-1284. DOI: 10.1109/MWC.2015.7306544.

[16] L. Ramalho, M. N. Fonseca, A. Klautau, *et al.*, « An LPC-Based Fronthaul Compression Scheme », *IEEE Communications Letters*, vol. 21, *2*, pp. 318–321, Feb. 2017, ISSN: 1089-7798. DOI: 10.1109/LCOMM.2016.2624296.

[17] G. Chen, F. Yang, K. Niu, and C. Dong, « A linear predictive coding based compression algorithm for fronthaul link in C-RAN », *in 2017 IEEE/CIC International Conference on Communications in China (ICCC)*, Qingdao, China: IEEE, Oct. 2017, pp. 1–6, ISBN: 978-1-5386-4502-4. DOI: 10.1109/ICCChina.2017.8330421.

[18] Libre Space Foundation. « IQzip: A library to facilitate the creation of signal databases », Gitlab. (2019), [Online]. Available: https://gitlab.com/librespacefoundation/sdrmakerspace/iqzip (visited on 01/26/2022).

[19] S. Nanba and A. Agata, « A new IQ data compression scheme for front-haul link in Centralized RAN », *in 2013 IEEE 24th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC Workshops)*, London, United Kingdom: IEEE, Sep. 2013, pp. 210–214, ISBN: 978-1-4799-0122-7. DOI: 10.1109/PIMRCW.2013.6707866.

[20] V. L. Do, T. B. Nguyen, V. K. Dao, and C. H. Nguyen, « Compression of wideband IQ samples using linear predictive coding and frame of reference », *in 2017 IEEE 2nd International Conference on Signal and Image Processing (ICSIP)*, Singapore: IEEE, Aug. 2017, pp. 371–375, ISBN: 978-1-5386-0969-9. DOI: 10.1109/SIPROCESS.2017.8124567.

[21] L. Ramalho, I. Freire, C. Lu, M. Berg, and A. Klautau, « Improved LPC-Based Fronthaul Compression With High Rate Adaptation Resolution », *IEEE Communications Letters*, vol. 22, *3*, pp. 458–461, Mar. 2018, ISSN: 1089-7798. DOI: 10.1109/LCOMM.2018.2791407.

[22] L. Ramalho, J. Fortuna, C. Lu, M. Berg, I. Almeida, and A. Klautau, « Rate Control for Entropy-Coded LPC: Application to Packet-Based Fronthauling », *IEEE Transactions on Communications*, pp. 1–1, 2022, ISSN: 0090-6778, 1558-0857. DOI: 10.1109/TCOMM.2022.3188827.

[23] A. Shehata, M. Crussière, and P. Mary, « Analysis of Baseband IQ Data Compression Methods for Centralized RAN », *in 2020 28th European Signal Processing Conference (EUSIPCO)*, Amsterdam, Netherlands: IEEE, Jan. 24, 2021, pp. 1762–1766, ISBN: 978-90-827970-5-3. DOI: 10.23919/Eusipco47968.2020.9287760.

[24] R. M. Gray and D. L. Neuhoff, « Quantization », *IEEE Transactions on Information Theory*, vol. 44, *6*, pp. 2325–2383, Oct. 1998, ISSN: 1557-9654. DOI: 10.1109/18.720541.

[25] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Springer US, 1992, ISBN: 978-1-4615-3626-0. DOI: 10.1007/978-1-4615-3626-0.

[26] C. E. Shannon, « A mathematical theory of communication », *The Bell System Technical Journal*, vol. 27, *3*, pp. 379–423, Jul. 1948, ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[27] J. Mairal. « SPAMS ». (), [Online]. Available: https://thoth.inrialpes.fr/people/mairal/spams/ (visited on 10/28/2022).

[28] G. Yoshimura, A. Kanemura, and H. Asoh, « Reconstructable and Interpretable Representations for Time Series with Time-Skip Sparse Dictionary Learning », *in Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, ser. Thematic Workshops '17, New York, NY, USA: Association for Computing Machinery, Oct. 23, 2017, pp. 323–331, ISBN: 978-1-4503-5416-5. DOI: 10.1145/3126686.3126724.

[29] S. K. Lam, A. Pitrou, and S. Seibert, « Numba: a LLVM-based Python JIT compiler », *in Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, ser. LLVM '15, New York, NY, USA: Association for Computing Machinery, Nov. 15, 2015, pp. 1–6, ISBN: 978-1-4503-4005-2. DOI: 10.1145/2833157.2833162.

[30] A. Paszke, S. Gross, F. Massa, *et al.* « PyTorch: An Imperative Style, High-Performance Deep Learning Library ». arXiv: 1912.01703 [cs, stat]. (Dec. 3, 2019), pre-published.

[31] D. A. Huffman, « A Method for the Construction of Minimum-Redundancy Codes », *Proceedings of the IRE*, vol. 40, *9*, pp. 1098–1101, Sep. 1952, ISSN: 2162-6634. DOI: 10.1109/JRPROC.1952.273898.

[32] J. Rissanen and G. G. Langdon, « Arithmetic Coding », *IBM Journal of Research and Development*, vol. 23, *2*, pp. 149–162, Mar. 1979, ISSN: 0018-8646. DOI: 10.1147/rd.232.0149.

[33] A. Shahbahrami, R. Bahrampour, M. S. Rostami, and Mostafa Ayoubi, « Evaluation of Huffman and Arithmetic Algorithms for Multimedia Compression Standards », *International Journal of Computer Science, Engineering and Applications*, vol. 1, *4*, pp. 34–47, Aug. 30, 2011, ISSN: 22310088. DOI: 10.5121/ijcsea.2011.1404.

[34] V. Levenshtein, « On the redundancy and delay of separable codes for the natural numbers », *Problems of Cybernetics*, vol. 20, pp. 173–179, 1968.

[35] D. Salomon, *Variable-Length Codes for Data Compression*. London: Springer, 2007, ISBN: 978-1-84628-959-0. DOI: 10.1007/978-1-84628-959-0.

[36] P. Elias, « Universal codeword sets and representations of the integers », *IEEE Transactions on Information Theory*, vol. 21, *2*, pp. 194–203, Mar. 1975, ISSN: 0018-9448. DOI: 10.1109/TIT.1975.1055349.

[37] S. Golomb, « Run-length encodings (Corresp.) », *IEEE Transactions on Information Theory*, vol. 12, *3*, pp. 399–401, Jul. 1966, ISSN: 0018-9448. DOI: 10.1109/TIT.1966.1053907.

[38] R. Rice and J. Plaunt, « Adaptive Variable-Length Coding for Efficient Compression of Spacecraft Television Data », *IEEE Transactions on Communication Technology*, vol. 19, *6*, pp. 889–897, Dec. 1971, ISSN: 0018-9332. DOI: 10.1109/TCOM.1971.1090789.

[39] H. Malvar, « Adaptive Run-Length / Golomb-Rice Encoding of Quantized Generalized Gaussian Sources with Unknown Statistics », *in Data Compression Conference (DCC'06)*, Snowbird, UT, USA: IEEE, 2006, pp. 23–32, ISBN: 978-0-7695-2545-7. DOI: 10.1109/DCC.2006.5.

[40] J. Ziv and A. Lempel, « A universal algorithm for sequential data compression », *IEEE Transactions on Information Theory*, vol. 23, *3*, pp. 337–343, May 1977, ISSN: 1557-9654. DOI: 10.1109/TIT.1977.1055714.

[41] J. Ziv and A. Lempel, « Compression of individual sequences via variable-rate coding », *IEEE Transactions on Information Theory*, vol. 24, *5*, pp. 530–536, Sep. 1978, ISSN: 1557-9654. DOI: 10.1109/TIT.1978.1055934.

[42] I. Pavlov. « 7-Zip », 7-Zip file archiver. (), [Online]. Available: https://www.7-zip.org/ (visited on 04/24/2022).

[43] T. Reichardt. « 7-Zip with support for Brotli, Fast-LZMA2, Lizard, LZ4, LZ5 and Zstandard (mcmilk/7-Zip-zstd) », 7-Zip with support for Brotli, Fast-LZMA2, Lizard, LZ4, LZ5 and Zstandard. (Jan. 22, 2024), [Online]. Available: https://github.com/mcmilk/7-Zip-zstd (visited on 01/23/2024).

[44] W. R. Bennett, « Spectra of Quantized Signals », *Bell System Technical Journal*, vol. 27, *3*, pp. 446–472, Jul. 1948, ISSN: 00058580. DOI: 10.1002/j.1538-7305.1948.tb01340.x.

[45] R. Gray, « Quantization noise spectra », *IEEE Transactions on Information Theory*, vol. 36, *6*, pp. 1220–1244, Nov. 1990, ISSN: 1557-9654. DOI: 10.1109/18.59924.

[46] B. Oliver, J. Pierce, and C. Shannon, « The Philosophy of PCM », *Proceedings of the IRE*, vol. 36, *11*, pp. 1324–1331, Nov. 1948, ISSN: 2162-6634. DOI: 10.1109/JRPROC.1948.231941.

[47] Y. Linde, A. Buzo, and R. Gray, « An Algorithm for Vector Quantizer Design », *IEEE Transactions on Communications*, vol. 28, *1*, pp. 84–95, Jan. 1980, ISSN: 1558-0857. DOI: 10.1109/TCOM.1980.1094577.

[48] « Linear Predictive coding vs AR modeling », Signal Processing Stack Exchange. (), [Online]. Available: https://dsp.stackexchange.com/questions/50719/linear-predictive-coding-vs-ar-modeling (visited on 09/12/2021).

[49] M. Van Beurden, « Lossless audio codec comparison : Revision 4 », Audiograaf, Netherlands, Lossless audio codec comparison : Revision 4 4, Jan. 4, 2015.

[50] F. Muin, T. Gunawan, M. Kartiwi, and E. Elsheikh, « A review of lossless audio compression standards and algorithms », presented at the AIP Conference Proceedings, vol. 1883, Sep. 14, 2017, p. 020 006. DOI: 10.1063/1.5002024.

[51] « FLAC - What is FLAC? », FLAC Audio Codec. (2011), [Online]. Available: https://xiph.org/flac/index.html (visited on 04/28/2024).

[52] Xiph. « Free Lossless Audio Codec (FLAC) xiph/flac », Free Lossless Audio Codec (FLAC). (Apr. 25, 2024), [Online]. Available: https://github.com/xiph/flac (visited on 04/28/2024).

[53] « The Apple Lossless Format macosforge/alac », The Apple Lossless Format. (Apr. 21, 2024), [Online]. Available: https://github.com/macosforge/alac (visited on 04/28/2024).

[54] D. Bryant. « WavPack Audio Compression », WavPack website. (), [Online]. Available: https://www.wavpack.com/ (visited on 11/14/2021).

[55] D. Bryant. « Hybrid Lossless Wavefile Compressor dbry/WavPack », Hybrid Lossless Wavefile Compressor. (Apr. 22, 2024), [Online]. Available: https://github.com/dbry/WavPack (visited on 04/28/2024).

[56] Matt Ashland. « Monkey's Audio - a fast and powerful lossless audio compressor ». (), [Online]. Available: https://www.monkeysaudio.com/ (visited on 04/28/2024).

[57] Florin Ghido. « OptimFROG - Main », OptimFROG - Lossless audio compressor. (2022), [Online]. Available: http://losslessaudio.org/ (visited on 04/22/2024).

[58] T. Robinson, « SHORTEN: Simple lossless and near-lossless waveform compression », Cambridge University Engineering Department, Technical report CUED/F-INFENG/TR.156, Dec. 1994.

[59] K. Brandenburg and G. Stoll, « ISO/MPEG-1 Audio: A Generic Standard for Coding of High-Quality Digital Audio », *Journal of the Audio Engineering Society*, vol. 42, *10*, pp. 780–792, Oct. 1, 1994.

