Strategies for Delay-Limited Source-Channel Coding

THÈSE N^{O} 4747 (2010)

PRÉSENTÉE LE 9 JUILLET 2010

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

LABORATOIRE DE COMMUNICATIONS MOBILES

PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Marius KLEINER

acceptée sur proposition du jury:

Prof. R. Urbanke, président du jury
Prof. B. Rimoldi, directeur de thèse
Prof. M. C. Gastpar, rapporteur
Prof. T. A. Ramstad, rapporteur
Prof. E. Telatar, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2010

Acknowledgments

The one person who has seen me through the not always smooth path from the beginning to the end of this thesis was my advisor, Bixio Rimoldi. With my strong tendency for distraction I certainly haven't made it any easier for him. Nevertheless he kept me going, always coming up with suggestions about what to do next, and applying ever so slight a pressure to get me to finally produce results. For all of this, and for becoming a good friend, I am extremely grateful to him.

I will never forget our collaborations when it came to writing papers. Better than anyone, Bixio understood that the quality of a paper equals content times form. It would often happen that after several hours of reflection I had found what I thought was clearly the rhetorically perfect way to express an idea. When Bixio would then tell me that what I wrote probably wouldn't be clear to a reader, I would be on the verge of declaring that I was going to withdraw my name from the paper if this sentence didn't stay in. Of course in the end he turned out to be right, and I have only learned from it.

This thesis has also greatly profited from the help of Emre Telatar. He gave only a few hints here and there, but these hints turned out to be of great value and directly led to some of the main ideas set forth here. Moreover, it was always a great pleasure to discuss with Emre such diverse subjects as books, restaurants, religion, or typography (the list would go on). Thanks Emre!

Rüdiger Urbanke, fellow early riser, was the driving force behind many of our more creative endeavors, in particular our home-brewed IPG movies. I am thankful to him for this, for his unrivaled sense of humor, and for introducing me to unicycling.

Many thanks to the members of my thesis committee: Emre, Rüdiger, and especially Michael Gastpar and Tor Ramstad, who took the time and effort to travel from far away. Michael also deserves thanks for his own thesis, which was a great source of inspiration to me from the start.

The great atmosphere during the last five years in the Information Processing Group is due to its present and past members. Besides the ones already mentioned, I must thank Suhas, Christina, Olivier, Nicolas, Emmanuel; our secretaries Françoise, Muriel, and Yvonne; our sysadmin Damir, the kindest person in the lab, for his enthusiasm to help out with even the smallest computer

problem (thanks again for your indulgence when I had my new MacBook Pro stolen); Giovanni; former LCM members Peter, Nicu, and Tarik; the gang of Indians: Satish, Dinkar, Sanket, Vish, and my brother-from-another-mother Shrinivas; furthermore Abdel, Alberto, Amin, Aslan, Christine, Cyril, Emre A., Etienne, Hamed, Lorenzo, Mahdi J., Marc, Shirin, Sibi, Soheil, Stéphane B., and Stéphane M.

Quite literally the closest people are those one shares an office with. I had the great luck and pleasure to have Ayfer as my officemate in INR036 (baptized "the ministry of silly talks" by Olivier) for four years. Days if not weeks of lost productivity will no doubt be attributed to our arguments about politics, religion (a constantly recurring subject, as others can testify), various medical conditions, as well as the French and the Turkish language (of the latter Ayfer taught me more than just the basics). I am thankful to Ayfer for remaining a close personal friend even though I've strained her patience enough over the years ("Can I ask you just a quick question…" comes to mind). During my last year at EPFL, Mohammad proved to be an equally pleasant officemate and a like-minded individual (though our office hours did not often coincide).

The list of people who have made my time in Lausanne so enjoyable is long. At the risk of forgetting someone, my thanks go to (in no particular order): Jérémie, for sharing my sense of humor and for the greatest movie collaborations EPFL has ever seen; Harm, co-inventor of the early lunch and a great friend throughout; Eren, for an endless supply of movie quotes recited with perfect accuracy; Maria and Denisa – I can't put it in words!; Nino, for teaching me Sicilian mafia hand gestures; Bertrand M., for lessons in Alsatian language and culture (one day I'll have that *Choucroute Formidable*); Florence and Vojislav, for inviting me to their wedding after all (sorry I couldn't make it in the end); Klaske, for introducing me to kayaking and for the afterwork beers at Sat; Luigi, my original friend-friend, for the Calabrian delicacies; Claudia, for the German-Italian tandem; Karin, for inviting us to the Heurigen and to the fanciest ball in Vienna; Jasper, an excellent colleague for too short a time, for being a great host in Enschede; Maya, for challenging my skepticism; Simon, for teaching me you can't eat Weisswurst after the midday bell has rung; Gizil, for more Turkish lessons; Daniel, for sharing his profound knowledge of all things programming; Zafer African, for organizing the best Turkish parties; Sébastien and Julien, for co-organizing the beer tasting; Andi, Gion, Manu, Nico, Piotr, and Stefan for the first five years at EPFL and for remaining very good friends to this day; also Albrecht, Ali, Bertrand N. N., Brammert, Davor, Dominique, Elena, Emily, Evelina, German, Hossein, Irina A., Irina B., Klaas, Mahdi C., Masoud, Nathan, Nicolas L.-B., Olivier B., Pooya, Ruud, Shahram, Sophie, Tobi, Veronica, Willem-Jan; the climbing group of the Club Montagne; the Kayak Club Lausanne; the cities of Budapest, Firenze, Barcelona, Gaborone, Swakopmund, Capetown, Roma, and München for the memorable holidays as well as The Great Escape, Holy Cow, Lausanne Moudon, and Denis Martin for my gastronomic and alcoholic welfare in Lausanne.

I would never even have made it to Lausanne were it not for my family,

whose generous support I could count on since I can remember. Thanking them here will never be enough for all they have done for me.

Last but absolutely, definitely, categorically not least, my deepest gratitude goes to Krem and to Ghid, for sharing some of the greatest moments during these last five years.

Abstract

In point-to-point source-channel communication with a fidelity criterion and a transmission cost constraint, the region of achievable cost and fidelity pairs is completely characterized by Shannon's separation theorem. However, this is in general only true if coding of arbitrary complexity and delay is admitted. If the delay is constrained, the separation theorem only provides an outer bound to the achievable cost/distortion region, and the exact shape of this region is in general not known.

The first part of this thesis studies source-channel communication when neither a required average fidelity nor a cost constraint are specified, but when the goal is to maximize the *ratio* of fidelity to cost. It is shown how the maximal ratio relates to existing quantities such as the capacity per unit cost. Finally, necessary and sufficient conditions are derived to test whether a given system operates at this maximal ratio and when this is possible using a single-letter code.

The second part of the thesis studies communication of continuous-valued sources over the additive white Gaussian noise channel when only a *single* source symbol is to be encoded at a time. In particular, the case is considered where several uses of the channel can be made for each source symbol. Inspired by communication with feedback, a simple communication strategy combining quantization and uncoded transmission is derived and analyzed. It is shown that this strategy achieves a mean squared error that performs as well as any known communication strategy that encodes a single source symbol at a time. On the other hand, it is strictly suboptimal in the sense that the gap (in dB) between the achievable signal-to-distortion ratio (SDR) and the best SDR achievable without a delay limit grows with increasing signal-to-noise ratio.

The thesis turns to a more practical subject in its last part. The case is made why object-oriented programming is particularly suited to implementing simulations. As a proof of concept, a complete implementation of an object-oriented simulator for source-channel coding is presented that allows for rapid development and analysis of arbitrary communication strategies.

Keywords: discrete-time memoryless sources, discrete-time memoryless channels, joint source-channel coding, bandwidth expansion, delay, feedback, capacity per unit cost, fidelity per unit cost, simulation, object-oriented programming

Kurzfassung

Shannons Separationstheorem liefert eine exakte Charakterisierung der Region der erreichbaren Qualität/Kosten-Paare für die Übertragung einer Informationsquelle über einen rauschenden Kanal. Dies aber nur für den Fall, dass eine beliebig grosse Verzögerung in Kauf genommen wird. Ist die tolerierbare Verzögerung begrenzt, so erhält man durch das Separationstheorem lediglich eine äussere Schranke dieser Region; ihre genaue Form ist im Allgemeinen nicht bekannt.

Der erste Teil dieser Dissertation befasst sich mit der Übertragung einer Quelle über einen Kanal, wenn es gilt, das *Verhältnis* der Wiedergabequalität zu den Übertragungskosten zu maximieren. Es wird eine Verbindung zwischen dem höchstmöglichen solchen Verhältnis und existierenden informationstheoretischen Grössen hergestellt. Zudem liefert dieser Teil exakte Bedingungen, unter welchen ein Punkt-zu-Punkt-Kommunikationssystem das Maximum erreicht.

Der zweite Teil der Dissertation befasst sich mit der Übertragung einer stetigwertigen Quelle über einen Kanal mit additivem weissem gaussschem Rauschen, wenn jedes Quellensymbol separat kodiert werden muss. Es geht im Speziellen um den Fall, dass für jedes Quellensymbol mehrere Übertragungen durchgeführt werden können. Dazu wird ein einfaches Verfahren vorgestellt, welches von einem bekannten Rückkopplungsverfahren inspiriert ist und asymptotisch eine mittlere quadratische Abweichung erreicht, die so gut wie die besten bekannten Verfahren ist. Andererseits reicht die Leistung dieses Verfahrens nicht an die theoretischen Schranken, welche ohne Beschränkung der Verzögerung erreicht werden können, heran. Weiter wird gezeigt, dass eine Erweiterungen des Verfahrens auf grössere Blocklängen derselben Leistungsbeschränkung unterliegt.

Der dritte und letzte Teil greift ein etwas handfesteres Thema auf, namentlich die Computersimulation von Übertragungsverfahren. Die These, dass sich objektorientierte Programmierung besonders für Simulationen eignet wird aufgestellt und durch die Präsentation eines kompletten Simulators für Punkt-zu-Punkt-Übertragung untermauert. Dieser Simulator erlaubt es, beliebige Kommunikationsverfahren besonders schnell zu implementieren und zu analysieren.

Stichwörter: zeitdiskrete gedächtnislose Quellen und Kanäle, kombinierte Quellen-Kanal-Kodierung, Erweiterung der Bandbreite, Verzögerung, Rückkopplung, Kapazität pro Kosteneinheit, Wiedergabequalität pro Kosteneinheit, Simulation, objektorientierte Programmierung

Contents

Introduction

The task of the communication engineer is to process information and to transform it such that it can be transmitted reliably across an unreliable medium. The information can be bits from a computer file, an audio signal recorded from a microphone, images captured by a TV camera, or any other physical signal. The unreliable medium is for example a wire undergoing corruption by thermal noise at the receiver, a wireless link subject to interference from other transmitters, or a hard disk with occasional read errors.

The processing of information at the transmitter is called *encoding*. It is usually done on *blocks* of data. For example, one kilobyte of a file is encoded at a time, or one second of recorded voice. The more information is encoded at once, the better the reconstruction quality one can expect at the receiver. This is because a longer sequence of data is more likely to contain a *structure* that can be efficiently exploited.

However, encoding long blocks of data at a time causes delay, as the encoder has to wait until the source has produced enough data. Suppose your cell phone encoded ten seconds of audio at once. This may increase the quality of your voice at the receiver, but your conversation partner will hardly care about this if she hears your voice only ten seconds after you have started talking.