[60] « Advanced Audio Coding (MPEG-4) ». (Apr. 28, 2022), [Online]. Available: https://www.loc.gov/preservation/digital/formats/fdd/fdd000114.shtml (visited on 04/22/2024).

[61] K. Brandenburg, « MP3 and AAC Explained »,

[62] S. Pfeiffer, « The Ogg Encapsulation Format Version 0 », Internet Engineering Task Force, Request for Comments RFC 3533, May 2003, 15 pp. DOI: 10.17487/RFC3533.

[63] J.-M. Valin, K. Vos, and T. B. Terriberry, « Definition of the Opus Audio Codec », Internet Engineering Task Force, Request for Comments RFC 6716, Sep. 2012, 326 pp. DOI: 10.17487/RFC6716.

[64] J. Cleary and I. Witten, « Data Compression Using Adaptive Coding and Partial String Matching », *IEEE Transactions on Communications*, vol. 32, *4*, pp. 396–402, Apr. 1984, ISSN: 1558-0857. DOI: 10.1109/TCOM.1984.1096090.

[65] A. Moffat, « Implementing the PPM data compression scheme », *IEEE Transactions on Communications*, vol. 38, *11*, pp. 1917–1921, Nov. 1990, ISSN: 1558-0857. DOI: 10.1109/26.61469.

[66] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, « Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition », *in Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, Nov. 1993, 40–44 vol.1. DOI: 10.1109/ACSSC.1993.342465.

[67] T. Blumensath and M. E. Davies. « Iterative Hard Thresholding for Compressed Sensing ». arXiv: 0805.0510 [cs, math]. (May 5, 2008), pre-published.

[68] Y. C. Eldar and G. Kutyniok, Eds., *Compressed Sensing: Theory and Applications*. Cambridge: Cambridge University Press, 2012, ISBN: 978-1-107-00558-7. DOI: 10.1017/CBO9780511794308.

[69] D. L. Donoho, « For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution », *Communications on Pure and Applied Mathematics*, vol. 59, *6*, pp. 797–829, Jun. 2006, ISSN: 0010-3640, 1097-0312. DOI: 10.1002/cpa.20132.

[70] D. L. Donoho, « For most large underdetermined systems of equations, the minimal l1-norm near-solution approximates the sparsest near-solution », *Communications on Pure and Applied Mathematics*, vol. 59, *7*, pp. 907–934, Jul. 2006, ISSN: 0010-3640, 1097-0312. DOI: 10.1002/cpa.20131.

[71] S. S. Chen, D. L. Donoho, and M. A. Saunders, « Atomic Decomposition by Basis Pursuit », *SIAM Journal on Scientific Computing*, vol. 20, *1*, pp. 33–61, Jan. 1998, ISSN: 1064-8275. DOI: 10.1137/S1064827596304010.

[72] R. Tibshirani, « Regression Shrinkage and Selection Via the Lasso », *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, *1*, pp. 267–288, 1996, ISSN: 2517-6161. DOI: 10.1111/j.2517-6161.1996.tb02080.x.

[73] E. J. Candès, J. K. Romberg, and T. Tao, « Stable signal recovery from incomplete and inaccurate measurements », *Communications on Pure and Applied Mathematics*, vol. 59, *8*, pp. 1207–1223, 2006, ISSN: 1097-0312. DOI: 10.1002/cpa.20124.

[74] E. Candes, J. Romberg, and T. Tao, « Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information », *IEEE Transactions on Information Theory*, vol. 52, *2*, pp. 489–509, Feb. 2006, ISSN: 1557-9654. DOI: 10.1109/TIT.2005.862083.

[75] E. Candes and T. Tao, « Decoding by Linear Programming », *IEEE Transactions on Information Theory*, vol. 51, *12*, pp. 4203–4215, Dec. 2005, ISSN: 0018-9448. DOI: 10.1109/TIT.2005.858979.

[76] E. Candes and T. Tao. « Near Optimal Signal Recovery From Random Projections: Universal Encoding Strategies? » arXiv: math/0410542. (Apr. 4, 2006), pre-published.

[77] D. Donoho, « Compressed sensing », *IEEE Transactions on Information Theory*, vol. 52, *4*, pp. 1289–1306, Apr. 2006, ISSN: 0018-9448. DOI: 10.1109/TIT.2006.871582.

[78] E. J. Candès, « The restricted isometry property and its implications for compressed sensing », *Comptes Rendus Mathematique*, vol. 346, *9*, pp. 589–592, May 1, 2008, ISSN: 1631-073X. DOI: 10.1016/j.crma.2008.03.014.

[79] J. Mairal, « Sparse coding for machine learning, image processing and computer vision »,

[80] J. Mairal, F. Bach, and J. Ponce, *Sparse Modeling for Image and Vision Processing* (Foundations and Trends in Computer Graphics and Vision 2-3), 1 vols. France: Now Foundations and Trends, Dec. 2014, vol. 8, 85-283, ISBN: 978-1-68083-009-5. DOI: 10.1561/0600000058.

[81] J. L. Paredes, G. R. Arce, and Z. Wang, « Ultra-Wideband Compressed Sensing: Channel Estimation », *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, *3*, pp. 383–395, Oct. 2007, ISSN: 1941-0484. DOI: 10.1109/JSTSP.2007.906657.

[82] B. Ansari, V. Kaushik, and S. K. Biswas, « Raw GNSS Data Compression using Compressive Sensing for Reflectometry Applications », *in 2020 XXXIIIrd General Assembly and Scientific Symposium of the International Union of Radio Science*, Aug. 2020, pp. 1–4. DOI: 10.23919/URSIGASS49373.2020.9232248.

[83] « Compressive Sensing Resources », Digital signal processing at rice university. (2017), [Online]. Available: http://dsp.rice.edu/cs/ (visited on 03/08/2021).

[84] I. Carron. « Compressive Sensing: The Big Picture ». (), [Online]. Available: https://sites.google.com/site/igorcarron2/cs#sparse (visited on 10/30/2022).

[85] S. Hosny, M. W. El-Kharashi, and A. T. Abdel-Hamid, « Survey on compressed sensing over the past two decades », *Memories - Materials, Devices, Circuits and Systems*, vol. 4, p. 100 060, Jul. 2023, ISSN: 27730646. DOI: 10.1016/j.memori.2023.100060.

[86] F. Amini, Y. Hedayati, and H. Zanddizari, « Determining the number of measurements for compressive sensing of traffic-induced vibration data », *Measurement*, vol. 152, p. 107 259, Feb. 2020, ISSN: 02632241. DOI: 10.1016/j.measurement.2019.107259.

[87] I. Daubechies, M. Defrise, and C. De Mol, « An iterative thresholding algorithm for linear inverse problems with a sparsity constraint », *Communications on Pure and Applied Mathematics*, vol. 57, *11*, pp. 1413–1457, 2004, ISSN: 1097-0312. DOI: 10.1002/cpa.20042.

[88] J. Max, « Quantizing for minimum distortion », *IRE Transactions on Information Theory*, vol. 6, *1*, pp. 7–12, Mar. 1960, ISSN: 2168-2712. DOI: 10.1109/TIT.1960.1057548.

[89] S. Lloyd, « Least squares quantization in PCM », *IEEE Transactions on Information Theory*, vol. 28, *2*, pp. 129–137, Mar. 1982, ISSN: 1557-9654. DOI: 10.1109/TIT.1982.1056489.

[90] « Open Radio equipment Interface ORI interface Specification; Part 1: Low Layers (Release 4) », ETSI, Sophia Antipolis, Group specification RGS/ORI-0017, Oct. 2014.

[91] J. MacQueen, « Some methods for classification and analysis of multivariate observations », *in Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, vol. 5.1, University of California Press, Jan. 1, 1967, pp. 281–298.

[92] F. M. Gardner, *Phaselock Techniques*, 3rd ed. Hoboken, NJ: John Wiley, 2005, 425 pp., ISBN: 978-0-471-43063-6.

[93] U. Mengali, *Synchronization Techniques for Digital Receivers*. Springer Science & Business Media, Oct. 31, 1997, 548 pp., ISBN: 978-0-306-45725-8. Google Books: 89Gscsw7PvoC.

[94] F. Harris, *Multirate Signal Processing for Communication Systems*. Upper Saddle River, N.J: Prentice Hall PTR, 2004, 478 pp., ISBN: 978-0-13-146511-4.

[95] F. M. Gardner, « Interpolation in Digital Modems-Part I: Fundamentals », *IEEE Transactions on Communications*, vol. 41, *3*, p. 7, 1993.

[96] L. Erup, « Interpolation in Digital Modems-Part II: Implementation and Performance », *IEEE Transactions on Communications*, vol. 41, *6*, p. 11, 1993.

[97] « Telemetry Standards, RCC Standard 106-20 Chapter 2, July 2020 », IRIG-106, Technical specification 106-20, Jul. 2020.

[98] M. Lustig, D. Donoho, and J. M. Pauly, « Sparse MRI: The application of compressed sensing for rapid MR imaging », *Magnetic Resonance in Medicine*, vol. 58, *6*, pp. 1182–1195, Dec. 2007, ISSN: 0740-3194, 1522-2594. DOI: 10.1002/mrm.21391.

[99] M. Dikaiakos, « AdaM: An Adaptive Monitoring Framework for Sampling and Filtering on IoT Devices. », Jan. 1, 2015.

[100] N. K. Sharma and T. V. Sreenivas, « Event-triggered sampling and reconstruction of sparse real-valued trigonometric polynomials », *in 2014 International Conference on Signal Processing and Communications (SPCOM)*, Jul. 2014, pp. 1–6. DOI: 10.1109/SPCOM.2014.6983916.

[101] R. Kumar, P. Seetharaman, A. Luebs, I. Kumar, and K. Kumar. « High-Fidelity Audio Compression with Improved RVQGAN ». arXiv: 2306.06546 [cs, eess]. (Oct. 26, 2023), pre-published.

[102] « TSAC: Very Low Bitrate Audio Compression ». (), [Online]. Available: https://bellard.org/tsac/ (visited on 05/03/2024).

[103] T. Reichardt, *Mcmilk/zstdmt*, Jan. 13, 2024.

[104] Welch, « A Technique for High-Performance Data Compression », *Computer*, vol. 17, *6*, pp. 8–19, Jun. 1984, ISSN: 1558-0814. DOI: 10.1109/MC.1984.1659158.

[105] M. Bartík, S. Ubik, and P. Kubalik, « LZ4 compression algorithm on FPGA », *in 2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, Dec. 2015, pp. 179–182. DOI: 10.1109/ICECS.2015.7440278.

[106] Y. Collet and M. Kucherawy, « Zstandard Compression and the application/zstd Media Type », Internet Engineering Task Force, Request for Comments RFC 8478, Oct. 2018, 54 pp. DOI: 10.17487/RFC8478.

[107] J. Alakuijala, A. Farruggia, P. Ferragina, *et al.*, « Brotli: A General-Purpose Data Compressor », *ACM Transactions on Information Systems*, vol. 37, *1*, 4:1–4:30, Dec. 6, 2018, ISSN: 1046-8188. DOI: 10.1145/3231935.

[108] J. Alakuijala and Z. Szabadka, « Brotli Compressed Data Format », Internet Engineering Task Force, Request for Comments RFC 7932, Jul. 2016, 128 pp. DOI: 10.17487/RFC7932.

[109] D. Salomon and G. Motta, *Handbook of Data Compression*, 5th ed. London ; New York: Springer, 2010, 1359 pp., ISBN: 978-1-84882-902-2.

[110] M. Burrows, D. J. W. L, and R. W. Taylor, « A Block-sorting Lossless Data Compression Algorithm », 1994.

[111] T. Piana, A. Aissa El Bey, M. Arzel, and A. Thomas, « Sparse Coding Techniques for Adjustable Lossy Compression of Baseband Signals in Centralized and Virtualized Communication Systems »,

[112] M. Aharon, M. Elad, and A. Bruckstein, « K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation », *IEEE Transactions on Signal Processing*, vol. 54, *11*, p. 12, 2006.

[113] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, « Online dictionary learning for sparse coding », *in Proceedings of the 26th Annual International Conference on Machine Learning*, Montreal Quebec Canada: ACM, Jun. 14, 2009, pp. 689–696, ISBN: 978-1-60558-516-1. DOI: 10.1145/1553374.1553463.

[114] E. Crespo Marques, N. Maciel, L. Naviner, H. Cai, and J. Yang, « A Review of Sparse Recovery Algorithms », *IEEE Access*, vol. 7, pp. 1300–1322, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2886471.

[115] A. Çivril, « A note on the hardness of sparse approximation », *Information Processing Letters*, vol. 113, *14-16*, pp. 543–545, Jul. 2013, ISSN: 00200190. DOI: 10.1016/j.ipl.2013.04.014.

[116] R. Fan, Q. Wan, Y. Liu, H. Chen, and X. Zhang. « Complex Orthogonal Matching Pursuit and Its Exact Recovery Conditions ». arXiv: 1206.2197 [cs, math, stat]. (Jun. 11, 2012), prepublished.

[117] J. Wang, S. Kwon, and B. Shim, « Generalized Orthogonal Matching Pursuit », *IEEE Transactions on Signal Processing*, vol. 60, *12*, pp. 6202–6216, Dec. 2012, ISSN: 1941-0476. DOI: 10.1109/TSP.2012.2218810.

[118] D. Needell and J. Tropp, « CoSaMP: Iterative signal recovery from incomplete and inaccurate samples », *Applied and Computational Harmonic Analysis*, vol. 26, *3*, pp. 301–321, May 2009, ISSN: 10635203. DOI: 10.1016/j.acha.2008.07.002.

[119] H. Mohimani, M. Babaie-Zadeh, and C. Jutten, « A Fast Approach for Overcomplete Sparse Decomposition Based on Smoothed L0 Norm », *IEEE Transactions on Signal Processing*, vol. 57, *1*, pp. 289–301, Jan. 2009, ISSN: 1941-0476. DOI: 10.1109/TSP.2008.2007606.

[120] H. Zou and T. Hastie, « Regularization and variable selection via the elastic net », *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, *2*, pp. 301–320, 2005, ISSN: 1467-9868. DOI: 10.1111/j.1467-9868.2005.00503.x.

[121] R. Rubinstein, M. Zibulevsky, and M. Elad, « Efficient Implementation of the K-SVD Algorithm Using Batch Orthogonal Matching Pursuit », *CS Technion*, vol. 40, Jan. 1, 2008.

[122] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, « Online Learning for Matrix Factorization and Sparse Coding », *Journal of Machine Learning Research*, vol. 11, Aug. 1, 2009. DOI: 10.1145/1756006.1756008.

[123] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. New York, NY: Springer New York, 2010, ISBN: 978-1-4419-7011-4. DOI: 10.1007/978-1-4419-7011-4.

[124] H. Wang and T. Celik, « Sparse Representation-Based Hyperspectral Data Processing: Lossy Compression », *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, *5*, pp. 2036–2045, May 2017, ISSN: 2151-1535. DOI: 10.1109/JSTARS.2017.2666264.

[125] X. Zhang, W. Lin, Y. Zhang, *et al.*, « Rate-Distortion Optimized Sparse Coding With Ordered Dictionary for Image Set Compression », *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, *12*, pp. 3387–3397, Dec. 2018, ISSN: 1558-2205. DOI: 10.1109/TCSVT.2017.2748382.

[126] T. Kavitha and K. Jaya Sankar, « A Fast and Efficient Lossless Compression Technique for Greyscale Images », *International Journal of Circuits, Systems and Signal Processing*, vol. 14, Apr. 30, 2020, ISSN: 1998-4464. DOI: 10.46300/9106.2020.14.18.

[127] M. Kalluri, M. Jiang, N. Ling, J. Zheng, and P. Zhang, « Adaptive RD Optimal Sparse Coding With Quantization for Image Compression », *IEEE Transactions on Multimedia*, vol. 21, *1*, pp. 39–50, Jan. 2019, ISSN: 1941-0077. DOI: 10.1109/TMM.2018.2847228.

[128] B. McKenzie and T. Bell, « Compression of sparse matrices by blocked Rice coding », *IEEE Transactions on Information Theory*, vol. 47, *3*, pp. 1223–1230, Mar. 2001, ISSN: 00189448. DOI: 10.1109/18.915692.

[129] S. St-Jean, *Samuelstjean/spams-python*, Feb. 21, 2024.

[130] S. Sarvotham, D. Baron, and R. Baraniuk, « Measurements vs. Bits: Compressed Sensing meets Information Theory », vol. 3, Jan. 1, 2006.

[131] H.-N. Lee, *Introduction to Compressed Sensing with Coding Theoretic Perspective*. CRC Press, Feb. 15, 2017, 250 pp., ISBN: 978-1-4987-1030-5.

[132] M. Elad, « Optimized Projections for Compressed Sensing », *IEEE Transactions on Signal Processing*, vol. 55, *12*, pp. 5695–5702, Dec. 2007, ISSN: 1053-587X, 1941-0476. DOI: 10.1109/TSP.2007.900760.

[133] J. A. Tropp and A. C. Gilbert, « Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit », *IEEE Transactions on Information Theory*, vol. 53, *12*, pp. 4655–4666, Dec. 2007, ISSN: 0018-9448. DOI: 10.1109/TIT.2007.909108.

[134] M. Nazari, A. Mehrpooya, M. Bastani, and M. Nayebi, « Sparse Recovery Using Modified SL0 Algorithm by Weighted Projection and Application to ISAR Imaging », *in 2020 28th Iranian Conference on Electrical Engineering (ICEE)*, Aug. 2020, pp. 1–7. DOI: 10.1109/ICEE50131.2020.9260573.

[135] V. Abolghasemi, S. Ferdowsi, B. Makkiabadi, and S. Sanei, « On optimization of the measurement matrix for compressive sensing », *in 2010 18th European Signal Processing Conference*, Aug. 2010, pp. 427–431.

[136] B. L. Sturm, « When “Exact Recovery" is Exact Recovery in Compressed Sensing Simulation », p. 26,

[137] A. Shiri and G. Khosroshahi, *An FPGA Implementation of Singular Value Decomposition*. Apr. 1, 2019, p. 422, 416 pp. DOI: 10.1109/IranianCEE.2019.8786719.

[138] M. TIAN, M. SIMA, and M. McGUIRE, « Behavioral Implementation of SVD on FPGA », *in 2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Dec. 2018, pp. 495–500. DOI: 10.1109/ISSPIT.2018.8642667.

[139] C. Morales-Pérez, J. Rangel-Magdaleno, I. Cruz-Vega, J. Ramirez-Cortes, and H. Peregrina-Barreto, « FPGA implementation of Orthogonal Matching Pursuit algorithm », *in 2016 13th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, Sep. 2016, pp. 1–6. DOI: 10.1109/ICEEE.2016.7751200.

[140] J. Li, P. Chow, Y. Peng, and T. Jiang, « FPGA Implementation of an Improved OMP for Compressive Sensing Reconstruction », *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, *2*, pp. 259–272, Feb. 2021, ISSN: 1557-9999. DOI: 10.1109/TVLSI.2020.3030906.

[141] G. Einarsson, « An improved implementation of predictive coding compression », *IEEE Transactions on Communications*, vol. 39, *2*, pp. 169–171, Feb. 1991, ISSN: 1558-0857. DOI: 10.1109/26.76478.

[142] H. Akaike, « A new look at the statistical model identification », *IEEE Transactions on Automatic Control*, vol. 19, *6*, pp. 716–723, Dec. 1974, ISSN: 0018-9286. DOI: 10.1109/TAC.1974.1100705.

[143] G. Schwarz, « Estimating the Dimension of a Model », *The Annals of Statistics*, vol. 6, *2*, Mar. 1, 1978, ISSN: 0090-5364. DOI: 10.1214/aos/1176344136.

[144] J. Rissanen, « Modeling by shortest data description », *Automatica*, vol. 14, *5*, pp. 465–471, Sep. 1, 1978, ISSN: 0005-1098. DOI: 10.1016/0005-1098(78)90005-5.

[145] J. Durbin, « The Fitting of Time-Series Models », *Revue de l'Institut International de Statistique / Review of the International Statistical Institute*, vol. 28, *3*, p. 233, 1960, ISSN: 03731138. DOI: 10.2307/1401322. JSTOR: 1401322.

[146] B. Atal, « The history of linear prediction », *IEEE Signal Processing Magazine*, vol. 23, *2*, pp. 154–161, Mar. 2006, ISSN: 1053-5888, 1558-0792. DOI: 10.1109/MSP.2006.1598091.

[147] G. Fleury, *Analyse spectrale. Méthodes non-paramétriques et paramétriques* (Technosup), 1st ed., 1 vols. Ellipses, Jul. 17, 2001, vol. 1, 190 pp., ISBN: 2-7298-0495-1.

[148] I. Ibrahim and M. El-Adawy, « Determination of the LPC coefficients using neural networks », *in Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 5, Perth, WA, Australia: IEEE, 1995, pp. 2669–2673, ISBN: 978-0-7803-2768-9. DOI: 10.1109/ICNN.1995.487832.

[149] R. Gallager and D. van Voorhis, « Optimal source codes for geometrically distributed integer alphabets », *IEEE Transactions on Information Theory*, vol. 21, *2*, pp. 228–230, Mar. 1975, ISSN: 1557-9654. DOI: 10.1109/TIT.1975.1055357.

[150] J. Xu, A. Ariyaeeinia, and R. Sotudeh, « Migrate Levinson-Durbin based Linear Predictive Coding algorithm into FPGAS », *in 2005 12th IEEE International Conference on Electronics, Circuits and Systems*, Dec. 2005, pp. 1–4. DOI: 10.1109/ICECS.2005.4633388.

[151] P. P. Vaidyanathan, *The Theory of Linear Prediction*. Cham: Springer International Publishing, 2008, ISBN: 978-3-031-02527-3. DOI: 10.1007/978-3-031-02527-3.

[152] Ye Li and Z. Ding, « ARMA system identification based on second-order cyclostationarity », *IEEE Transactions on Signal Processing*, vol. 42, *12*, pp. 3483–3494, Dec. 1994, ISSN: 1053-587X, 1941-0476. DOI: 10.1109/78.340782.

[153] A. Harma, « Linear predictive coding with modified filter structures », *IEEE Transactions on Speech and Audio Processing*, vol. 9, *8*, pp. 769–777, Nov. 2001, ISSN: 10636676. DOI: 10.1109/89.966080.

[154] V. Volterra, *Theory of Functionals and of Integral and Integro-differential Equations*. Dover Publications, Jan. 1, 2005, 294 pp., ISBN: 978-0-486-44284-6. Google Books: HhAoddpkcWgC.

[155] D. J. Curtis, « Volterra kernels, Oldroyd models, and interconversion in superposition rheometry »,

[156] M. Fréchet, « Sur les fonctionnelles continues », *Annales scientifiques de l'École Normale Supérieure*, vol. 27, pp. 193–216, 1910, ISSN: 0012-9593.

[157] K. Batselier, Z. Chen, H. Liu, and N. Wong, « A tensor-based volterra series black-box nonlinear system identification and simulation framework », *in Proceedings of the 35th International Conference on Computer-Aided Design*, Austin Texas: ACM, Nov. 7, 2016, pp. 1–7, ISBN: 978-1-4503-4466-1. DOI: 10.1145/2966986.2966996.

[158] G. Mileounis and N. Kalouptsidis, « Blind Identification of Second Order Volterra Systems With Complex Random Inputs Using Higher Order Cumulants », *IEEE Transactions on Signal Processing*, vol. 57, *10*, pp. 4129–4135, Oct. 2009, ISSN: 1053-587X, 1941-0476. DOI: 10.1109/TSP.2009.2023372.