The subject of this thesis is communication under a strict delay constraint. In particular, the following themes are addressed.

Themes

- **Minimal-Delay Source-Channel Coding.** If coding delay and complexity is unconstrained, the region of achievable cost and distortion pairs for a given source, channel, cost measure and distortion measure is completely characterized by Shannon's separation theorem. On the one hand this theorem provides an outer bound to the achievability region; on the other hand it shows that any point in this region can be achieved using separate source and channel coding. Under a delay constraint, however, only the outer bound applies, and the exact achievability region is no longer known.

  This thesis considers the strictest form of a delay constraint, where a

single source symbol is to be encoded at a time. The focus is on the situation where the channel accepts inputs for transmission at a higher rate than the source produces them (this is sometimes called *bandwidth expansion*). The analysis is concentrated on the Gaussian channel with an input power constraint and the squared error distortion measure.

- **Optimal Cost–Fidelity Ratio.** The source-channel communication problem represents a tradeoff between cost and fidelity. Usually, the goal is either to maximize the fidelity obtained for a fixed transmission cost constraint, or to minimize the cost required to achieve a given fidelity of reproduction. The bigger the cost one is willing to pay, the bigger a fidelity one can obtain.

  An alternative point of view is to look at the *ratio* between cost and fidelity. This is the approach taken in Chapter 2 of this thesis, where the question of interest is how to characterize the largest fidelity per cost.

- **Simulation.** For the communication engineer, simulations are a valuable tool. For all their limitations, they not only help to quickly test new ideas and to determine which ones deserve a more thorough theoretical analysis, but they also allow one to perform a reality check on theoretical derivations.

  The ideas to be simulated are in most cases simple enough that an engineer can write a complete simulator from the ground up. Over time, however, simulation code often becomes cluttered up from countless modifications in countless places, which gradually decreases clarity and productivity. Sometimes the original code changes so much that it is no longer possible to reproduce previous results.

  In Chapter 5, this thesis explores how the object-oriented programming paradigm helps to manage the complexity of simulators while at the same time allowing rapid, undistracted implementation of new communication strategies.

Contributions

1. **Characterization of Optimal Fidelity–Cost Ratio.** The three problems of source coding with a fidelity criterion, channel coding with an input cost constraint, and reproducing a source across a noisy channel all face a similar tradeoff between resource and performance. Of particular interest is the operating point with the highest performance per resource. In the case of channel coding, channel input cost is traded for rate, and the optimal tradeoff corresponds to the *capacity per unit cost*, a subject that has been well studied in the past. This thesis defines the equivalent notions of *fidelity per unit rate* for the source coding problem and *fidelity per unit cost* for the end-to-end source-channel communication problem and shows how they relate.

2. **Matching Conditions for Fidelity Per Unit Cost.** It is known that when the channel transition distribution, the input distribution, and the input cost measure are matched in a particular way then the channel operates at capacity. Here a refined condition is provided under which the channel operates at capacity per unit cost. The condition under which a source operates at the rate-distortion function is refined in a similar way. When combined, these refined conditions yield a necessary and sufficient condition for a source-channel communication system to operate at fidelity per unit cost.

3. **Asymptotic Performance of a Minimal-Delay Communication Strategy.** A simple hybrid transmission strategy to transmit a single continuous-valued source symbol across n uses of a Gaussian channel is analyzed. It consists of repeatedly quantizing the source symbol, transmitting the quantizer outputs in the first $n-1$ channel uses, and sending the remaining error uncoded in the n^{th} channel use. This thesis provides the first exact characterization of the best asymptotic behavior of this strategy at large signal-to-noise ratios. It is shown that the asymptotic performance (signal-to-distortion ratio) achieved by the hybrid strategy is strictly bounded away from that achievable without a delay constraint, and that the gap (in dB) to the optimum increases with increasing signal-to-noise ratio.

4. **Extensions of Minimal-Delay Communication Scheme.** Two extensions to the hybrid scheme are provided. The first extension is to encode several source symbols at a time using vector quantization with lattices. The second extension is to encode not one, but k source symbols into n channel uses, where $1 < k < n$. It is found that for a fixed ratio k/n, these extensions are subject to the same asymptotic performance limits (at large SNR) as when a single source symbol is encoded at a time, no matter how large k and n are.

5. **Connection to Feedback.** It is well known that a Gaussian source can be communicated optimally with minimal delay across n uses of a Gaussian channel if the encoder has access to perfect feedback from the receiver. In this work, the use of the hybrid communication scheme sketched above is justified by drawing on insights from the case with feedback. Furthermore, it is shown that there are clear parallels between this point of view and the more traditional way of analyzing minimal-delay source-channel codes geometrically.

6. **An Object-Oriented Source-Channel Coding Simulator.** A full, workable implementation of a joint source-channel coding simulator in MATLAB is presented. It allows rapid testing of ideas, while its structure is such that the code remains clean even when many different configurations exist.

Outline

Chapter 1 introduces the fundamentals of the source-channel communication problem and defines the relevant performance measures as well as their theoretical limits. This is followed by a review of the general conditions required for a system to operate at these limits, i.e., on the border of the region of achievable cost and distortion pairs. In addition, the "measure matching" perspective is reviewed, where an optimal system is characterized in terms of a *match* between the various quantities of the system.

Chapter 2 establishes an upper bound to the average fidelity per cost of a point-to-point source-channel communication system and shows that this bound is tight. It then extends the measure matching point of view to this problem, yielding necessary and sufficient conditions for a communication system to operate at the maximal fidelity per cost ratio.

Chapters 3 and 4 deal with delay-limited block codes for the Gaussian channel. If one channel use is available per source symbol, it is well known that uncoded transmission (with minimal delay) achieves the same performance as the best codes with unconstrained delay. If more than one value can be transmitted across the channel per source symbol, a similarly simple optimal scheme exists only if the encoder has perfect feedback. These optimal schemes are revisited in **Chapter 3**, and it is shown how uncoded communication can be exploited even in the absence of feedback. This results in an alternative justification for a well known hybrid communication strategy combining quantization and uncoded transmission.

This hybrid strategy is described in detail in **Chapter 4**, which then derives asymptotic upper as well as lower bounds on the achieved mean squared error as a function of the SNR and shows that they coincide. The hybrid communication scheme is extended to two more general cases, and the respective asymptotic performance is given. The last section of the chapter looks ahead and suggests possible directions to obtain a more general characterization of the performance of minimal-delay schemes.

Chapter 5 is of a more practical nature. It argues that object-oriented programming is particularly suitable for implementing simulations and presents a complete simulator for joint source-channel coding that allows for fast and easy testing of arbitrary point-to-point communication schemes.

Lastly, **Chapter 6** presents conclusions and possible directions for future research.

Fundamentals of Source-Channel Communication

1

Source-channel communication under a delay constraint is a rather marginal topic in information theory. The main reason is perhaps that two pillars of information theory, Shannon's source coding and channel coding theorems, are asymptotic results and the codes they use are not allowed if one is restricted to operate on sequences of short length.

This chapter has two goals. The first goal is to introduce the source-channel communication problem in its generality. The second goal is to quote the relevant previous work in order to set the stage for the chapters ahead.

Section 1.1 starts the chapter by introducing the elements that make up the source-channel communication problem and by defining the performance criteria of interest. The known fundamental limits for these criteria are then reviewed in Section 1.2. Section 1.3 considers the conditions for a given communication system to achieve the theoretical performance limits. Furthermore, this section cites results from previous work that characterize an optimal communication system in terms of a "match" between the quantities that make up the system. Lastly, Section 1.4 provides a preview of the remaining chapters and positions these chapters within the larger framework.

1.1 Problem Set-Up

This thesis is about point-to-point communication of a memoryless source across a memoryless channel. In its most general form, this problem is made up of the following six elements, displayed schematically in Figure 1.1.

- A discrete-time memoryless *source* with distribution P_S, producing a source symbol every τ_s seconds.

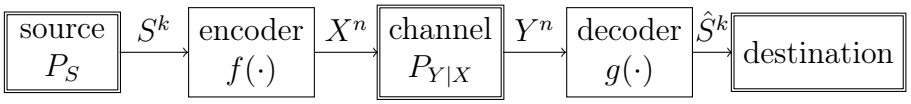


Figure 1.1: A general memoryless point-to-point communication system.

- A discrete-time memoryless *channel* with transition distribution $P_{Y|X}$, accepting an input symbol for transmission every τ_c seconds.
- An *encoder* function f that maps a block of k source symbols $S^k = (S_1, \ldots, S_k)$ into n channel input symbols $X^n = (X_1, \ldots, X_n)$.
- A *decoder* function g that maps a block of n channel output symbols $Y^n = (Y_1, \ldots, Y_n)$ into k source estimates $\hat{S}^k = (\hat{S}_1, \ldots, \hat{S}_k)$.
- A *cost measure* $\rho(x)$ that assigns a transmission cost to each channel input symbol, and a *distortion measure* $d(s, \hat{s})$ that assigns a reconstruction "badness" to every pair of source symbol and estimate.

To match the number of channel inputs produced by the encoder to the rate at which the channel accepts them, k and n must satisfy $n/k \leq \tau_s/\tau_c$. The communication consists thus of identical rounds of transmission of length $k\tau_s$ (or $n\tau_c$).

If the discrete-time source and channel represent an underlying continuous bandlimited source and channel, the sampling theorem relates τ_s and τ_c to the respective bandwidths. If $\tau_s = \tau_c$, one therefore says that the source and channel are *bandwidth matched*. Correspondingly, a code is said to be *bandwidth matched* if $k = n$, to be a *bandwidth expansion* code if $k < n$, and to be a *bandwidth compression* code if $k > n$.

The source and the channel, together with the encoder and decoder, imply a joint distribution of the tuple $(S^k, X^n, Y^n, \hat{S}^k)$. Depending on the encoder, the sequence of channel inputs X^n may not be identically distributed; similarly, the marginal (joint) distribution of the source/estimate pairs $(S_i, \hat{S}_i)$ may not be the same for all i. The average cost and distortion of a communication system are therefore defined as the empirical average over a block of channel inputs and source symbols, respectively.

Definition 1.1. *The* average channel input cost *incurred by a memoryless point-to-point communication system is*

$$P = \sum_{i=1}^{n} \mathbb{E}[\rho(X_i)],$$

where the expectations are taken over the marginal distributions of the X_i.

Definition 1.2. *The* average distortion *incurred by a memoryless point-to-point communication system is*

$$D = \sum_{i=1}^{k} \mathbb{E}[d(S_i, \hat{S}_i)],$$

where the expectation is taken over the joint distribution of S_i and $\hat{S}_i$.

Definition 1.3. *The* capacity-cost function *of a channel $P_{Y|X}$ with cost function $\rho(x)$ is*

$$C(P) = \max_{P_X : \mathbb{E}[\rho(X)] \leq P} I(X;Y).$$

Definition 1.4. *The* rate-distortion function *of a source P_S with distortion measure $d(s,\hat{s})$ is*

$$R(D) = \min_{P_{\hat{S}|S} : \mathbb{E}[d(S,\hat{S})] \leq D} I(S;\hat{S}).$$

1.2 Fundamental Limits of Performance

The goal of source-channel communication is to transmit a source at a low cost of transmission and with a reconstruction at the receiver that has little distortion from the original. The region of achievable cost and distortion pairs is thus fundamental in establishing the best possible performance of a given communication problem.