[159] S. Boyd, Y. Tang, and L. Chua, « Measuring Volterra kernels », *IEEE Transactions on Circuits and Systems*, vol. 30, *8*, pp. 571–577, Aug. 1983, ISSN: 0098-4094. DOI: 10.1109/TCS.1983.1085391.

[160] J. Wray and G. G. R. Green, « Calculation of the Volterra kernels of non-linear dynamic systems using an artificial neural network », *Biological Cybernetics*, vol. 71, *3*, pp. 187–195, Jul. 1994, ISSN: 0340-1200, 1432-0770. DOI: 10.1007/BF00202758.

[161] R. Logeswaran, « A prediction-based neural network scheme for lossless data compression », *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 32, *4*, pp. 358–365, Nov. 2002, ISSN: 1094-6977. DOI: 10.1109/TSMCC.2002.806744.

[162] N. Sriraam, R. Kannan, and C. Eswaran, « Lossless compression of EEG data using neural network predictors », *in Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP '02.*, vol. 4, Singapore: Nanyang Technol. Univ, Nov. 2002, pp. 2046–2048, ISBN: 978-981-04-7524-6. DOI: 10.1109/ICONIP.2002.1199034.

[163] A. F. Shehab, M. A. Elshafey, and T. A. Mahmoud, « Recurrent Neural Network Based Prediction to Enhance Satellite Telemetry Compression », *in 2020 IEEE Aerospace Conference*, Big Sky, MT, USA: IEEE, Mar. 2020, pp. 1–11, ISBN: 978-1-72812-734-7. DOI: 10.1109/AERO47225.2020.9172719.

[164] T. Wang, C.-K. Wen, H. Wang, F. Gao, T. Jiang, and S. Jin, « Deep learning for wireless physical layer: Opportunities and challenges », *China Communications*, vol. 14, *11*, pp. 92–111, Nov. 2017, ISSN: 1673-5447. DOI: 10.1109/CC.2017.8233654.

[165] T. Piana, M. Arzel, A. Aissa El Bey, and A. Thomas, « Neural Networks as Catalysts for Enhanced Lossless Compression in Satellite Communications », presented at the International Conference on Emerging Technologies for Communication, ICETC 2023, ICETC, Nov. 29, 2023, p. 4.

[166] K. Cho, B. van Merrienboer, C. Gulcehre, *et al.* « Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation ». arXiv: 1406.1078 [cs, stat]. (Sep. 2, 2014), pre-published.

[167] V. Dumoulin and F. Visin. « A guide to convolution arithmetic for deep learning ». arXiv: 1603.07285 [cs, stat]. (Jan. 11, 2018), pre-published.

[168] « ETSI EN 302 307 - Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2) », 1.2.1, Aug. 2009.

[169] « ETSI EN 302 307-2 - Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications; Part 2: DVB-S2 Extensions (DVB-S2X) | Engineering360 », 1.2.1, Aug. 2022.

[170] « ETSI TR 102 376 - Digital Video Broadcasting (DVB) User guidelines for the second generation system for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2) », Technical report 1.1.1, Feb. 2005.

[171] C. Trabelsi, O. Bilaniuk, Y. Zhang, *et al.* « Deep Complex Networks ». arXiv: 1705.09792 [cs]. (Feb. 25, 2018), pre-published.

[172] Z. Wang, F. Lu, D. Wang, X. Zhang, J. Li, and J. Li, « A Transmission Efficiency Evaluation Method of Adaptive Coding Modulation for Ka-Band Data-Transmission of LEO EO Satellites », *Sensors*, vol. 22, *14*, p. 5423, Jul. 20, 2022, ISSN: 1424-8220. DOI: 10.3390/s22145423.

[173] C. N. Hesse, « Analysis and Comparison of Performance and Power Consumption of Neural Networks on CPU, GPU, TPU and FPGA », Dec. 5, 2021.

[174] « Fpgas Vs Cpus In Ai Processing | Restackio ». (), [Online]. Available: https://www.restack.io/p/fpga-ai-integration-techniques-answer-fpga-vs-cpu-ai-cat-ai (visited on 11/28/2024).

[175] C. Wang and Z. Luo, « A Review of the Optimal Design of Neural Networks Based on FPGA », *Applied Sciences*, vol. 12, *21*, p. 10 771, 21 Jan. 2022, ISSN: 2076-3417. DOI: 10.3390/app122110771.

[176] E. Nurvitadhi, R. D'Souza, and M. Won, « Real Performance of FPGAs Tops GPUs in the Race to Accelerate AI »,

[177] Y. LeCun, J. Denker, and S. Solla, « Optimal Brain Damage », *in Advances in Neural Information Processing Systems*, vol. 2, Morgan-Kaufmann, 1989.

[178] S. Han, J. Pool, J. Tran, and W. Dally, « Learning both Weights and Connections for Efficient Neural Network », *in Advances in Neural Information Processing Systems*, vol. 28, Curran Associates, Inc., 2015.

[179] J. Frankle and M. Carbin. « The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks ». arXiv: 1803.03635 [cs]. (Mar. 4, 2019), pre-published.

[180] Y. He, X. Zhang, and J. Sun, « Channel Pruning for Accelerating Very Deep Neural Networks », *in 2017 IEEE International Conference on Computer Vision (ICCV)*, Venice: IEEE, Oct. 2017, pp. 1398–1406, ISBN: 978-1-5386-1032-9. DOI: 10.1109/ICCV.2017.155.

[181] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. « Pruning Filters for Efficient ConvNets ». arXiv: 1608.08710 [cs]. (Mar. 10, 2017), pre-published.

[182] J.-H. Luo, J. Wu, and W. Lin. « ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression ». arXiv: 1707.06342 [cs]. (Jul. 19, 2017), pre-published.

[183] Y. Umuroglu, N. J. Fraser, G. Gambardella, *et al.*, « FINN: A Framework for Fast, Scalable Binarized Neural Network Inference », *in Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17, New York, NY, USA: Association for Computing Machinery, Feb. 22, 2017, pp. 65–74, ISBN: 978-1-4503-4354-1. DOI: 10.1145/3020078.3021744.

[184] J. Duarte, S. Han, P. Harris, *et al.*, « Fast inference of deep neural networks in FPGAs for particle physics », *Journal of Instrumentation*, vol. 13, *07*, P07027, Jul. 2018, ISSN: 1748-0221. DOI: 10.1088/1748-0221/13/07/P07027.

[185] A. Vaswani, N. Shazeer, N. Parmar, *et al.* « Attention Is All You Need ». arXiv: 1706.03762 [cs]. (Aug. 1, 2023), pre-published.

[186] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi. « SoundStream: An End-to-End Neural Audio Codec ». arXiv: 2107.03312 [cs, eess]. (Jul. 7, 2021), pre-published.

[187] F. Mentzer, D. Minnen, E. Agustsson, and M. Tschannen. « Finite Scalar Quantization: VQ-VAE Made Simple ». arXiv: 2309.15505 [cs]. (Oct. 12, 2023), pre-published.

[188] C.-C. Chiu, J. Qin, Y. Zhang, J. Yu, and Y. Wu. « Self-supervised Learning with Random-projection Quantizer for Speech Recognition ». arXiv: 2202.01855 [cs, eess]. (Jun. 29, 2022), pre-published.

[189] K. Hsu, W. Dorrell, J. C. R. Whittington, J. Wu, and C. Finn. « Disentanglement via Latent Quantization ». arXiv: 2305.18378 [cs, stat]. (Oct. 22, 2023), pre-published.

[190] Wes Bouaziz. « SoundStream neural network for audio compression (GitHub - wesbz/SoundStream) », SoundStream: An End-to-End Neural Audio Codec. (Nov. 4, 2021), [Online]. Available: https://github.com/wesbz/SoundStream (visited on 04/24/2024).

[191] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi. « State-of-the-art deep learning based audio codec supporting both mono 24 kHz audio and stereo 48 kHz audio. (GitHub - facebookresearch/encodec) », EnCodec: High Fidelity Neural Audio Compression. (Nov. 2022), [Online]. Available: https://github.com/facebookresearch/encodec (visited on 04/24/2024).

[192] *Half-normal distribution*, *in Wikipedia*, Apr. 4, 2024.

[193] F. C. Leone, L. S. Nelson, and R. B. Nottingham, « The Folded Normal Distribution », *Technometrics*, vol. 3, *4*, pp. 543–550, Nov. 1, 1961, ISSN: 0040-1706. DOI: 10.1080/00401706.1961.10489974.

Part IV

# Appendices

# Average compression of Rice coding & ideal value of Rice parameter

## A.1 Introduction

Rice coding, a type of entropy coding, is closely related to Golomb coding and is particularly effective for encoding integers that follow a geometric distribution. The effectiveness of Rice coding heavily relies on a parameter, commonly denoted as $k$, **which determines the partitioning of the input integers into a unary and binary part. This parameter plays a crucial role in optimizing the compression efficiency.**

**Choosing the appropriate parameter $k$ is vital because an inappropriate value can significantly reduce compression performance**. However, determining the optimal $k$ is challenging due to its dependence on the underlying data distribution, which is often unknown or varies over time.

This appendix addresses the problem of analytically selecting the optimal Rice parameter. Using certain approximations, we propose a method to estimate the best parameter $k$ that maximizes the compression efficiency. This approach aims to provide a practical solution to the parameter selection problem, enhancing the utility of Rice coding in various applications.

## A.2 Normal & half-normal distribution

Let $Y$ be a random variable following a normal distribution centered at zero with variance $\sigma^2$. We define the random variable $Y^{[\mathrm{F}]}$ as:

$$Y^{[\mathrm{F}]} = |Y| \tag{A.1}$$

This distribution is known as the half-normal distribution (or *Folded Normal Distribution*) [192], [193]. Its properties are as follows:

$$\forall y > 0 \quad \begin{cases} f_Y(y) = \dfrac{\sqrt{2}}{\sigma\sqrt{\pi}} \exp\left(-\dfrac{y^2}{2\sigma^2}\right) & \text{(The PDF)} \\ F_Y(y) = \operatorname{erf}\left(\dfrac{y}{\sqrt{2}\sigma}\right) & \text{(The CDF)} \\ \mathbb{E}(Y) = \dfrac{\sigma\sqrt{2}}{\sqrt{\pi}} & \text{(The Mean)} \end{cases} \tag{A.2}$$

## A.3 Investigation of the residual distribution

Let $\mathcal{R}$ be the random variable that represents the residuals of the signal, defined as the difference between the original signal $y$ and a predicted signal $\tilde{y}$. We denote $\sigma_{\mathcal{R}}$ as the standard deviation of $\mathcal{R}$. Since $\mathcal{R}$ is the difference between two quantized signals –the prediction and the original signal, both quantized on $N_q$ bits–, **we can easily suppose that $\mathcal{R}$ is also a quantized signal**. Then, if $Y$ is a random variable following the same probability distribution as $\mathcal{R}$, we can note: $\mathcal{R} = \lfloor Y \rfloor$.

$\mathcal{R}$ then follows a discrete probability distribution, which can be described with the probability density function (PDF) $f_Y$ of $Y$. Therefore, the relation used to determine the probability of each symbols is defined by

$$\forall x \in \mathbb{R},\ \mathbb{P}(\mathcal{R} = i) = \int_i^{i+1} f_Y(x)\,dx \tag{A.3}$$

In practice, the standard deviation depends on several parameters (signal noise level, number of coefficients, quantization of coefficients, precision of coefficients, approximation method used (LPC, AR, MA, ARMA, etc.). Providing an exact description would be impractical, and a numerical approximation of sigma with some fixed parameters yields more reliable results.

---

Now, let $\mathcal{R}_{\mathrm{P}}$ be the random variable that represents the residuals after pre-processing. This preprocessing involves applying the transformation $\mathcal{T} : \mathbb{R} \longrightarrow \mathbb{R}^+$ to each residual $x \in \mathcal{R}$, defined as:

$$\mathcal{T}(x) = \begin{cases} 2x & \text{if } x > 0 \\ -2x - 1 & \text{otherwise} \end{cases} \tag{A.4}$$

Through this transformation, $\mathcal{R}_{\mathrm{P}}$ follows a discrete probability distribution that can be deduced from the probability distribution of a half-normal distribution with parameter $2\sigma_{\mathcal{R}}$ –it means we can assume that $\mathcal{R}_{\mathrm{P}} = \lfloor 2Y^{[\mathrm{F}]} \rfloor = \lfloor 2|Y| \rfloor$–. Thus,

$$\mathbb{P}(\mathcal{R}_{\mathrm{P}} = i) = \int_i^{i+1} f_{Y^{[\mathrm{F}]}}(x)\,dx = \operatorname{erf}\left(\frac{i+1}{2\sigma_Y\sqrt{2}}\right) - \operatorname{erf}\left(\frac{i}{2\sigma_Y\sqrt{2}}\right) \tag{A.5}$$

$$= \operatorname{erf}\left(\frac{i+1}{2\sigma_{\mathcal{R}}\sqrt{2}}\right) - \operatorname{erf}\left(\frac{i}{2\sigma_{\mathcal{R}}\sqrt{2}}\right) \tag{A.6}$$

$$\tag{A.7}$$

Let $Z_k^1(\mathcal{R}_{\mathrm{P}})$ be the random variable that represents the sizes of $\mathcal{R}_{\mathrm{P}}$ coded using the Rice coding with parameter $k$. This coding consists of a unary part and a binary part:

$$\underbrace{11\ldots10}_{\text{Unary part}}\ \underbrace{01001}_{\text{Binary part}}$$

$$\underbrace{11\ldots10}_{\frac{x}{2^k}+1}\ \underbrace{01001}_{k}$$

## A.4 Rice coding

Let $k$ be an integer greater than or equal to 0. Let $r_n$ be a sequence of residuals represented by perfectly encoded integers over $N$ bits following an arbitrary distribution. The expected value of $Z_k^1(\mathcal{R}_\mathrm{P})$ is defined by:

$$\mathbb{E}\{Z_1\} = (k+1)\underbrace{\mathbb{P}\left(\mathcal{R}_\mathrm{P} < 2^k\right)}_{p_1} + \left[k+1+\underbrace{\sum_{i=0}^{\infty} i\mathbb{P}\left(\left\lfloor\frac{\mathcal{R}_\mathrm{P}}{2^k}\right\rfloor = i\right)}_{P_k(\mathcal{R}_\mathrm{P})}\right](1-p_1) \tag{A.8}$$

$$= k+1+(1-p_1)\,P_k(\mathcal{R}_\mathrm{P}) \tag{A.9}$$

The expected value of $Z_k^1(\mathcal{R}_\mathrm{P})$ is derived from the structure of Rice coding, which consists of a unary part and a binary part. The unary part represents the quotient of the division of $\mathcal{R}_\mathrm{P}$ by $2^k$, while the binary part encodes the remainder. Specifically, the unary part requires $\left\lfloor\frac{\mathcal{R}_\mathrm{P}}{2^k}\right\rfloor + 1$ bits, where $\left\lfloor\frac{\mathcal{R}_\mathrm{P}}{2^k}\right\rfloor$ is the integer quotient of $\mathcal{R}_\mathrm{P}$ divided by $2^k$, and the binary part requires $k$ bits. Therefore, the total number of bits required to encode $\mathcal{R}_\mathrm{P}$ using Rice coding with the parameter $k$ is given by $\left\lfloor\frac{\mathcal{R}_\mathrm{P}}{2^k}\right\rfloor + 1 + k$. The expected value is then obtained by taking the sum of the bits required for all possible values of $\mathcal{R}_\mathrm{P}$, weighted by their respective probabilities. This results in the expected value $\mathbb{E}\{Z_1\} = (k+1)\mathbb{P}\left(\mathcal{R}_\mathrm{P} < 2^k\right) + \left[k+1+\sum_{i=0}^{\infty} i\mathbb{P}\left(\left\lfloor\frac{\mathcal{R}_\mathrm{P}}{2^k}\right\rfloor = i\right)\right](1-\mathbb{P}\left(\mathcal{R}_\mathrm{P} < 2^k\right))$, which simplifies to $\mathbb{E}\{Z_1\} = k+1+(1-p_1)P_k(\mathcal{R}_\mathrm{P})$.

### A.4.1 Normal distribution case

In this section, we examine the scenario in which the residuals, represented by the random variable $\mathcal{R}_\mathrm{P}$, follow a normal distribution. This assumption simplifies the analysis and allows us to derive more precise estimations for the parameters involved in Rice coding. Using the properties of the normal distribution, we can analytically estimate the probabilities and expected values that influence the performance of the Rice coding scheme. This approach facilitates a deeper understanding of the coding efficiency and provides a foundation for further optimization of the Rice parameter.

Assuming that $\mathcal{R}$ follows a normal distribution with parameters $(0, \sigma_\mathcal{R})$, we show that $\mathcal{R}_\mathrm{P}$ follows a half-normal distribution with parameter $\sigma_{\mathcal{R}_\mathrm{P}} = 2\sigma_\mathcal{R} = 2\sigma_Y$.

#### A.4.1.1 $p_1$ estimation

Using the definition of the Cumulative Distribution Function (CDF) given by A.2, the probability $p_1$ is given by:

$$p_1 = \mathrm{erf}\left(\frac{2^{k-1}}{2\sqrt{2}\sigma_\mathcal{R}}\right) \tag{A.10}$$

$$= \mathrm{erf}\left(\frac{2^{k-1}}{\sqrt{2}\sigma_\mathcal{R}}\right) \tag{A.11}$$

#### A.4.1.2 $P_k$ estimation

The function $P_k(\mathcal{R}_\mathrm{P})$ represents the expected value of the quotient $\left\lfloor\frac{\mathcal{R}_\mathrm{P}}{2^k}\right\rfloor$, where $\mathcal{R}_\mathrm{P}$ is the pre-processed residual. This expected value is calculated by summing all possible values of the quotient, weighted by their respective probabilities. The derivation of $P_k(\mathcal{R}_\mathrm{P})$ follows from the definition of the expected value, which incorporates the probability distribution of $\mathcal{R}_\mathrm{P}$. Specifically, we express $P_k(\mathcal{R}_\mathrm{P})$ as follows:

$$P_k(\mathcal{R}_\mathrm{P}) = \sum_{i=0}^{\infty} i\mathbb{P}\left(\left\lfloor \frac{\mathcal{R}_\mathrm{P}}{2^k} \right\rfloor = i\right) \tag{A.12}$$

$$= \sum_{i=0}^{\infty} i\mathbb{P}\left(i \leq \frac{\mathcal{R}_\mathrm{P}}{2^k} < i+1\right) \tag{A.13}$$

$$= \sum_{i=0}^{\infty} i \int_i^{i+1} f_{\mathcal{R}_\mathrm{P}}(x)\, dx \tag{A.14}$$

With $f_{\mathcal{R}_\mathrm{P}}(x)$ being the PDF of $\mathcal{R}_\mathrm{P}$. A good approximation consists of replacing $i$ with $x$:

$$i \int_i^{i+1} f_{\mathcal{R}_\mathrm{P}}(x)\, dx \leq \int_i^{i+1} x f_{\mathcal{R}_\mathrm{P}}(x)\, dx < (i+1) \int_i^{i+1} f_{\mathcal{R}_\mathrm{P}}(x)\, dx \tag{A.15}$$

Using the expression $A_i(\mathcal{R}_\mathrm{P})$:

$$0 \leq \int_i^{i+1} x f_{\mathcal{R}_\mathrm{P}}(x)\, dx - \int_i^{i+1} i f_{\mathcal{R}_\mathrm{P}}(x)\, dx < A_k^{\mathcal{R}_\mathrm{P}}(i) \tag{A.16}$$

For a half-normal distribution, we have $A_k^{\mathcal{R}_\mathrm{P}}(i) = \mathrm{erf}\left(\frac{i+1}{2\sqrt{2}\sigma_\mathcal{R}}\right) - \mathrm{erf}\left(2\frac{i}{\sqrt{2}\sigma_\mathcal{R}}\right)$. By numerical simulation, we can estimate the actual approximation error. Figure A.1 illustrates the simulation results assessing the approximation error $A_k^{\mathcal{R}_\mathrm{P}}(i)$. The figure demonstrates that the error is quite minimal within the range of values we are working with.

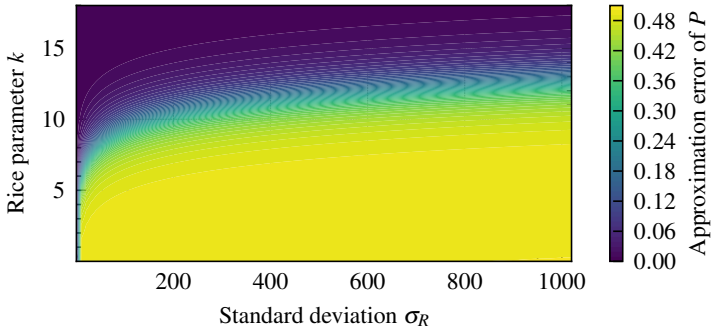
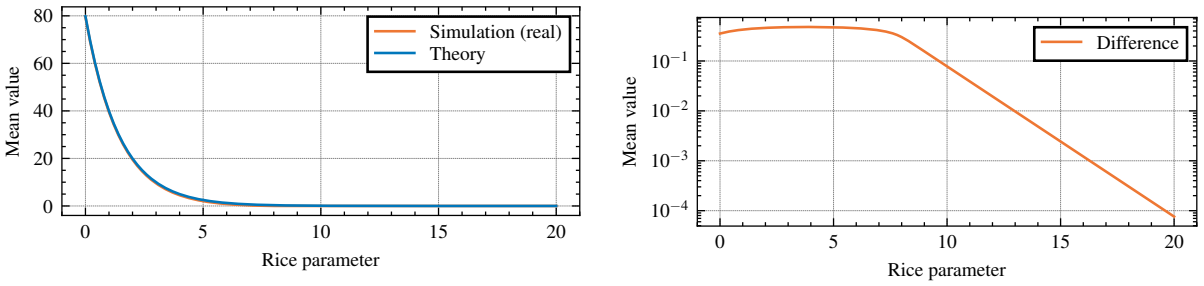



**Figure A.1:** Value of $A_k^{\mathcal{R}_\mathrm{P}}$ depending on $\sigma_\mathcal{R}$ and $k$.

Obtaining an expression for the expectation of $f_{\mathcal{R}_\mathrm{P}}$, we have $P_k(\mathcal{R}_\mathrm{P}) = \mathbb{E}\left\{\frac{\mathcal{R}_\mathrm{P}}{2^k}\right\}$. With some manipulation, we derive the following expression:

$$\mathbb{E}\left(\frac{\mathcal{R}_\mathrm{P}}{2^k}\right) = \frac{\sigma_\mathcal{R}\sqrt{2}}{2^k\sqrt{\pi}} = \frac{2\sigma_\mathcal{R}}{2^k\sqrt{2\pi}} = \frac{4\sigma_\mathcal{R}}{2^k\sqrt{2\pi}} = \frac{\sigma_\mathcal{R}}{2^{k-2}\sqrt{2\pi}} \tag{A.17}$$

Analysis of the estimation error is provided by Figure A.2.