Definition 1.5. *For a given source P_S producing a symbol every τ_s seconds, a channel $P_{Y|X}$ accepting an input every τ_c seconds, a cost measure $\rho(x)$ and a distortion measure $d(s,\hat{s})$, the* achievable cost/distortion region *is the set of all pairs (D,P) for which there exists a sequence of codes that approach the distortion D and the cost P in the limit.*

The most fundamental bound on the achievable cost/distortion region is given by the following result [33, Theorem 21].

Theorem 1.1. *In any memoryless point-to-point source-channel communication system, the average cost P and the average distortion D are related by*

$$kR(D) \leq nC(P). \tag{1.1}$$

Proof. See Appendix 1.A. □

Because $R(D)$ is a decreasing function of D and $C(P)$ is an increasing function of P, Theorem 1.1 can alternatively be written as $D \geq R^{-1}((n/k)C(P))$ or $P \geq C^{-1}((k/n)R(D))$. It thus specifies the smallest distortion achievable for a given cost constraint or the smallest cost required to achieve a given distortion, respectively, and so provides an *outer bound* to the achievable cost/distortion region. Regardless of how powerful an encoder and decoder are, the incurred

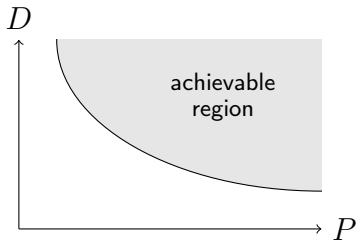


Figure 1.2: Schematic display of the achievable cost and distortion region that follows from Theorem 1.1.

cost and distortion always lie in the region specified by (1.1); this is illustrated in Figure 1.2. Note also that (1.1) only depends on the ratio k/n, so that the same bound applies to a code that encodes $1000k$ source symbols into $1000n$ channel inputs.

This theorem is often referred to as the converse part of the separation theorem, because it establishes that a (D, P) pair which cannot be achieved (or approached) using separately performed source and channel coding cannot be achieved at all. However, the theorem should not only be seen in relation to separation-based coding. In fact, while the forward part of the separation theorem (given for reference in Appendix 1.B) is only valid if one allows for codes of unrestricted delay (and complexity), Theorem 1.1 applies to *all* codes and is thus in a way more general than the forward part. In particular, it also applies to codes with limited delay, which are a central subject of this thesis.

1.3 Optimality Conditions

For the purpose of this chapter, an optimal communication system is defined as follows.

Definition 1.6. *An* optimal communication system *is a communication system whose average cost and distortion satisfy* (1.1) *with equality.*[1]

By going step by step through the inequalities in the proof of Theorem 1.1, the following result can be established.

Theorem 1.2. *A point-to-point memoryless communication system is optimal according to Definition 1.6 if and only if the following conditions are all satisfied.*

[1]This is not the strictest form of optimality, and it precludes the existence of optimal communication systems in some situations, for example when $\max_D R(D) < \min_P C(P)$. See Gastpar [12] for a more general definition of optimality.

1. *For each $i = 1, \ldots, k$, the joint distribution $p(s_i, \hat{s}_i)$ achieves the rate-distortion function of the source such that*

$$R\left(\frac{1}{k}\sum_{i=1}^{k}\mathbb{E}[d(S_i,\hat{S}_i)]\right) = \frac{1}{k}\sum_{i=1}^{k}R\left(\mathbb{E}[d(S_i,\hat{S}_i)]\right).$$

2. *The* reverse test channel $p(s^k|\hat{s}^k)$ *factors as* $\prod_{i=1}^{k} p(s_i|\hat{s}_i)$.

3. *The encoder is information lossless in the sense that it satisfies*

$$I(S^k;Y^n) = I(X^k;Y^n).$$

4. *The estimate sequence $\hat{S}^k$ is a sufficient statistic for S^k given Y^n. (Equivalently, we can say that the decoder must be memoryless in the sense that $I(X^n;Y^n) = I(X^n;\hat{S}^k)$.)*

5. *The channel output sequence Y^n consists of independent random variables.*

6. *The marginal distributions $p(x_i)$ all achieve the capacity of the channel such that*

$$\frac{1}{n}\sum_{i=1}^{n}C(\mathbb{E}[\rho(X_i)]) = C\left(\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[\rho(X_i)]\right).$$

According to the above definition of optimality, a communication system that uses codes based on the separation principle can only approach, but not achieve optimality, in the sense that for any $\epsilon > 0$ there exists a separation based code for which $kR(D) = nC(P) - \epsilon$, but in general not for $\epsilon = 0$.

Can only codes based on the separation principle achieve (or approach) any point of the achievable cost/distortion region? Not at all. As the next section shows, one can construct infinitely many examples of very simple joint source-channel codes that incur minimal delay yet whose average cost and distortion lie on the boundary of the achievable region characterized by (1.1) and are thus optimal.

1.3.1 Optimality through Measure Matching

In a bandwidth-matched system, where the number n of channel inputs per second equals the number k of source symbols per second, a communication system is optimal according to Definition 1.6 if $R(D) = C(P)$. Consider now a *single-letter code* (f, g) that maps each source symbol S into a single channel input symbol X and each channel output symbol Y into an estimate $\hat{S}$. For this case, Theorem 1.2 simplifies to the following statement.

Theorem 1.3 (Single-letter optimality [14])**.** *A bandwidth-matched point-to-point communication system is optimal according to Definition 1.6 if and only if the following conditions are all satisfied.*

1. *The joint distribution $p(s,\hat{s})$ achieves the rate-distortion function of the source at average distortion $D = \mathbb{E}[d(S,\hat{S})]$, i.e., $I(S;\hat{S}) = R(D)$.*
2. *The code is information lossless in the sense that $I(S;\hat{S}) = I(X;Y)$.*
3. *The channel input distribution $p(x)$ achieves the capacity of the channel at average cost P, i.e., $I(X;Y) = C(P)$.*

Proof. Apply Theorem 1.2 with $k = n = 1$. □

One way to verify condition 1 of Theorem 1.3 is to find the conditional distribution $p_{\hat{S}|S}$ that achieves the rate-distortion function of the source given the distortion measure $d(s,\hat{s})$ and then to compare it to the actual distribution. Similarly, to verify condition 3 one can find the capacity achieving distribution of the channel $p_{Y|X}$ given the cost measure $\rho(x)$ and then check whether it matches the actual input distribution.

Alternatively, the following results by Gastpar et al. [14, Lemmas 2.2 and 2.3] allow to test conditions 1 and 3 of Theorem 1.3 without solving the capacity problem and the rate-distortion problem.

Lemma 1.4. *For a fixed discrete source distribution p_S, a discrete channel conditional distribution $p_{Y|X}$, and a single-letter code (f,g):*

1. *If $0 < I(S;\hat{S})$, condition 1 of Theorem 1.3 is satisfied if and only if the distortion measure satisfies*

$$d(s,\hat{s}) = -c_1 \log_2 \frac{p(\hat{s}|s)}{p(\hat{s})} + d_0(s), \tag{1.2}$$

   where $c_1 > 0$ and $d_0(\cdot)$ is an arbitrary function.

2. *If $I(S;\hat{S}) = 0$, condition 1 of Theorem 1.3 is satisfied for any distortion measure $d(s,\hat{s})$.*

Lemma 1.5. *For a fixed discrete source distribution p_S, a single-letter encoder f, and a discrete channel conditional distribution $p_{Y|X}$ with unconstrained capacity $C_0 \stackrel{def}{=} \max_{p_X} I(X;Y)$:*

1. *If $I(X;Y) < C_0$, condition 3 of Theorem 1.3 is satisfied if and only if the input cost measure satisfies*

$$\rho(x) \begin{cases} = c_2 D(p_{Y|X}(\cdot|x)\|p_Y(\cdot)) + \beta, & \text{if } p(x) > 0, \\ \geq c_2 D(p_{Y|X}(\cdot|x)\|p_Y(\cdot)) + \beta, & \text{otherwise,} \end{cases} \tag{1.3}$$

   where $c_2 > 0$ and β are constants, and $D(\cdot\|\cdot)$ denotes the Kullback-Leibler divergence between two distributions.

2. *If $I(X;Y) = C_0$, condition 3 of Theorem 1.3 is satisfied for any cost measure $\rho(x)$.*

(For sources and channels with continuous alphabets, the conditions are sufficient but not necessary.)

As a consequence of these two lemmas, one can find for any source, channel, and (information lossless) single-letter code a suitable cost measure and a suitable distortion measure such that the resulting system satisfies $R(D) = C(P)$. There is therefore an infinity of optimal single-letter communication systems. Two well known examples of such optimal single-letter systems are the transmission of a binary symmetric source across a binary symmetric channel under Hamming distortion, and the transmission of a Gaussian source across an AWGN channel under an average power constraint and squared error distortion measure (the latter will be described in detail in Chapter 3).

1.3.2 Systems with Bandwidth Mismatch

In a communication system with $k \neq n$, Lemmas 1.4 and 1.5 still apply: for each i the conditional marginal distribution of $\hat{S}_i$ given S_i must relate to the distortion measure according to Lemma 1.4, and the marginal distribution of X_i must relate to the cost function according to Lemma 1.5. But this is no longer enough, as Theorem 1.2 shows. Thus, for a given source, channel, and code, one cannot necessarily find suitable cost and distortion measures to make the system optimal, as opposed to the case when $k = n$. In particular, even in the canonical case of transmitting a Gaussian source across a Gaussian channel under squared error distortion and an input power constraint, for which uncoded transmission is optimal when $k = n$, no simple transmission scheme exists when $k \neq n$.

1.4 Two Facets of the Problem

The contributions of this thesis address two facets of the source-channel communication problem. In Chapter 2 the question is: when does a communication system maximize the *ratio* of fidelity per cost, rather than maximize the fidelity (i.e., minimize the distortion) for a fixed cost or minimize the cost for a fixed fidelity. It turns out that a simple refinement of Lemmas 1.4 and 1.5 yields new matching conditions for this case. In particular, these new matching conditions allow to check when uncoded transmission (with minimal delay) achieves the maximal fidelity per cost ratio. Moreover, the problem of maximizing the fidelity per cost can be broken down into separate problems that concern only the source and the channel, respectively, thus providing a separation theorem.

Chapters 3 and 4 focus solely on the Gaussian channel. They consider the problem of how to transmit an analog-valued source across *multiple uses* of a Gaussian channel at minimal delay (and complexity). Chapter 3 draws insight from a well known feedback communication scheme to obtain intuition about how to take advantage of uncoded transmission in the absence of feedback. These reflections lead to a tradeoff between coded and uncoded communication,

mirroring a well studied tradeoff between the length of the signal curve and the distance between the curve's folds that results from a geometric analysis of the problem.

The results of Chapter 3 give an alternative *raison d'être* to a well known hybrid communication strategy combining quantization and uncoded transmission. Chapter 4 analyzes this strategy in detail and provides an exact characterization of its performance in the large SNR regime, as well as several extensions to the transmission strategy.