**(a)** $\mathbb{E}\left(\frac{\mathcal{R}_P}{2^k}\right)$ vs theory

**(b)** Absolute error

**Figure A.2:** Difference between our estimated value of $\mathbb{E}\left(\frac{\mathcal{R}_P}{2^k}\right)$ and its actual value. In this context, $\sigma_{\mathcal{R}} = 10^6$

#### A.4.1.3 Estimation of the expectation of the reduced random variable

We aim to estimate the expectation of $Z_1$. This is given by:

$$\mathbb{E}\{Z_1\} = k + 1 + \left(\frac{\sigma_{\mathcal{R}}}{2^{k-2}\sqrt{2\pi}} - \frac{1}{2}\right)\left(1 - \operatorname{erf}\left(\frac{2^{k-1}}{\sqrt{2}\sigma_{\mathcal{R}}}\right)\right) \tag{A.18}$$

$$= k + 1 + \left(\frac{\sigma_{\mathcal{R}}}{2^{k-2}\sqrt{2\pi}} - \frac{1}{2}\right)\operatorname{erfc}\left(\frac{2^{k-1}}{\sqrt{2}\sigma_{\mathcal{R}}}\right) \tag{A.19}$$

$$\approx k + 1 + \left(\frac{\sigma_{\mathcal{R}}}{2^{k-2}\sqrt{2\pi}} - \frac{1}{2}\right)\left(1 - \frac{2}{\sqrt{\pi}}\frac{2^{k-1}}{\sqrt{2}\sigma_{\mathcal{R}}}\right) \tag{A.20}$$

Here, the approximation in the last step uses the series expansion of the complementary error function for small arguments, $\operatorname{erfc}(x) \approx 1 - \frac{2}{\sqrt{\pi}}x$.

#### A.4.1.4 Optimal Rice parameter

To minimize the expectation of $Z_1$, we solve for $k_{\min}$:

$$k_{\min} = \log_2(\sigma_{\mathcal{R}}) + \log_2\left(\frac{\sqrt{2(\pi + 4\ln^2(2))} - \sqrt{2\pi}}{\ln(2)}\right) \tag{A.21}$$

$$\approx \log_2(\sigma_{\mathcal{R}}) - 0.037 \tag{A.22}$$

Figure A.3 present the results of this formula against the real value of $k$ observed for a Gaussian distribution. **It shows that our approximation is very close to the actual optimal value.**

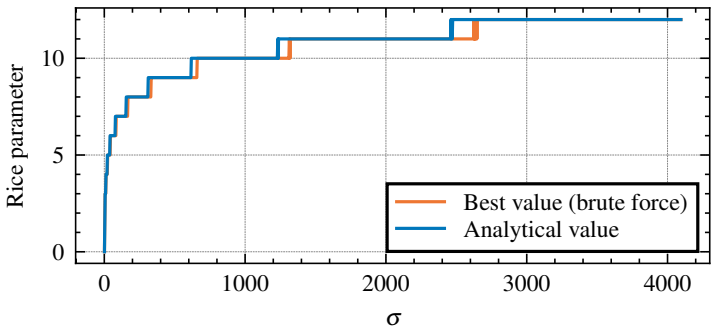



**Figure A.3:** Computed vs optimal value of Rice parameter $k$ for a Gaussian distribution of standard deviation $\sigma$.

For practical purposes, we can use the following approximation:

$$\boxed{k_{\min} = \lfloor \log_2(\sigma_{\mathcal{R}}) - 0.037 \rceil \approx \lfloor \log_2(\sigma_{\mathcal{R}}) \rceil} \tag{A.23}$$

By substituting the optimal parameter back into the expectation of $Z_1$, we find the minimal expected value:

$$\min_k(\mathbb{E}\{Z_1\}) = \log_2(\sigma_{\mathcal{R}}) + 1 + \frac{\sigma_{\mathcal{R}}}{2^{\log_2(\sigma_{\mathcal{R}})-2}\sqrt{2\pi}} - \frac{2}{\pi} \tag{A.24}$$

$$= \log_2(\sigma_{\mathcal{R}}) + 1 + \sqrt{\frac{2}{\pi}} - \frac{2}{\pi} \tag{A.25}$$

$$\boxed{= \log_2(\sigma_{\mathcal{R}}) + c_1} \tag{A.26}$$

Here, $c_1$ is a constant, $\sqrt{\frac{2}{\pi}} - \frac{2}{\pi}$, which adjusts the minimal expected value by a small constant factor.

### A.4.2 Conclusion on Rice coding efficiency for Gaussian distribution

In conclusion, the analysis of Rice coding efficiency for Gaussian distributed residuals reveals several important insights. Rice coding, being a type of entropy coding, is highly effective when the parameter $k$ is optimally chosen. For Gaussian-distributed residuals, the optimal Rice parameter can be approximated by $k_{\min} \approx \lfloor \log_2(\sigma_{\mathcal{R}}) \rceil$, where $\sigma_{\mathcal{R}}$ is the standard deviation of the residuals. This approximation is derived from minimizing the expected value of the encoded length, $\mathbb{E}\{Z_1\}$, under the assumption that the residuals follow a normal distribution.

The key to achieving high compression efficiency lies in accurately estimating the standard deviation of the residuals, as this directly influences the selection of the Rice parameter. When the parameter is chosen correctly, Rice coding can significantly reduce the bit length required to encode the residuals, thus improving the overall compression performance.

Furthermore, the expected minimal bit length to encode the residuals is given by $\log_2(\sigma_{\mathcal{R}}) + c_1$, where $c_1$ is a small constant. This result highlights the efficiency of Rice coding in scenarios where the residuals are normally distributed, providing a clear guideline for parameter selection and performance expectations.

Overall, the analytical approach presented in this appendix offers a practical method for optimizing Rice coding parameters in applications where the underlying data follow a Gaussian distribution. This contributes to the broader understanding and application of Rice coding in various fields, enhancing its utility and effectiveness in data compression tasks.

# Algorithm decision tree

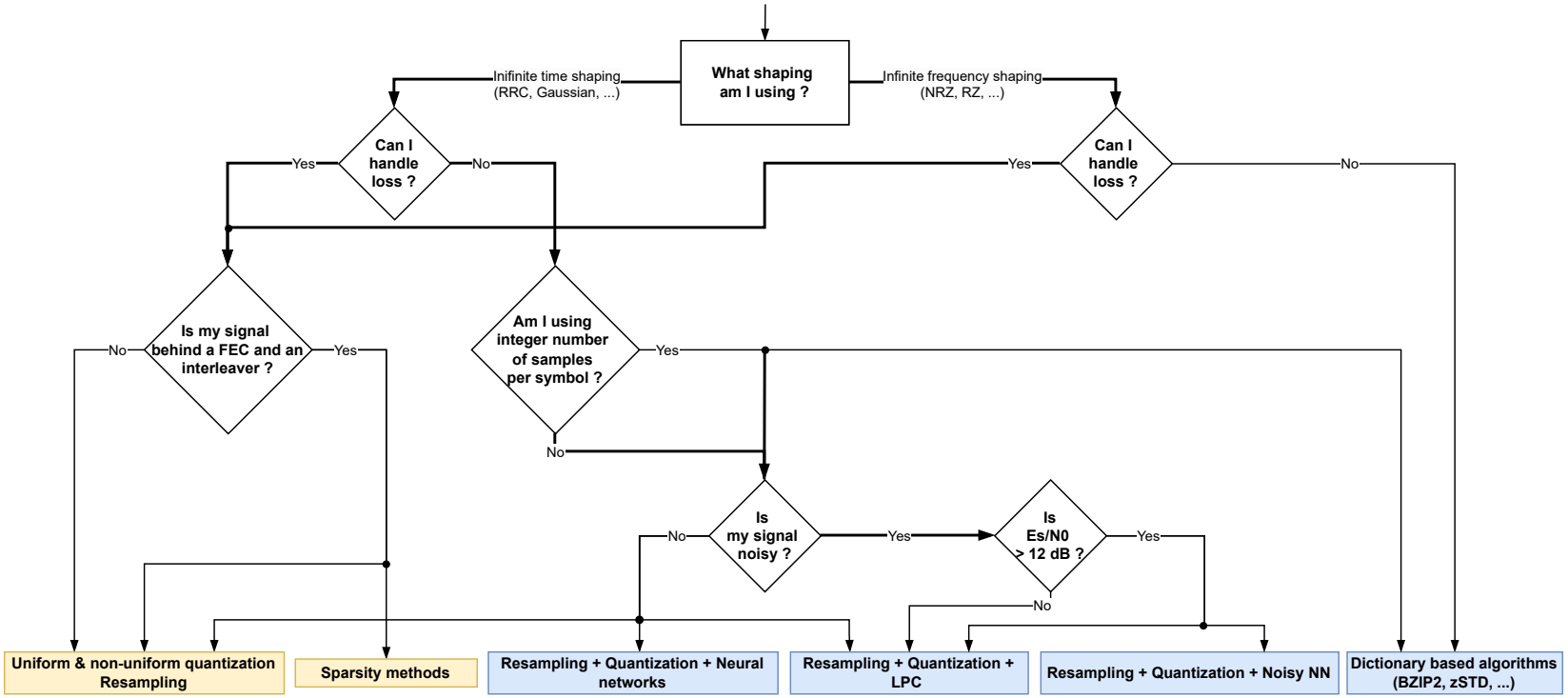



**Figure B.1:** A streamlined decision tree employed to select the appropriate algorithm family for signal compression.

# Compressed sensing architecture

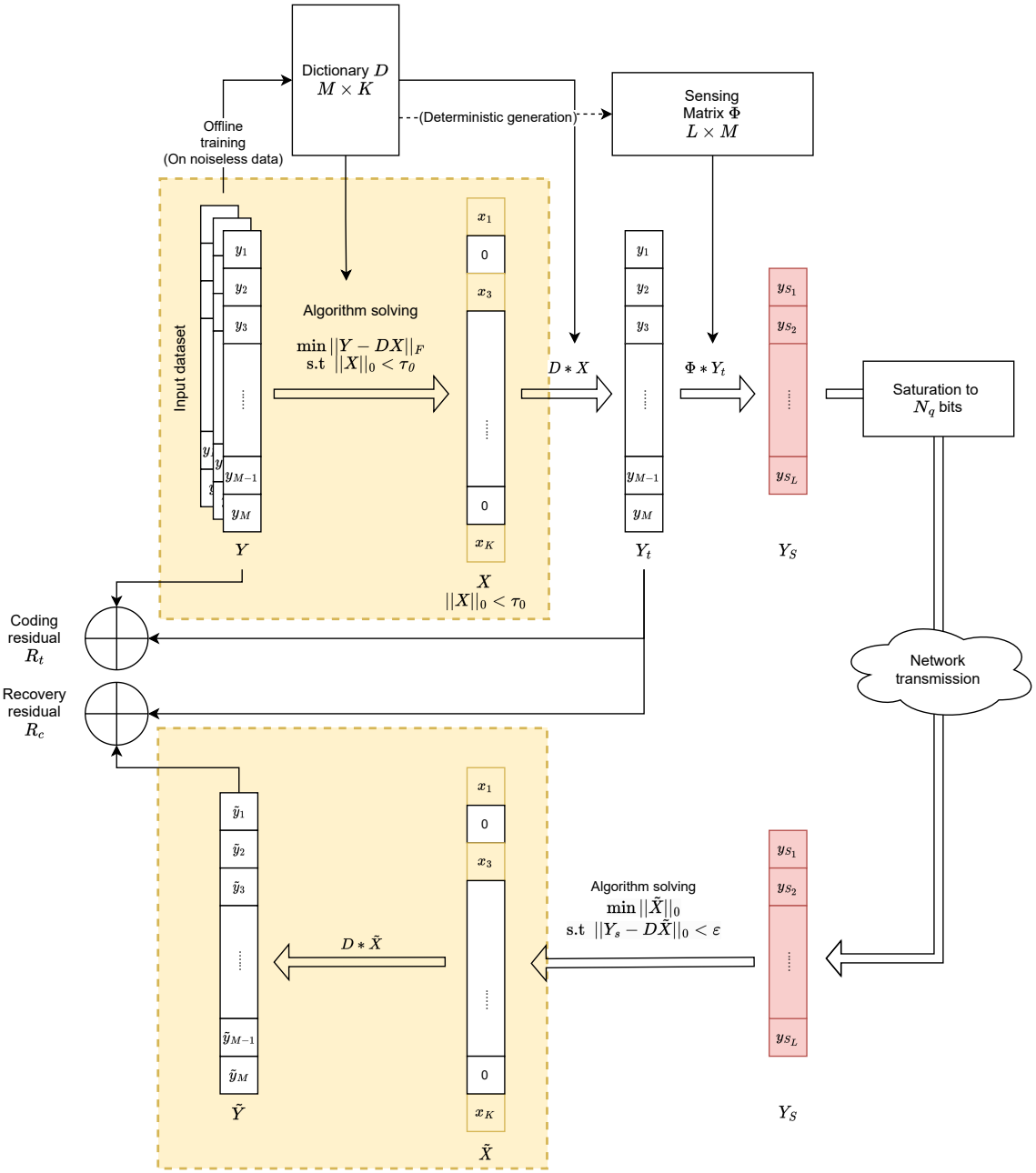



**Figure C.1:** Proposed compression chain for signal compression using *compressed sensing*

# Sparsity compression chain

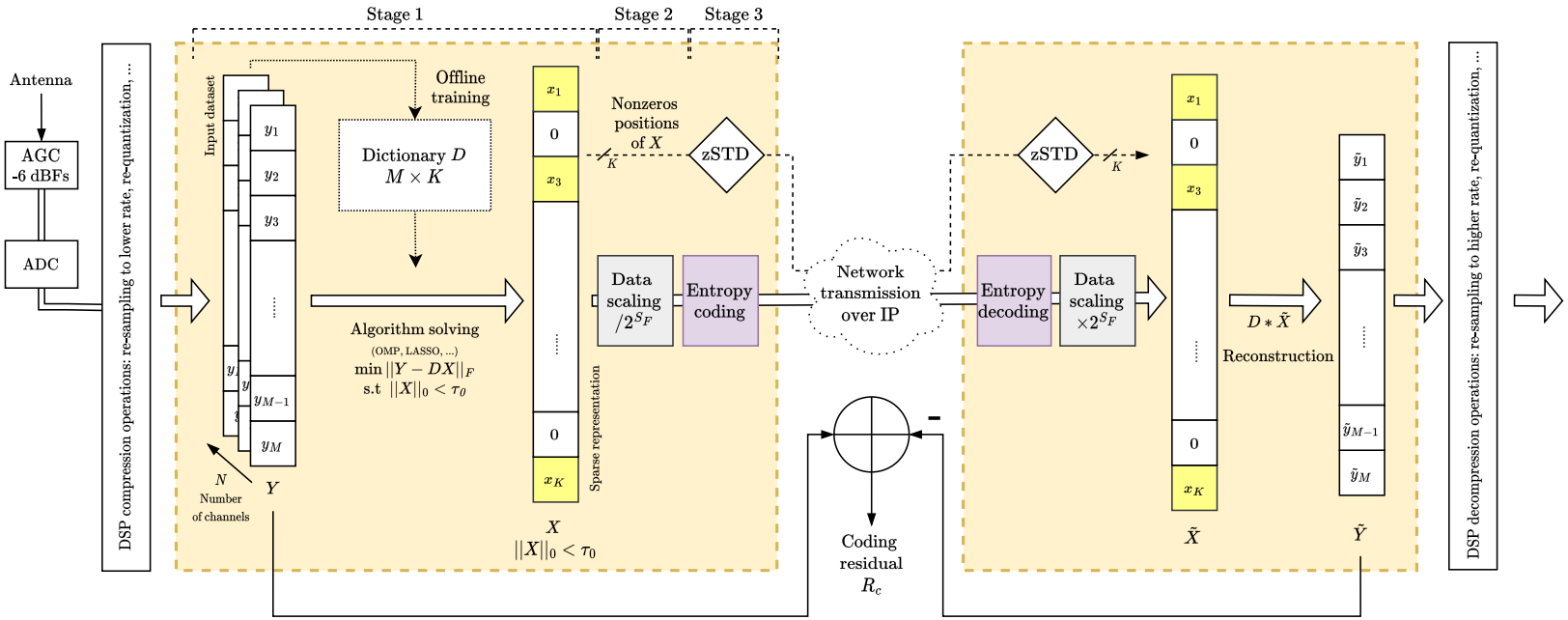



**Figure D.1:** Proposed comprehensive compression chain for signal compression using sparse coding

# Predictive compression chain

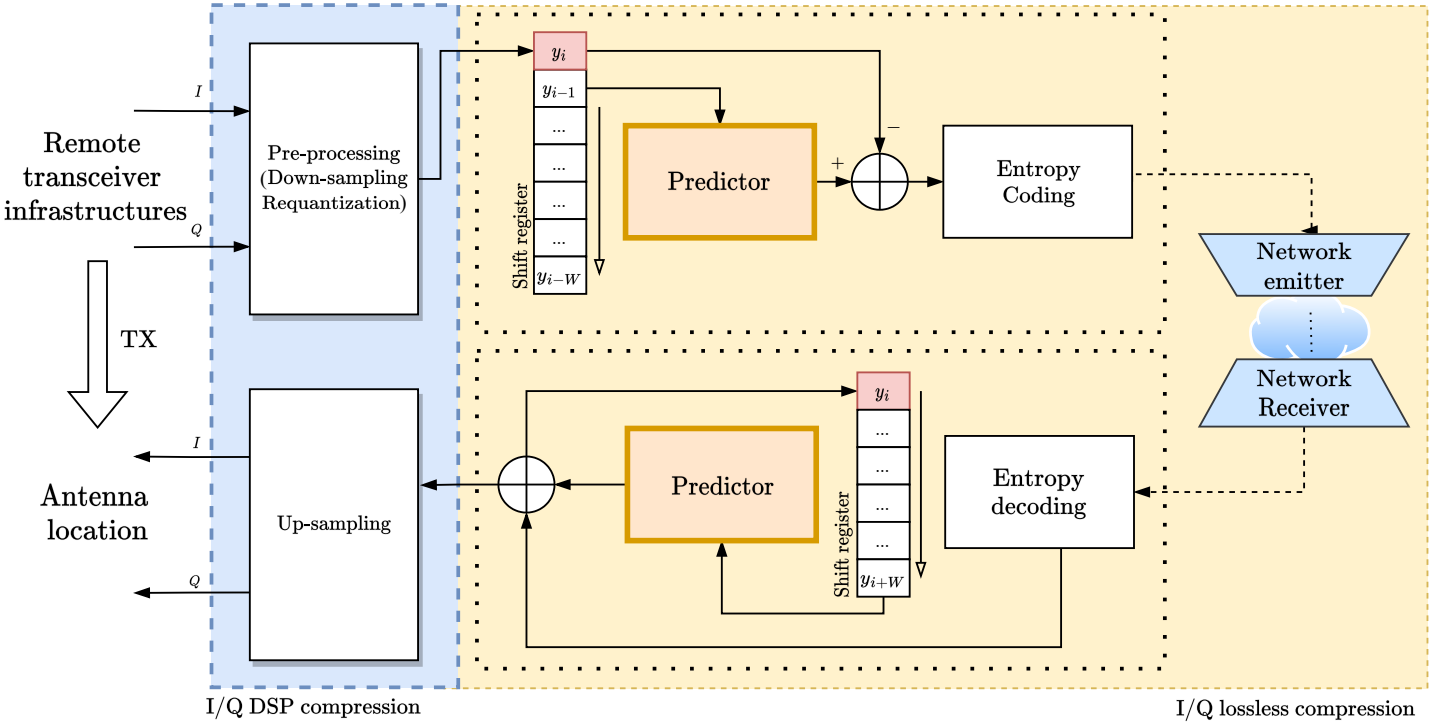



**Figure E.1:** Classical architecture for compression/decompression schema from a remote modulation unit to an antenna using predictors. Entropy coding is usually performed by a Golomb-Rice encoder (Zoomed).

# Résumé en Français

La présente thèse explore la compression de signaux pour les communications satellitaires à des fins de déploiement efficace d'architectures de type Centralized Radio Access Network (C-RAN) pour le segment sol du réseau d'accès satellite. L'objectif principal de cette thèse est de réduire la bande passante nécessaire à la transmission de données sur le segment sol tout en maintenant une haute fidélité des signaux radio-fréquence (RF) échangés entre le sol et un satellite. Ce travail s'inscrit dans un contexte où l'efficacité de la transmission et l'intégrité des données sont cruciales, en abordant spécifiquement les défis posés par la compression des signaux RF en bande de base, comme la complexité matérielle et la robustesse aux variations de conditions de compression.

La thèse exposée a été étudiée, développée et rédigée dans le cadre d'un contrat CIFRE, en collaboration avec *Safran Data Systems* et sous la supervision d'Abdeldjalil AÏSSA-EL-BEY (*Lab-STICC*), Matthieu ARZEL (*Lab-STICC*) et Alain THOMAS (*Safran Data Systems*). Le tout étant encadré dans le cadre de la chaire Pracom.

---

Notre recherche prend pour cadre les récentes évolutions des architectures de réseaux adaptées au segment sol des communications satellitaires, s'inspirant des réussites des réseaux mobiles et en particulier des Centralized Radio Access Network (C-RAN). La transformation technologique actuelle, (Figure E.2), promet une révolution dans la conception et le déploiement des réseaux. En particulier, elle se manifeste par le découplage des fonctionnalités de traitement de l'infrastructure de réception, permettant ainsi une maintenance centralisée, une allocation efficace des ressources et des avantages opérationnels améliorés, spécialement dans le cadre de déploiements à grande échelle.

Ce mouvement est soutenu par des acteurs majeurs tels qu'*Amazon Web Services (AWS)* et *Microsoft Azure*, qui ont introduit le concept de Ground Station as a Service (GSaaS). Ce service a permis à de nombreuses entreprises, notamment dans le domaine de l'Observation de la Terre (EO), de **proposer des services sans pour autant disposer d'infrastructures coûteuses**. Afin d'atteindre l'efficacité et la flexibilité requises par ces architectures avancées, une réévaluation et une adaptation des architectures du réseau de communication s'avèrent indispensables. Cette transformation vise à optimiser le dialogue entre le réseau interconnectant les stations de base, au sol, et les centres de données. Dans ce processus, l'intégration et la considération de nouveaux standards et normes de communication jouent un rôle pivot. Parmi ceux-ci, la norme VITA-49 représente un exemple pertinent, illustrant la manière dont l'industrie cherche à standardiser et à faciliter le transport d'échantillons numérisés au sein de ces architectures remodelées. Ce choix d'exemples met en lumière la nécessité d'une approche globale et cohérente pour la conception des systèmes de communication de demain, visant à améliorer la connectivité et l'efficience dans le traitement et la transmission des signaux.

L'adoption de cette approche, basée sur l'expérience des réseaux mobiles rentables et l'exploitation des avantages des C-RAN, souligne l'**importance de la compression efficace et adaptative des signaux**. En effet, les communications satellitaires, utilisant de très larges bandes passantes (de quelques GHz, comme celles disponibles dans la bande Ka pour les applications EO), requièrent des méthodes de compression efficaces et adaptatives pour réduire la bande passante nécessaire sur le réseau

interconnectant les stations de base/sol et les centres de données.