1.A Proof of the Converse Part of Theorem 1.1

The proof results from the following chain of inequalities.

$$
\begin{aligned}
kR(D) &= kR\left(\frac{1}{k}\sum_{i=1}^{k}\mathbb{E}[d(S_i,\hat{S}_i)]\right)\\
&\overset{(a)}{\leq} \sum_{i=1}^{k} R\left(\mathbb{E}[d(S_i,\hat{S}_i)]\right)\\
&\overset{(b)}{\leq} \sum_{i=1}^{k} I(S_i;\hat{S}_i)\\
&= \sum_{i=1}^{k}\left(H(S_i)-H(S_i|\hat{S}_i)\right)\\
&= H(S^k)-\sum_{i=1}^{k}H(S_i|\hat{S}_i)\\
&\overset{(c)}{\leq} H(S^k)-\sum_{i=1}^{k}H(S_i|S^{i-1}\hat{S}^k)\\
&= I(S^k;\hat{S}^k)\\
&\overset{(d)}{\leq} I(S^k;Y^n)\\
&\overset{(e)}{\leq} I(X^n;Y^n)\\
&= \sum_{i=1}^{n}\left(H(Y_i|Y^{i-1})-H(Y_i|Y^{i-1}X^n)\right)\\
&\overset{(f)}{\leq} \sum_{i=1}^{n}\left(H(Y_i)-H(Y_i|X_i)\right)\\
&= \sum_{i=1}^{n} I(X_i;Y_i)
\end{aligned}
$$

$$\begin{aligned}
&\overset{(g)}{\leq} \sum_{i=1}^{n} C(\mathbb{E}\rho(X_i)) \\
&\overset{(h)}{\leq} nC\left(\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\rho(X_i)\right) \\
&= nC(P).
\end{aligned}$$

The inequalities are justified as follows. (a) follows from the convexity$_\cup$ of $R(D)$ and (b) from its definition. (c) is because conditioning can only decrease the entropy. (d) and (e) follow from the data processing inequality. (f) is because conditioning can only decrease entropy and because the channel is memoryless. Finally, (g) applies by definition and (h) is due to the concavity$_\cap$ of $C(P)$. □

1.B Separation Theorem, Forward Part

Theorem 1.6. *Consider a memoryless source that has rate-distortion function $R(D)$ and produces a source symbol every τ_s seconds, and a memoryless channel with capacity-cost function $C(P)$ that accepts an input symbol for transmission every τ_c seconds. Then for any (D, P) pair satisfying*

$$R(D)/\tau_s \leq C(P)/\tau_c - \epsilon, \tag{1.4}$$

where $\epsilon > 0$, there exists a source code and a channel code that, when combined to transmit the source across the channel, result in an average distortion of at most D and an average cost of at most P.

Proof. Assume $R(D)/\tau_s \leq C(P)/\tau_c - \epsilon$. According to the source coding theorem there exists, for an arbitrary $\epsilon' > 0$, a source code that achieves distortion at most D and produces $R(D) + \epsilon'$ bits per source symbol, or $(R(D) + \epsilon')/\tau_s$ bits per second. According to the channel coding theorem, for any $\epsilon' > 0$ there exists a channel code that uses average input cost at most P and can reliably transmit $C(P) - \epsilon'$ bits per channel use or $(C(P) - \epsilon')/\tau_c$ bits per second.

The output of the source code can be mapped to the input of the channel code provided that $(R(D) + \epsilon')/\tau_s \leq (C(P) - \epsilon')/\tau_c$, or equivalently

$$R(D)/\tau_s \leq C(P)/\tau_c - \epsilon'\left(\frac{1}{\tau_s} + \frac{1}{\tau_c}\right).$$

By choosing ϵ' small enough, it follows from the assumption on D and P that this inequality holds. □

2 On Fidelity per Unit Cost

The previous chapter mentioned the principle of *measure matching* to characterize an optimal communication system: in an optimal communication system (in the sense of Definition 1.6) the cost measure and the distortion measure are matched in a specific way to the statistics of the system. The present chapter refines the concept of an optimal communication system and looks at the conditions necessary to maximize the *ratio of fidelity per cost*, where fidelity is a quantity playing the role of the distortion but which should be maximized rather than minimized.

Three standard communication problems are considered, namely source coding with a fidelity criterion, channel coding with a channel input constraint, and the combined problem of reproducing a source across a noisy channel. The focus here is on discrete memoryless sources and channels, but the essence of the results extends to continous alphabets.[1]

The chapter begins by addressing the channel coding problem in Section 2.1. It first considers the capacity per unit cost, a concept that is traceable in various forms as far back as Shannon [33] Reza [27] (see also references in [42]) and was more recently studied by Gallager ([11], [2, Ch. 14]) and developed by Verdú in his influential paper [42] (see also [44]).

The capacity per unit cost for a channel with transition probabilities $P_{Y|X}$ and input cost measure $\rho(x)$ is defined as

$$\hat{C} = \sup_{P_X} \frac{I(X;Y)}{\mathbb{E}[\rho(X)]},$$

where the supremum is taken over all possible input distributions P_X. Usually $P_{Y|X}$ and $\rho(x)$ are given, and to compute the capacity per unit cost one maximizes over the space of input distributions. For the case when a channel

[1] Up to editorial changes, this chapter is identical to [21].

input symbol of zero cost exists, however, Verdú showed that the capacity per unit cost can be obtained by a simpler maximization over the channel input alphabet.

We look at the problem from a different angle. We show that for fixed P_X and $P_{Y|X}$ there is a simple expression for the cost measure $\rho(x)$ for which P_X achieves the capacity per unit cost, obtained by refining Lemma 1.5. This criterion is applicable whether or not a zero-cost symbol exists.

We illustrate our result with a detailed example of a Gaussian channel with Gaussian input. First we find that the cost measure for which the system operates at capacity per unit cost is of the form $\rho(x) = x^2 + k$ for some positive constant k (to be specified). We show that a discrete-time channel with this cost measure relates to a continuous-time channel for which power can be traded against bandwidth. Furthermore, the tradeoff that maximizes the rate (in bits/second) of the continuous-time channel corresponds to the operating point at which the discrete-time channel achieves the capacity per unit cost.

For the rate-distortion problem a similar result exists, which is explored in Section 2.2. For reasons that will become apparent, it is more convenient to study this result in terms of a fidelity measure, defined as the negative of the distortion measure. Following this line of thought we define the *fidelity per unit rate* of a source: it is to source coding what the capacity per unit cost is to channel coding. Refining Lemma 1.4, we give a condition for a test channel[2] to achieve the fidelity per unit rate of the source.

Section 2.3 considers the end-to-end problem of reproducing a source across a noisy channel. Drawing on the results of Sections 2.1 and 2.2, it is shown that the capacity per unit cost and the fidelity per unit rate can be combined to give the *fidelity per unit cost*, i.e., the maximum ratio of fidelity per cost at which a source-channel coding system can operate.

The set of communication systems that operate at fidelity per unit cost is a subset of the set of communication systems that are optimal in the sense of Definition 1.6 of Chapter 1 (page 8). In the last result of the present chapter we give necessary and sufficient conditions under which a given communication system operates at fidelity per unit cost. This can be seen in analogy to Theorem 1.2, which gave necessary and sufficient conditions for a communication system to be optimal according to Definition 1.6.

[2]In rate-distortion theory, a test channel is a conditional distribution of the reconstruction given a source symbol.

2.1 Achieving Capacity per Unit Cost

Let $P_{Y|X}$ be the transition distribution of a discrete memoryless channel and let P_X and Q_X be two arbitrary but fixed input distributions. Let P_Y and Q_Y be the corresponding output distributions, i.e.,

$$P_Y(y) = \sum_x P_X(x) P_{Y|X}(y|x) \quad \text{and}$$
$$Q_Y(y) = \sum_x Q_X(x) P_{Y|X}(y|x).$$

Define $\rho_0(x)$ through P_X as

$$\rho_0(x) = D(P_{Y|X}(\cdot|x)||P_Y(\cdot)). \tag{2.1}$$

Let $\mathbb{E}_P$ and $\mathbb{E}_Q$ denote expectations with respect to P_X and Q_X, respectively, and let $I_P(X;Y)$ and $I_Q(X;Y)$ denote the mutual informations corresponding to these input distributions.

Proposition 2.1. *We have*

$$\frac{I_Q(X;Y)}{\mathbb{E}_Q[\rho_0(X)]} \leq \frac{I_P(X;Y)}{\mathbb{E}_P[\rho_0(X)]} = 1, \tag{2.2}$$

with equality if and only if $Q_Y = P_Y$.

Proposition 2.1 implies that if the channel input cost function is as in (2.1) (up to scaling), then P_X achieves the capacity per unit cost.

Proof. First note that $\mathbb{E}_P[\rho_0(X)] = I_P(X;Y)$, i.e., $I_P(X;Y)/\mathbb{E}_P[\rho_0(X)] = 1$. On the other hand, if the input distribution is Q_X then

$$\begin{aligned}
&\mathbb{E}_Q[\rho_0(X)] - I_Q(X;Y) \\
&= \sum_{x,y} Q_X(x) P_{Y|X}(y|x) \left[\log \frac{P_{Y|X}(y|x)}{P_Y(y)} - \log \frac{P_{Y|X}(y|x)}{Q_Y(y)} \right] \\
&= D(Q_Y \| P_Y) \geq 0.
\end{aligned}$$

It follows that

$$\frac{I_Q(X;Y)}{\mathbb{E}_Q[\rho_0(X)]} \leq 1$$

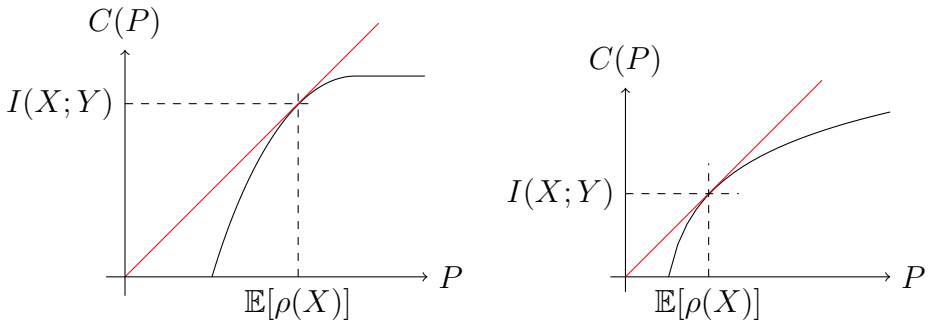
with equality if and only if $P_Y = Q_Y$. □

Remark 2.1. The proposition also holds for channels with continuous alphabets. This follows directly if the sums in the proof are replaced by integrals.

Example 2.1 (BSC)**.** *Consider a binary symmetric channel with crossover probability* $\epsilon = 0.1$. *Let the input distribution be such that* $P_X(0) = 0.7$ *and* $P_X(1) = 0.3$. *Evaluating* $\rho_0(x)$ *for this situation yields*

$$\rho_0(0) \approx 0.23 \quad \textit{and} \quad \rho_0(1) \approx 0.99.$$

The corresponding capacity curve is plotted in Figure 2.1a.



(a) BSC (b) Gaussian

Figure 2.1: Capacity curves for Examples 2.1 (left) and 2.2 (right). The optimal cost function is such that a tangent through the origin touches the the curve in the operating point that corresponds to the given input distribution P_X.