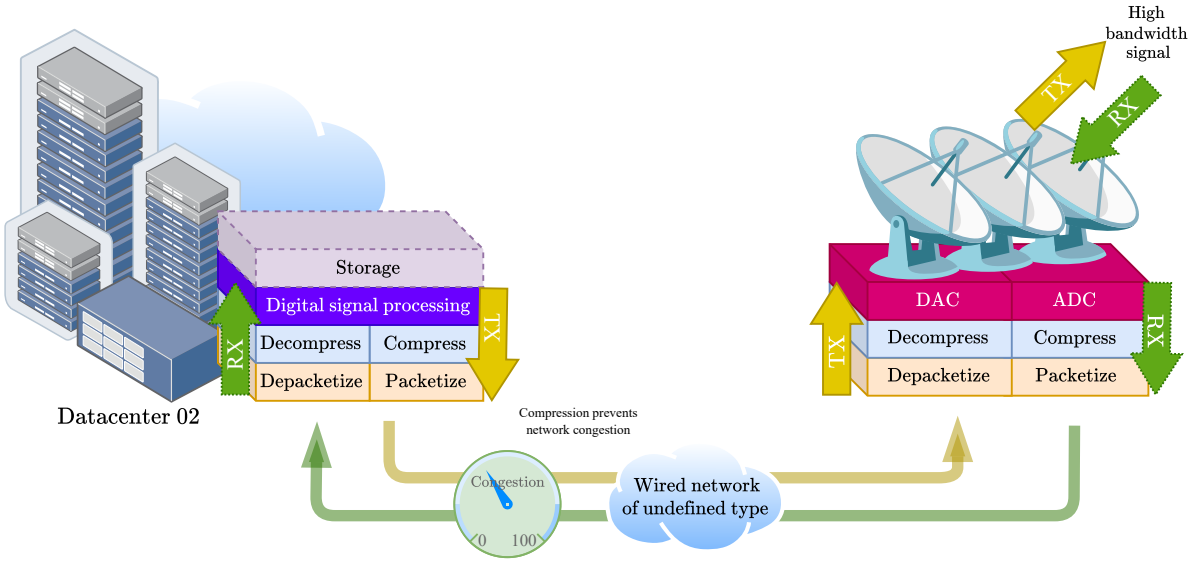



**Figure E.2 :** Architecture d'un réseau te type C-RAN pour le segment sol satellitaire : le couple modulateur-démodulateur (modem) est déplacé du pied d'antenne dans une localisation déportée. Il est alors nécessaire de faire transiter les échantillons bruts entre l'antenne et le modem en utilisant un réseau fibré de haute bande passante.

L'adoption de ces techniques entraîne donc un **inconvénient majeur lié à la consommation de bande passante**. Pour illustrer ce point, prenons l'exemple d'un flux de données ayant un débit initial de 1 Mb/s que l' on veut transmettre d'un datacenter à une antenne déportée sur un autre site. Une fois ce flux modulé en QPSK et échantillonné de façon classique à raison de deux échantillons par symbole, puis quantifié sur 16 bits, la quantité de données nécessaires pour représenter le signal en bande de base – qui inclut les composantes en phase (I) et en quadrature (Q) – devient considérable. Plus précisément, la bande passante requise pour transmettre ces échantillons au démodulateur grimpe à 32 Mb/s, illustrant un **facteur d'expansion significatif de 32**. Bien-sûr, c'est une estimation optimiste, puisque l'idée est de transmettre la bande passante d'un signal RF qui n'est pas complètement dédiée à de la donnée utile et qui va exploiter une forme d'onde qui requiert un supplément de bande. Un supplément sera dû encore à l'overhead des protocoles comme VITA-49.

Pour contrer cette augmentation drastique de la bande passante, différentes stratégies peuvent être envisagées. Dans le cadre de notre travail, nous avons opté pour **l'exploration de techniques de compression** qui tirent parti de notre compréhension approfondie des caractéristiques temporelles, fréquentielles et statistiques des signaux émis et reçus. L'objectif est de compresser efficacement les signaux avant leur transmission sur le réseau, dans le but de réduire considérablement le facteur d'expansion et, par conséquent, la charge sur le réseau fibré qui relie antenne et datacenter. L'enjeu lié à la bande passante à compresser est de plus double : il faut concevoir des algorithmes qui soient d'une part efficaces pour réduire la charge en débit sur le segment sol et d'autre part adaptés au débit (extrême) des données à traiter.

La thèse se concentre ainsi sur une analyse des méthodes de compression pour les signaux **quantifiés en bande de base**, abordant l'étude sous plusieurs angles critiques et prenant en compte plusieurs considérations :

1 · Une exploration détaillée de la voie montante, ou voie TX, qui relie le modulateur à l'antenne pour la transmission des signaux. Cette portion de la communication se caractérise par l'absence de bruit, ce qui la rend idéale pour l'étude de la compression des signaux dans des conditions

optimales.

2 · Inversement, nous examinons la voie descendante, ou voie RX, qui conduit les signaux de l'antenne au démodulateur. Cette dernière est intrinsèquement plus complexe en raison des divers phénomènes qui la perturbent, tels que le bruit du canal de transmission. Une attention particulière est accordée à l'intégration de ces facteurs dans l'évaluation de la performance de compression.

3 · Une distinction claire entre les techniques de compression **sans perte** et celles **avec perte** est établie. Cette séparation est essentielle pour évaluer les compromis entre le niveau de compression souhaité et l'impact sur la qualité des données, un équilibre crucial dans le contexte des communications satellitaires où l'intégrité des données est de la plus haute importance.

   1 · Pour ce qui est de la compression avec pertes, notre étude s'oriente vers la **définition et l'utilisation de métriques précises** pour **quantifier l'impact de la perte introduite**. Nous développons des stratégies permettant un ajustement précis de cette perte en fonction des exigences spécifiques de la mission et de la communication.

   2 · Nous procédons également à une catégorisation des différents types de signaux, ou plus exactement des formes d'onde, selon leur susceptibilité à la compression par divers algorithmes. Cette classification aide à identifier les techniques de compression les plus efficaces en fonction du type de signal.

3 · Enfin, la thèse explore les défis et les stratégies liés à l'implémentation de ces méthodes de compression sur différents supports matériels, tels que les FPGA et les CPU. Cette partie de la recherche est cruciale pour le **déploiement réel des technologies développées**, et aborde les questions d'efficacité et de viabilité de ces méthodes dans un contexte opérationnel.

Cet examen approfondi des méthodes de compression des signaux en bande de base est conçu pour fournir une compréhension globale des défis, des solutions et des compromis impliqués dans les communications par satellite, en soulignant les spécificités techniques et pratiques de chacune des voies de transmission étudiées.

---

Dans le cadre de cette thèse, nous explorons diverses facettes de la compression de données numériques, mettant l'accent sur les techniques de compression classiques, les méthodes basées sur des dictionnaires, et la compression prédictive. Chacune de ces approches offre des avantages uniques et répond à des défis spécifiques dans le domaine de la compression des signaux pour les communications satellitaires.

Les techniques de compression classiques, qui forment le socle de notre étude, sont issues des pratiques standardisées dans le traitement du signal numérique : la quantification, qu'elle soit uniforme ou non, et le rééchantillonage. Ces méthodes, bien établies et largement utilisées dans l'industrie, servent de point de référence pour évaluer les nouvelles approches de compression que nous développons. Elles jouent un rôle crucial dans la compréhension des limites et des potentialités des technologies actuelles de compression et fournissent un cadre pour discuter des améliorations possibles.

En progressant vers des techniques plus avancées, nous nous intéressons particulièrement aux méthodes de compression basées sur des dictionnaires, représentant une avancée significative pour la compression avec pertes. Cette approche, axée sur la parcimonie (sparsity), vise à représenter le signal de manière efficace en utilisant un nombre réduit d'éléments actifs ou 'atomes' issus d'un dictionnaire préétabli. L'objectif est de construire une représentation parcimonieuse du signal qui minimise la quantité de données nécessaires tout en contrôlant la perte de qualité. En développant des algorithmes capables

d'identifier les composantes les plus significatives du signal et de les encoder efficacement, nous pouvons atteindre des taux de compression élevés sans sacrifier de manière critique la fidélité des données. Cette partie de la recherche illustre comment la sélection minutieuse et l'assemblage des atomes peuvent conduire à une compression des données à la fois efficace et économe, offrant ainsi des solutions pratiques pour les défis posés par la gestion de la bande passante dans les communications satellitaires. *Cette partie à donné lieu à un article actuellement à l'étude à la conférence EUSIPCO 2024.*

Enfin, notre exploration se tourne vers la compression prédictive, une méthode sans perte qui se base sur la prédiction du signal à partir de son historique. L'**approche consiste à utiliser les données passées du signal pour prédire les valeurs futures et à compresser efficacement les écarts entre les valeurs prédites et les valeurs réelles**. Ce type de compression est particulièrement pertinent pour les applications nécessitant une préservation intégrale des données, comme dans certains contextes de communications satellitaires. Notre travail contribue significativement à ce champ en développant des algorithmes de prédiction plus précis et des méthodes de compression plus efficaces pour gérer ces écarts. Nous proposerons notamnent une approche novatrice basée sur des réseaux de neuronnes. *Cette recherche a été présentée sous forme d'un article lors de la conférence ICETC 2023. Un autre article issu de ce travail est actuellement en cours d'évaluation pour une publication dans une revue scientifique.*

En conclusion, cette thèse apporte des contributions significatives à la théorie et à la pratique de la compression de signaux en bande de base, avec des implications pratiques importantes pour les systèmes de communication satellitaire numériques. Les techniques et méthodes développées offrent des voies pour réduire la bande passante nécessaire, tout en maintenant l'intégrité et la fiabilité de la transmission des données. Nos travaux ouvrent la voie à des améliorations substantielles dans la gestion des données dans des environnements exigeants et pourraient servir de fondement à de futures recherches et innovations dans le domaine de la compression de données.

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

**Titre :** Étude et développement de techniques de compression pour les signaux de télécommunications en bande de base

**Mot clés :** communications numériques, compression IQ, codage parcimonieux, réseaux de neurones, RF sur IP

**Résumé :** Cette thèse examine la compression de signaux pour améliorer l'efficacité de la bande passante dans les communications satellitaires, en se concentrant sur les architectures Cloud Radio Access Networks (C-RAN) pour le segment sol. Réalisée sous contrat CIFRE avec Safran Data Systems et encadrée par le Lab-STICC, cette recherche explore les techniques de compression des signaux RF en bande de base, essentielles pour maintenir une haute fidélité des données transmises entre les stations terrestres et les satellites.
L'étude s'appuie sur des avancées comme le Ground Station as a Service (GSaaS), qui permet une gestion optimisée des ressources. Elle aborde spécifiquement les défis liés à la compression efficace des larges bandes passantes utilisées, nécessitant des techniques innovantes pour réduire la charge sur les réseaux terrestres sans compromettre la qualité des signaux.
Les méthodes de compression sans perte et avec perte sont évaluées, avec une attention particulière aux techniques de compression basée sur des dictionnaires pour leur efficacité en termes de parcimonie et aux méthodes prédictives pour leur capacité à minimiser les écarts entre prédictions et valeurs réelles. Cette recherche illustre un potentiel significatif pour améliorer la gestion des données dans les communications satellitaires, offrant des solutions viables pour les défis actuels et futurs de la compression de données.

**Title:** Analysis and design of baseband signal compression methods for telecommunications

**Keywords:** digital communications, IQ compression, sparse coding, neural networks, RF over IP

**Abstract:** This thesis investigates signal compression to enhance bandwidth efficiency in satellite communications, focusing on Cloud Radio Access Network (C-RAN) architectures for the ground segment. Conducted under a CIFRE contract with Safran Data Systems and supervised by Lab-STICC, this research explores the techniques of RF baseband signal compression, crucial for maintaining high fidelity of data transmitted between terrestrial stations and satellites. The study leverages advances such as Ground Station as a Service (GSaaS), which facilitates optimized resource management. It specifically addresses the challenges associated with efficiently compressing the wide bandwidths used, which requires innovative techniques to reduce the load on terrestrial networks without compromising signal quality.
Lossless and lossy compression methods are evaluated, with particular emphasis on dictionary-based compression techniques for their efficiency in sparsity, and predictive methods for their ability to minimize discrepancies between predicted and actual values. This research demonstrates significant potential to improve data management in satellite communications, offering viable solutions for current and future data compression challenges.