Example 2.2 (Gaussian)**.** *Consider the AWGN channel $Y = X + Z$, where $X \sim \mathcal{N}(0, \sigma_X^2)$ and $Z \sim \mathcal{N}(0, 1)$. Evaluating $\rho_0(x)$, we obtain*

$$\rho_0(x) = D(P_{Y|X}(\cdot|x)||P_Y(\cdot)) = ax^2 + b,$$

where

$$a = \frac{1}{2\ln 2(1+\sigma_X^2)}$$

and

$$b = \frac{1}{2}\left(\log_2(1+\sigma_X^2) - \frac{1}{\ln 2} + \frac{1}{\ln 2(1+\sigma_X^2)}\right).$$

The cost constraint $\mathbb{E}[\rho_0(x)] \leq P$ can be written as $\mathbb{E}[X^2] \leq \frac{P-b}{a}$, so the corresponding capacity-cost function is

$$C(P) = \frac{1}{2}\log_2(1 + \frac{P-b}{a}), \quad P \geq b.$$

It is plotted in Figure 2.1b.

When does a cost measure of the form $\rho(x) = x^2 + b$ make sense? The following example gives an answer to this question.

Example 2.3. *Consider the problem of designing a communication system that maximizes the communication rate across a continuous-time Gaussian channel. We are free to trade bandwidth for power, provided that $P + kB \leq A$ where B is the (two-sided) bandwidth, P is the power, and k and A are positive constants. Using the sampling theorem at intervals of length T, $BT = 1$, we translate the above problem into its discrete-time equivalent. To do so we first rewrite the constraint as $\frac{1}{T}\int_0^T \mathbb{E}[X^2(t)]dt \leq A - kB$ and use the fact that the discrete-time equivalent of the integral is $\mathbb{E}[X^2]$. Hence the discrete-time constraint becomes $\frac{1}{T}\mathbb{E}[X^2] \leq A - \frac{k}{T}$ or, equivalently, $\mathbb{E}[X^2 + k] \leq AT$.*

With this constraint, the maximum rate at which we can transmit reliably (in bits/second) is

$$\frac{C(AT)}{T} = A\frac{C(AT)}{AT} \leq A\hat{C},$$

where $\hat{C}$ is the capacity per unit cost of the discrete-time channel. Notice that we can choose to operate at capacity per unit cost by choosing T such that $\frac{C(AT)}{AT}$ achieves the maximum. This implies the optimal power–bandwidth tradeoff.

Example 2.4 (Exponential)**.** *Consider an additive exponential noise channel with nonnegative input alphabet $Y = X + N$, where N is exponential with mean b. Let the input distribution P_X be equal to 0 with probability $\frac{b}{a+b}$ and exponentially distributed with mean $a + b$ conditioned on it being positive, i.e.,*

$$\Pr[X = 0] = \frac{b}{a+b} \tag{2.3}$$

and

$$\Pr[X > x | X > 0] = e^{-x/(a+b)} \; . \tag{2.4}$$

Then the input X has mean a, and the output Y is exponential with mean $a+b$, as shown by Verdú in [43].

Drawing further on results from [43] we have

$$\begin{aligned} \rho_0(x) &= D(P_{Y|X}(\cdot|x)||P_Y(\cdot)) \\ &= \log_2(1 + \frac{a}{b}) + \frac{x-a}{a+b}\log_2 e \end{aligned}$$

and

$$\begin{aligned} I(X;Y) &= \mathbb{E}[D(P_{Y|X}(\cdot|X)||P_{Y|X}(\cdot))] \\ &= \log_2(1 + \frac{a}{b}) \; . \end{aligned} \tag{2.5}$$

Solving the cost constraint $\mathbb{E}[\rho_0(X)] \leq P$ for $\mathbb{E}[X]$ then yields

$$\mathbb{E}[X] \leq \frac{a+b}{\log_2 e}(P - \log_2(1 + \frac{a}{b})) + a \; . \tag{2.6}$$

For a constraint on the mean, an input distribution in the form of (2.3) *and* (2.4) *(with corresponding mean a) achieves capacity on the exponential channel. We can thus replace a in* (2.5) *by the right hand side of* (2.6) *to obtain the cost-constrained capacity*

$$C(P) = \log_2\left(1 + \frac{\frac{a+b}{\log_2 e}(P - \log_2(1 + a/b)) + a}{b}\right). \tag{2.7}$$

The corresponding capacity curve is displayed in Figure 2.2.

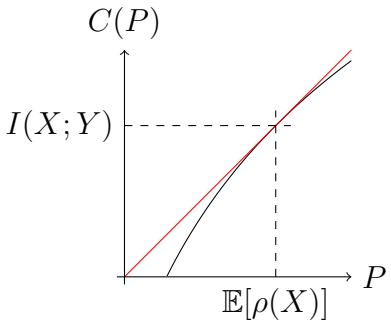


Figure 2.2: Capacity curve for the exponential channel of Example 2.4.

In the next proposition we show that under a minor technical condition, the cost function for which an input distribution achieves capacity per unit cost is unique (up to a scaling factor).

Proposition 2.2. *If P_X achieves the capacity per unit cost of the channel $P_{Y|X}$ with cost function $\rho(x)$ and if the derivative of the corresponding capacity-cost function $C(P)$ exists at $P = \mathbb{E}_P[\rho(X)]$, then $\rho(x) = c\rho_0(x)$ for some $c > 0$.*

Proof. Assume that P_X achieves the capacity per unit cost of the channel for some cost function $\rho(x)$. This implies that P_X also achieves the "regular" capacity of the channel at average cost $P^* = \mathbb{E}_P[\rho(X)]$, i.e.,

$$I_P(X;Y) = C(P^*).$$

By Lemma 1.5, a necessary condition for this is that $\rho(x) = c\rho_0(x) + \beta$ for some β. We therefore only have to show that if $\beta \neq 0$ then P_X does not achieve the capacity per unit cost.

Assume thus that $\rho(x) = c\rho_0(x) + \beta$, and let $C_0(P)$ and $C_\beta(P)$ be the capacity-cost functions corresponding to $\beta = 0$ and $\beta \neq 0$, respectively. $C_0(P)$ and $C_\beta(P)$ are related by

$$\begin{aligned} C_0(P) &= \max_{P_X:\mathbb{E}[c\rho_0(X)]\leq P} I(X;Y) \\ &= \max_{P_X:\mathbb{E}[c\rho_0(X)+\beta]\leq P+\beta} I(X;Y) \\ &= C_\beta(P+\beta). \end{aligned} \tag{2.8}$$

If $\beta = 0$ then P_X achieves the capacity per unit cost (cf. Proposition 2.1) and we have

$$\frac{d}{dP}\left.\frac{C_0(P)}{P}\right|_{P=P^*} = 0,$$

where $P^* = \mathbb{E}_P[\rho(X)]$. This is equivalent to

$$C_0'(P^*) = C_0(P^*)/P^*, \tag{2.9}$$

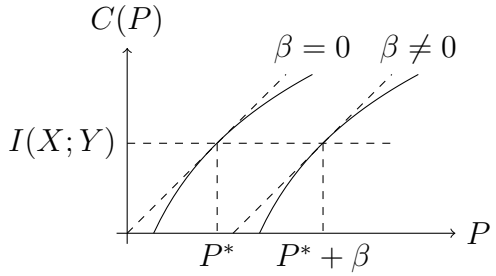


Figure 2.3: A geometric interpretation of the proof of Proposition 2.2. A nonzero β shifts the capacity-cost curve such that the tangent at the operating point no longer goes through the origin.

which says nothing else than the tangent of $C_0(P)$ at P^* goes through the origin, as shown in Figure 2.3.

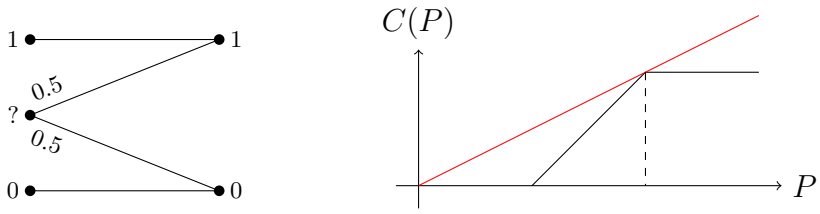
If $\beta \neq 0$ then the average cost under P_X is $P^* + \beta$. Then,

$$\begin{aligned} C_\beta{}'(P^* + \beta) &\overset{(a)}{=} C_0{}'(P^*) \\ &\overset{(b)}{=} \frac{C_0(P^*)}{P^*} \\ &\overset{(c)}{=} \frac{C_\beta(P^* + \beta)}{P^*} \\ &\neq \frac{C_\beta(P^* + \beta)}{P^* + \beta}, \end{aligned}$$

where (a) and (c) follow from (2.8) and (b) follows from (2.9). This means that the tangent of $C_\beta(P)$ at $P = P^* + \beta$ does not go through the origin, and so according to (2.9) we are not at capacity per unit cost. See Figure 2.3 for a geometric interpretation of this proof. □

The following counterexample shows that if the capacity function is not differentiable then the cost function for which the capacity per unit cost is achieved is not unique.

Example 2.5. *Consider the ternary-input channel shown in Figure 2.4a. The input alphabet is* $\{0, 1, ?\}$ *and the output alphabet is* $\{0, 1\}$*. Let the cost be given by* $\rho(?) = 1$ *and* $\rho(0) = \rho(1) = 2$*. If the average cost constraint* P *is between* 1 *and* 2*, the capacity achieving distribution is* $\Pr[X = 0] = \Pr[X = 1] = (P - 1)/2$ *and* $\Pr[X = ?] = 2 - P$*. Then* $H(Y) = 1$*,* $H(Y|X = 0) = H(Y|X = 1) = 0$ *and* $H(Y|X = ?) = 1$*, so the resulting mutual information is* $I(X;Y) = 1 - (2 - P) = P - 1$*. If* $P \geq 2$*, the capacity achieving distribution is* $\Pr[X = 0] = \Pr[X = 1] = 1/2$*, and the mutual information is* $I(X;Y) = 1$*. The corresponding capacity function, shown in Figure 2.4b, has a sharp edge at* $P = 2$*. It is thus clear that when it is moved horizontally (by changing* $\rho(x)$ *by*



(a) Ternary input channel.

(b) Capacity curve.

Figure 2.4: Ternary input channel and corresponding capacity function for Example 2.5. If a constant is added to the cost function, the capacity curve in (b) shifts horizontally but a tangent through the origin still touches the curve in the same point. Hence the cost function for which capacity per unit cost is achieved is not unique in this case.

a constant), the new operating point will still achieve the capacity per unit cost.

Propositions 2.1 and 2.2 are summarized in the following theorem.

Theorem 2.3. *Under the technical condition of Proposition 2.2, the channel input distribution P_X achieves the capacity per unit cost of the DMC $P_{Y|X}$ if and only if the cost function satisfies*

$$\rho(x) = cD(P_{Y|X}(\cdot|x)||P_Y(\cdot)), \quad c > 0. \tag{2.10}$$

For continuous alphabets, the condition is sufficient but not necessary.

This result shows that the extra requirement that the capacity per unit cost be achieved restricts the cost measures allowed by Lemma 1.5 to those with $\beta = 0$.

2.2 A Dual Result for the Source Coding Problem

Consider a discrete memoryless source (DMS) S with distribution P_S. Let $V(\hat{s}|s)$ and $W(\hat{s}|s)$ be two test channels for the given source. Let $P_{\hat{S}}$ and $Q_{\hat{S}}$ be the corresponding unconditional distributions of $\hat{S}$, i.e.,

$$P_{\hat{S}}(\hat{s}) = \sum_s P_S(s)V(\hat{s}|s) \quad \text{and}$$
$$Q_{\hat{S}}(\hat{s}) = \sum_s P_S(s)W(\hat{s}|s).$$

Let d_0 be defined through $V(\hat{s}|s)$ as follows:

$$d_0(s,\hat{s}) = -\log_2 \frac{V(\hat{s}|s)}{P_{\hat{S}}(\hat{s})} + \gamma(s), \tag{2.11}$$

where $\gamma(s)$ is an arbitrary function of s satisfying $\mathbb{E}[\gamma(S)] = 0$. Let $\mathbb{E}_V$ and $\mathbb{E}_W$ denote expectations over S and $\hat{S}$ with respect to V and W, respectively, and let $I_V(S;\hat{S})$ and $I_W(S;\hat{S})$ be the corresponding mutual informations.

Proposition 2.4.

$$\frac{\mathbb{E}_W[d_0(S,\hat{S})]}{I_W(S;\hat{S})} \geq \frac{\mathbb{E}_V[d_0(S,\hat{S})]}{I_V(S;\hat{S})} = -1, \tag{2.12}$$

with equality if and only if $V(\hat{s}|s) = W(\hat{s}|s)$ for all s and $\hat{s}$.

Proposition 2.4 implies that when the distortion measure is as in (2.11) then $V(\hat{s}|s)$ is the test channel that minimizes the ratio of distortion per rate.

Proof. First note that $\mathbb{E}_V[d_0(S,\hat{S})] = -I_V(S;\hat{S})$, i.e., $\mathbb{E}_V[d_0(S,\hat{S})]/I_V(S;\hat{S}) = -1$. Next,

$$\begin{aligned}
&I_W(S;\hat{S}) + \mathbb{E}_W[d_0(S,\hat{S})] \\
&= \sum_{s,\hat{s}} P_S(s)W(\hat{s}|s)\left[\log_2 \frac{W(\hat{s}|s)}{Q_{\hat{S}}(\hat{s})} - \log_2 \frac{V(\hat{s}|s)}{P_{\hat{S}}(\hat{s})}\right] \\
&= \sum_{s,\hat{s}} P_S(s)W(\hat{s}|s)\left[\log_2 \frac{W(\hat{s}|s)}{V(\hat{s}|s)} - \log_2 \frac{Q_{\hat{S}}(\hat{s})}{P_{\hat{S}}(\hat{s})}\right] \\
&= \sum_{s} P_S(s) D(W(\cdot|s)\|V(\cdot|s)) - D(Q_{\hat{S}}\|P_{\hat{S}}) \\
&\geq 0,
\end{aligned}$$

where the inequality follows from the convexity of the Kullback-Leibler divergence (see e.g. [6, Thm 2.7.2]); it becomes an equality iff $V(\hat{s}|s) = W(\hat{s}|s)$. Rearranging the inequality, we obtain

$$\frac{\mathbb{E}_W[d_0(S,\hat{S})]}{I_W(S;\hat{S})} \geq -1.$$

□

Remark 2.2. This proposition holds also for continuous alphabets; just replace the sums in the proof by integrals.

The distortion as defined in (2.11) may take positive and negative values. In fact, Proposition 2.4 shows that the smallest achievable distortion per rate is -1. The reader may find this awkward; indeed it is hard to imagine an application where one would define a distortion this way. The issue disappears

if we define a *fidelity measure* $\phi_0(s,\hat{s}) = -d_0(s,\hat{s})$ and ask for the largest possible $\mathbb{E}_W[\phi_0(S,\hat{S})]/I_W(S;\hat{S})$, which will of course be 1. Hence, for the rest of the paper we will work with the the notion of fidelity, defined as the negative of the distortion.[3]

For later reference we restate Proposition 2.4 in terms of fidelity:

Proposition 2.5. *If $\phi_0(s,\hat{s}) = -d_0(s,\hat{s})$, then*

$$\frac{\mathbb{E}_W[\phi_0(S,\hat{S})]}{I_W(S;\hat{S})} \leq \frac{\mathbb{E}_V[\phi_0(S,\hat{S})]}{I_V(S;\hat{S})} = 1, \tag{2.13}$$

with equality if and only if $V(\hat{s}|s) = W(\hat{s}|s)$ for all s and $\hat{s}$.

Hence, if the fidelity measure is $\phi_0(s,\hat{s})$ then $V(\hat{s}|s)$ is the test channel that maximizes the *fidelity per rate* of the source.

Analog to the rate-distortion function we can define

$$R(\Phi) = \min_{p(\hat{s}|s):\mathbb{E}[\phi(S,\hat{S})]\geq\Phi} I(S;\hat{S})$$

as the *rate-fidelity function* of the source at average fidelity Φ. The rate-fidelity function has the same operational meaning as the rate-distortion function: for each rate larger than $R(\Phi)$ there exists a source code that achieves fidelity Φ, and conversely no such code exists with a rate smaller than $R(\Phi)$.

Example 2.6 (Gaussian)**.** *Consider a memoryless Gaussian source S with zero mean and unit variance, and a test channel $V(\hat{s}|s)$ such that $\hat{S}$ given $S = s$ is distributed as*[4]

$$\mathcal{N}\left(\frac{\alpha}{\alpha+1}s, \frac{\alpha}{(\alpha+1)^2}\right)$$

for some $\alpha > 0$. Then $\hat{S}$ is distributed as $\mathcal{N}(0,\alpha/(\alpha+1))$, and $\phi_0(s,\hat{s})$ evaluates to

$$\log_2 \frac{V(\hat{s}|s)}{P_{\hat{S}}(\hat{s})} = c\left(k_1 + k_2 s^2 - (s-\hat{s})^2\right)$$

for some postive constants c, k_1, and k_2.

The rate-fidelity function corresponding to $\phi_0(s,\hat{s})$ is

$$R(\Phi) = \frac{1}{2}\log_2 \frac{\sigma_S^2}{k_1 + k_2\sigma_S^2 - \Phi/c}.$$

The plot of Φ vs. $R(\Phi)$ in Figure 2.5 bears a strong resemblance to Figure 2.1b: a tangent of slope 1 touches the curve in the operating point corresponding to P_S and $V(\hat{s}|s)$.

[3] Of course mathematically the two notions are quivalent.

[4] We use $\mathcal{N}(\mu,\sigma^2)$ to denote a Gaussian distribution of mean μ and variance σ^2.

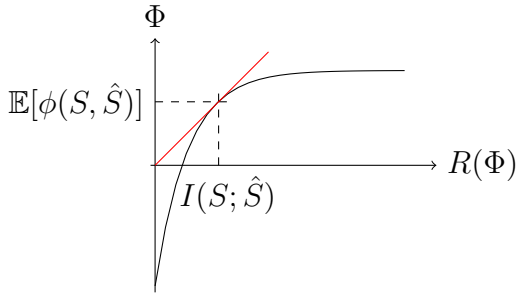


Figure 2.5: Average fidelity Φ vs. minimum rate $R(\Phi)$ for Example 2.6. The similarity to Figure 2.1b illustrates that the fidelity per rate is to a source what the capacity per cost is to a channel.

Does the fidelity obtained in the above example make sense? The multiplicative constant c is unavoidable: it accounts for the fact that one is free to choose the units. The term $k_2 s^2 - (s-\hat{s})^2$ indicates that the tolerance for the error $(s-\hat{s})^2$ is relative to s^2. In other words, small errors when s^2 is small are considered at the same level of satisfaction as large errors when s^2 is large, provided that $k_2 s^2 - (s-\hat{s})^2$ is constant. Finally, the constant k_1 accounts for the fact that the degree of satisfaction at the same value of $k_2 s^2 - (s-\hat{s})^2$ may vary from one application to another: a radiologist looking at an x-ray will have a smaller error tolerance than a person listening to music in a noisy environment.

The next proposition says that the fidelity measure $\phi(s,\hat{s})$ for which a given test channel maximizes the fidelity per rate is unique up to addition of a function of s with zero mean.

Proposition 2.6. *Let P_S be a discrete memoryless source with fidelity measure $\phi(s,\hat{s})$ and rate-fidelity function $R(\Phi)$. If the test channel $V(\hat{s}|s)$ maximizes the ratio*

$$\frac{\mathbb{E}[\phi(S,\hat{S})]}{I(S;\hat{S})}$$

among all test channels and if the derivative of $R(\Phi)$ exists at $\Phi = \mathbb{E}_V[\phi(S,\hat{S})]$, then $\phi(s,\hat{s}) = c\phi_0(s,\hat{s})$ for some $c > 0$.

The proof mimics that for Proposition 2.2. In particular, after taking expectation over S the function $\gamma(s)$ becomes a constant that plays the same role that β plays in the proof of Proposition 2.2.

Propositions 2.5 and 2.6 are summarized in the following theorem.

Theorem 2.7. *Under the technical condition of Proposition 2.6, the test channel $V(\hat{s}|s)$ maximizes the fidelity per rate of a discrete memoryless source if*

and only if the fidelity measure satisfies

$$\phi(s,\hat{s}) = c\log_2 \frac{V(\hat{s}|s)}{P_{\hat{S}}} + \gamma(s), \quad c > 0 \tag{2.14}$$

with $\mathbb{E}[\gamma(S)] = 0$. *For continous sources, the condition is sufficient but not necessary.*

This result shows that the extra requirement that the maximum fidelity per rate be achieved restricts the fidelity measures allowed by Lemma 1.4 to those with $\mathbb{E}[\gamma(S)] = 0$.

In analogy to the capacity per unit cost we define the *fidelity per unit rate* of a discrete memoryless source as

$$\hat{\Phi} = \sup_{V(\hat{s}|s)} \frac{\mathbb{E}_V[\phi(S,\hat{S})]}{I_V(S;\hat{S})} = \sup_{\Phi} \frac{\Phi}{R(\Phi)}.$$

In light of the previous observations, this quantity is to the source what the capacity per unit cost is to the channel.

Remark 2.3. To establish a connection with the perspective offered in Section IV of Verdú's paper [42], let Φ_0 be the maximum fidelity that can be achieved when representing the source by a fixed symbol, i.e., $\Phi_0 = \max_{\hat{s}} \mathbb{E}[\phi(S,\hat{s})]$. The following applies:

- If $\Phi_0 > 0$ then $\hat{\Phi} = \infty$. Indeed, the positive fidelity Φ_0 is achieved at zero rate.
- If $\Phi_0 < 0$ then $0 < \hat{\Phi} < \infty$.
- If $\Phi_0 = 0$ then $\hat{\Phi} = \lim_{\Phi \searrow 0} \Phi/R(\Phi)$.

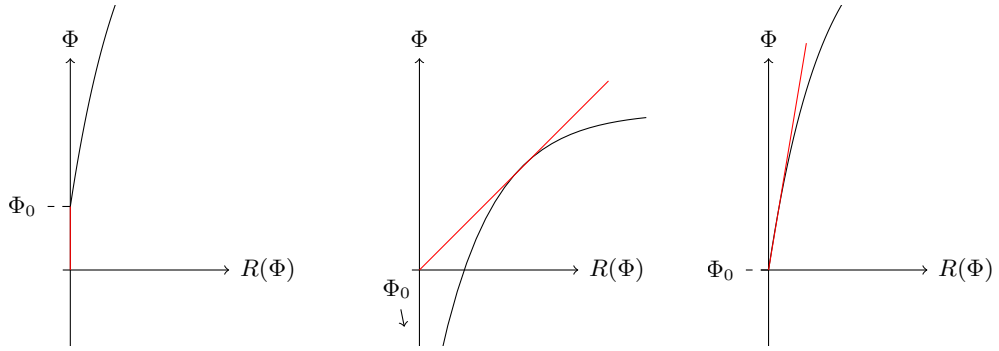
The three cases are illustrated in Figure 2.6.

This differs from Verdú's approach in that the latter – if stated in terms of fidelity – defines the counterpart to the capacity per unit cost to be $\lim_{\Phi \searrow \Phi_0}(\Phi - \Phi_0)/R(\Phi)$ regardless of the value of Φ_0.

2.3 Fidelity Per Unit Cost

The problems considered in the two previous sections are instances of a general class of problems where resources are traded against performance. In the channel case, we spend a cost (the resource) to achieve a high rate (the performance), and we want to maximize the performance per unit of resource. In the source case, we spend bits (the resource) to achieve a high fidelity (the performance), and once again we want to maximize the performance per unit of resource.

In this section we consider the end-to-end communication problem, where the resource is the channel input cost and the performance is the fidelity of



(a) $\Phi_0 > 0$ (b) $\Phi_0 < 0$ (c) $\Phi_0 = 0$

Figure 2.6: The three possible cases for Φ_0 as discussed in Remark 2.3.

the source reproduction. The goal is to maximize the *fidelity per cost*. The aim of this section is to find a tight bound on this ratio and to characterize the operating point that achieves the bound.

From the separation theorem, the average fidelity and cost of any communication system must satisfy

$$kR(\Phi) \leq nC(P), \tag{2.15}$$

where k and n are the number of source symbols and channel uses per time, respectively. Furthermore, the largest achievable Φ for a fixed P is the one for which the above inequality is satisfied with equality. Conversely, any fidelity/cost pair satisfying the inequality can be approached by a suitable communication system using separate source and channel coding.

Using the definitions of $\hat{\Phi}$ and $\hat{C}$,

$$\hat{\Phi} = \sup_{\Phi} \frac{\Phi}{R(\Phi)} \qquad\qquad \hat{C} = \sup_{P} \frac{C(P)}{P},$$

together with (2.15), we obtain the following tight bound on the fidelity per cost:

$$\frac{\Phi}{P} = \frac{\Phi}{R(\Phi)} \cdot \frac{R(\Phi)}{C(P)} \cdot \frac{C(P)}{P} \leq \hat{\Phi}\frac{n}{k}\hat{C}. \tag{2.16}$$

We call the right-hand side of (2.16) the *fidelity per unit cost* of the communication system.

A separation-based communication system achieves the fidelity per unit cost if and only if

1. the source encoder and decoder operate at $\hat{\Phi}$,
2. the channel encoder and decoder operate at $\hat{C}$, and
3. the number of bits produced by the source encoder for k source symbols equals the number of bits transmitted on the channel in n channel uses.

Figure 2.7: A point-to-point communication system consisting of a discrete memoryless source (DMS) with distribution P_S, a discrete memoryless channel (DMC) with transition probabilities $P_{Y|X}$, an encoder and a decoder.

Using the converse to the source-channel coding theorem as done in [12] along with the results of the previous two sections we can translate the conditions for equality in (2.16) into conditions that apply to any system that achieves the maximum fidelity per cost, regardless of whether the system relies on the separation principle, on joint source-channel block coding, or on uncoded communication. These general conditions are stated in the following theorem.

Theorem 2.8. *A discrete point-to-point communication system of the type shown in Figure 2.7 achieves the fidelity per unit cost if and only if all of the following conditions are satisfied.*

1. *The distribution of the reproduction sequence given the source sequence factors as*

$$P_{\hat{S}^k|S^k}(\hat{s}^k|s^k) = \prod_{i=1}^{k} V(\hat{s}_i|s_i),$$

   where $s^k = (s_1, \ldots, s_k)$ and where V is the test channel that achieves fidelity per unit rate, i.e., it relates to the fidelity function according to (2.14).

2. *The channel input is* iid *with distribution P_X that achieves capacity per unit cost, i.e., it relates to the cost function according to* (2.10).

3. *The encoder and decoder are information lossless in the sense that they satisfy $kI(S;\hat{S}) = nI(X;Y)$.*

This result is a refinement of Theorem 1.2, which gave necessary and sufficient conditions for the cost P and distortion D of a point-to-point communication system to satisfy $kR(D) = nC(P)$. According to (2.16), a communication system that operates at fidelity per unit cost also satisfies $kR(D) = nC(P)$, so it is also optimal in the sense of Definition 1.6.

2.4 Conclusions

This chapter provides a tight upper bound to the ratio of average fidelity per average cost of an arbitrary memoryless point-to-point communication system. Using the conditions in Theorems 2.3 and 2.7, one can verify whether

a given communication system does indeed operate at the maximum fidelity per cost ratio. In particular, the conditions can be used to check when uncoded communication performs optimally. This chapter also provides a separation theorem for fidelity per unit cost: it shows how the problem of maximizing the fidelity per cost corresponds to simultaneously achieving capacity per unit cost (on the channel) and achieving fidelity per unit rate (on the source), which can be approached arbitrarily close using appropriate sequences of source codes and channel codes.

Why would one want to maximize the *ratio* of fidelity per cost, rather than to obtain the greatest fidelity for a given cost constraint or incur the least cost for a required fidelity? It is not trivial to find an application for which this is indeed the case. There seems to be an important practical difference between fidelity on one hand, and rate and cost on the other hand. Rate and cost are naturally additive quantities. Suppose a channel is used n times per second at cost P/n per channel use. Then the total cost per second is P, regardless of n, and the total number of bits transmitted per second is the sum of the bits transmitted in each of the n channel uses. This gives important practical significance to capacity per unit cost: it determines the maximum number of bits per second one can transmit for a cost constraint *per second* if the number of channel uses is a free parameter (such as in wideband communication).

Fidelity is not obviously additive. For a given cost per second, one has the choice to either transmit a single source symbol at high fidelity, or several source symbols at a lower fidelity. Which one is better? In many situations the two cases may not be comparable. One possible application is oversampling with low-resolution quantization. For example, if a bandlimited analog source is oversampled at a sufficiently high rate, the quantization resolution can be as low as 1 bit per sample. Our results may yield a tradeoff between quantization resolution and sampling frequency; we have not investigated this issue further though. Nevertheless, it certainly presents an interesting opportunity for future research.

Delay-Limited Block Coding and Feedback

3

As seen in Chapter 1, the forward part of the separation theorem relies on the source coding and channel coding theorems, which assume unrestricted block length and delay. In a first part, this chapter looks at the consequences that arise when only block codes causing minimal delay can be used, with particular focus on the Gaussian channel.

If the number of channel uses per second is equal to the number of source symbols produced per second, then uncoded transmission (i.e., a simple scaling operation at the encoder and at the decoder) is optimal (in the sense of Definition 1.6) to transmit a Gaussian source across a Gaussian channel. If the number of channel uses per second is larger or smaller than the number of channel uses per second, no such simple communication strategy exists. If there is a perfect feedback link from the receiver to the encoder, however, there exists a simple scheme to transmit a single source symbol in n channel uses without any coding. The second part of the chapter discusses why such uncoded transmission works in the feedback case and provides insight on how to exploit the benefits of uncoded transmission even in the absence of feedback.

To be precise, we first define what we exactly mean by the *delay* of a code.

Definition 3.1. *The* delay *incurred by a point-to-point source-channel communication system is the largest time between the instant a source symbol is produced and the instant the decoder produces an estimate of that symbol.*

What is the delay of a block code that encodes k source symbols into n channel input symbols? Since the source produces a symbol every τ_s seconds it takes $k\tau_s$ seconds to gather k source symbols. Assuming that the encoding process takes place instantaneously, it takes an additional $n\tau_c$ seconds to transmit the encoded source sequence across the channel. Neglecting the transmission time of the channel, the receiver can thus decode the first source symbol after

$k\tau_s + n\tau_c$ seconds, or $2k\tau_s$ seconds[1]. Consequently, the smallest delay is caused by a code that encodes a *single* source symbol at a time. We call such a code a *minimal-delay code*.

Codes based on the separation theorem have the advantage that separate codes can be designed for the source and the channel. This way, the designer of the source code does not have to know a priori over what channel the source will be transmitted, and neither does the designer of the channel code have to know what kind of source will be transmitted across the channel. The cost paid for this flexibility is the large delay and complexity required by a separation based code.

In situations where external circumstances (such as real-time communication) put a constraint on the tolerable delay, the flexibility of separate source and channel coding must be sacrificed in favor of codes that are designed jointly for a particular source and a particular channel. As pointed out in Chapter 1, there exist such joint source-channel codes that perform as good as any separation-based code, yet only cause minimal delay.

From a theoretical point of view, the drawback of codes with a delay limit is that there exists in general no exact characterization of the achievable cost and distortion region for such codes. While the bound of Theorem 1.1 still applies, it only depends on the *ratio* k/n and therefore does not take into account delay constraints. There have certainly been attempts to refine this bound to apply to delay limited codes, most notably Ziv and Zakai's observation that tighter bounds can be obtained by replacing the logarithm in the mutual information by a different function, as long as this function satisfies certain constraints [50]. This can result in outer bounds that become tighter for stricter delay constraints. However, none of the bounds obtained (so far) using this method are provably tight for a given delay constraint, not even for the canonical case of a Gaussian source and channel, which is the subject of the next section.

Figure 3.1 summarizes the current state of knowledge. For codes without delay limits, the achievable cost/distortion region is completely characterized by the separation theorem. For delay limited codes, on the other hand, no general achievability result exists, but there are some refined outer bounds. The lower right corner of the figure is thus empty except for a few dots that represent special cases where minimal delay codes achieve the outer bound of Theorem 1.1. This is the region that we shall explore in this chapter and the next.

3.1 Fundamental Limits for the Gaussian Case

The Gaussian source and channel play a special role in information theory not only because of the significance of the Gaussian distribution due to the central limit theorem, but also because the Gaussian case allows for analytical

[1] because $k/n = \tau_c/\tau_s$

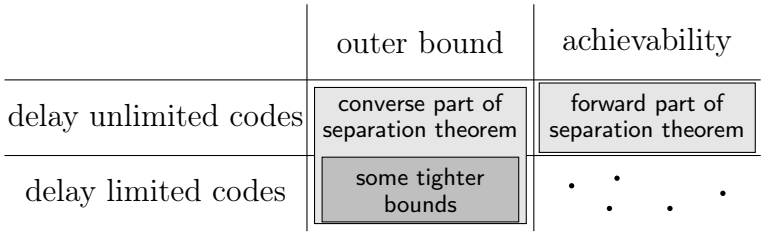


Figure 3.1: The current knowledge of the theoretical limits of source-channel coding. For codes without delay constraint, the outer bound and the region achievable using separation-based codes coincide and thus characterize exactly the achievable cost/distortion region. For delay limited codes, some bounds exist that improve on the converse separation theorem, but no general characterization of the achievable cost/distortion region exist. The few special cases where minimal-delay codes achieve the separation theorem bound are represented by the dots in the lower right corner.

solutions of many information theoretic quantities, most notably the rate-distortion function (under squared error distortion) and the capacity (under an average power constraint). On the search for the achievable cost and distortion region under a delay constraint, it is therefore plausible to start by looking at the Gaussian source and channel.

Definition 3.2 (Gaussian source and channel)**.** *A discrete-time memoryless* Gaussian source *of variance σ_S^2 is a source whose distribution is zero-mean Gaussian with variance σ_S^2.*

A discrete-time memoryless Gaussian channel *(also called discrete-time additive white Gaussian noise channel, AWGN) with noise variance ${\sigma_Z}^2$ is a channel whose transition distribution satisfies $p(y|x) \sim \mathcal{N}(x, {\sigma_Z}^2)$, i.e., given the input x, the output is Gaussian with mean x and variance ${\sigma_Z}^2$.*

The distortion measure considered in the sequel is the squared error distortion $d(s, \hat{s}) = (s - \hat{s})^2$, and the cost measure is the channel input power $\rho(x) = x^2$, such that $D = \frac{1}{k}\sum_{i=1}^{k} \mathbb{E}[(S_i - \hat{S}_i)^2]$ and $P = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}[X_i^2]$.

Plugging in the formulas for the rate-distortion function and the capacity-cost function for the Gaussian source and channel, the bound (1.1) becomes

$$\frac{\sigma_S^2}{D} \leq \left(1 + \frac{P}{{\sigma_Z}^2}\right)^{n/k} \tag{3.1}$$

or equivalently

$$\text{SDR} \leq (1 + \text{SNR})^{n/k}, \tag{3.2}$$

where we have defined the *signal-to-distortion ratio* $\text{SDR} = \sigma_S^2/D$ and the *signal-to-noise ratio* $\text{SNR} = P/{\sigma_Z}^2$. For arbitrary continuous-valued sources, the SDR can be bounded as

$$\text{SDR} \leq 2^{2D(P_S \| \phi_{\sigma_S^2})}(1 + \text{SNR})^{n/k}, \tag{3.3}$$

where $D(\cdot\|\cdot)$ is the relative entropy or Kullback-Leibler divergence between two distributions (see e.g. [6]) and $\phi_{\sigma_S^2}$ is the pdf of a centered Gaussian distribution of variance σ_S^2. This bound follows directly from Shannon's lower bound on the rate distortion function for arbitrary sources under a difference distortion measure[2] [35]. (If P_S is a Gaussian distribution, the divergence becomes zero and (3.3) simplifies to (3.1).)

In the asymptotic case where $\text{SNR} \to \infty$, the SDR for an arbitrary continuous-valued source when k source symbols are mapped to n channel inputs therefore behaves at best as $\text{SNR}^{n/k}$, or expressed formally,

$$\text{SDR} \in O(\text{SNR}^{n/k}),$$

where the "Big-O" asymptotic notation is defined in Appendix 3.A.

3.2 When Uncoded Transmission is Optimal

It is a well known fact that when $\tau_s = \tau_c$ (i.e., the number of source symbols produced per second is equal to that of channel inputs accepted per second), plugging a Gaussian source directly into a Gaussian channel results in an optimal communication system, as the following example, originally due to Goblick [16], makes clear.

Example 3.1. *Let the source be zero-mean Gaussian with variance σ_S^2 and let the channel be AWGN with noise variance ${\sigma_Z}^2$. Consider the encoder given by $X = f(S) = \sqrt{P/\sigma_S^2}S$ and the decoder given by $\hat{S} = g(Y) = \sqrt{P}\sigma_S^2 Y/(P+{\sigma_Z}^2)$. Using $Y = X + Z$ it is quickly verified that*

$$D = \mathbb{E}[(S - \hat{S})^2] = \sigma_S^2/(1 + P/{\sigma_Z}^2),$$

which is indeed the optimal distortion according to (3.1).

The example shows that when $\tau_s = \tau_c$, a joint source-channel code with the smallest possible delay (a single source symbol is encoded at a time) leads to the same region of achievable (D, P) pairs as when the delay is unrestricted.

It is instructive to look at how this example satisfies the conditions of Theorem 1.2. First, since $k = n = 1$, conditions 2 and 5 of the theorem are trivially satisfied. Because the encoder and the decoder are invertible, conditions 3 and 4 are satisfied as well. More interestingly, condition 6 is satisfied because the distribution of the source is in fact the capacity achieving distribution of the channel (up to scaling), so there is no need for coding to achieve capacity. Similarly, the resulting joint distribution of $(S, \hat{S})$ is the one that achieves the rate-distortion function of the source.

Is this another particularity of the Gaussian distribution? The answer is no: as explained in Section 1.3.1, *any* input distribution achieves the capacity of a

[2]Shannon calls a *difference distortion measure* a distortion measure that depends only on the difference $s - \hat{s}$. Examples are the squared error $(s - \hat{s})^2$ or the absolute error $|s - \hat{s}|$.

given channel for *some* way of measuring the channel input cost. For example, the Gaussian distribution achieves the capacity of a Gaussian channel when the cost measure has the form $\rho(x) = ax + b$ (for arbitrary constants $a > 0$ and b), which applies to the input power measure used here. The uniform distribution also achieves the capacity of a Gaussian channel, just for a different cost measure. In the same way, *any* joint distribution of $(S, \hat{S})$ achieves the rate-distortion function of the source for *some* way of measuring the distortion.

3.3 Bandwidth Expansion

Does a similarly simple transmission scheme exist for the Gaussian case when the channel accepts more than one input for each source symbol, i.e., when $\tau_c < \tau_s$? Unfortunately, no. In fact, the performance of any code that encodes a single Gaussian source symbol into $n > 1$ inputs to a Gaussian channel is strictly bounded away from the optimum given by (3.1) [19] (shown using a result by Ziv and Zakai [50]). The bound given in [19] is not necessarily tight, however, so the region of achievable cost and distortion pairs is not known when $k = 1$ and $n > 1$.

While conditions 1 and 6 of Theorem 1.2 can still be achieved by a simple linear encoder and a MMSE decoder (just as in Example 3.1), condition 5 is no longer trivially satisfied when $n > 1$. In fact, the difficulty in this situation is to deterministically encode one Gaussian source symbol into n independent Gaussian channel inputs. The result of [19] implies that not all conditions of Theorem 1.2 can be simultaneously satisfied when $k = 1$ and $n > 1$. This changes, however, if we modify the scenario and allow the encoder access to perfect feedback from the receiver, as the next section shows.

3.4 Optimality Through Feedback

If a single Gaussian source symbol is communicated using n sequential transmissions on a Gaussian channel and if there is a causal, noiseless feedback link from the receiver to the encoder as illustrated in Figure 3.2, then the i^{th} channel input symbol can depend on the past channel outputs $Y_1, \ldots, Y_{i-1}$ as well as on the source. Because feedback does not increase the capacity of the channel, the bound of Theorem 1.1 and thus of (3.2) still applies and so do the conditions of Theorem 1.2. The big advantage brought by the feedback, though, is that it permits a simple transmission scheme that has minimal delay, yet achieves the bound (3.1) with equality, as the following example, due to Schalkwijk and Kailath [30], demonstrates.

Example 3.2. *In this example, a memoryless Gaussian source of variance σ_S^2 is transmitted across n uses of a memoryless Gaussian channel with power constraint P and noise variance ${\sigma_Z}^2$. Define $E_0 = S$. In the i^{th} channel use*

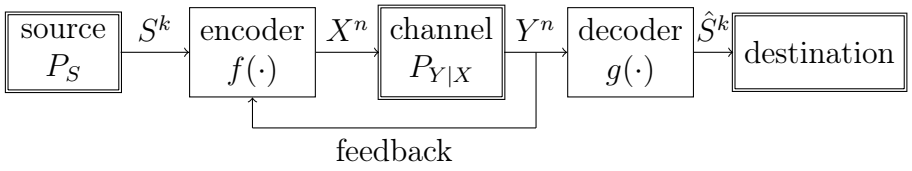


Figure 3.2: A source-channel communication system where the encoder has access to causal noiseless feedback from the receiver.

($i = 1, \ldots, n$), the encoder produces

$$X_i = \sqrt{\frac{P}{\operatorname{Var} E_{i-1}}} E_{i-1}. \tag{3.4}$$

Both the receiver and the sender now compute the minimum mean-squared error (MMSE) estimator $\hat{E}_{i-1}$ of E_{i-1} given Y_i. The sender then computes $E_i = \hat{E}_{i-1} - E_{i-1}$ and proceeds to the next round.

After n rounds of transmission, the receiver has n estimates $\hat{E}_0$ to $\hat{E}_{n-1}$. Using these, it computes the final estimate $\hat{S}$ as

$$\hat{S} = \hat{E}_0 - \hat{E}_1 + \hat{E}_2 - \cdots \pm \hat{E}_{n-1}. \tag{3.5}$$

(The sign of the last term is $+$ if n is even and $-$ if n is odd.)

To compute the overall distortion $\mathbb{E}[(\hat{S} - S)^2]$, note that $\hat{E}_{i-1} = E_{i-1} + E_i$ by definition, so (3.5) can be written as

$$\begin{aligned}\hat{S} &= (E_0 + E_1) - (E_1 + E_2) + (E_2 + E_3) - \cdots \pm (E_{n-1} + E_n) \\ &= E_0 \pm E_n,\end{aligned}$$

and since we have defined $E_0 = S$, we have $\mathbb{E}[(\hat{S} - S)^2] = \mathbb{E}[E_n^2]$, where E_n is the remaining error after the last round of transmission.

To compute $\mathbb{E}[E_n^2]$, note that since $\hat{E}_i$ is the MMSE estimator of E_i, the estimation error variance is given by (see e.g. [31, Section 8.3])

$$\mathbb{E}[E_{i+1}^2] = \mathbb{E}[(\hat{E}_i - E_i)^2] = \frac{\mathbb{E}[E_i^2]}{1 + P/{\sigma_Z}^2}. \tag{3.6}$$

Using $\mathbb{E}[E_0^2] = \mathbb{E}[S^2] = \sigma_S^2$ and recursively applying the above, we find that

$$\begin{aligned}\frac{\sigma_S^2}{\mathbb{E}[E_n^2]} &= \frac{\sigma_S^2(1 + P/{\sigma_Z}^2)}{\mathbb{E}[E_{n-1}^2]} \\ &= \frac{\sigma_S^2(1 + P/{\sigma_Z}^2)^2}{\mathbb{E}[E_{n-2}^2]} = \cdots \\ &= \frac{\sigma_S^2(1 + P/{\sigma_Z}^2)^n}{\mathbb{E}[E_0^2]} = (1 + P/{\sigma_Z}^2)^n,\end{aligned}$$

which is indeed the largest possible SDR according to (3.2).

As we have already seen before, achieving the capacity and the rate-distortion function (conditions 1 and 6 of Theorem 1.2) are just a matter of using the "right" cost measure and distortion measure, respectively. The transmission scheme of Example 3.2 in addition satisfies all other conditions of Theorem 1.2. In particular, it produces a sequence of independent channel outputs.[3] Again, we may ask if this is just due to some special property of the Gaussian distribution. And again, the answer is no: the next section explains how, using feedback, one can design a minimal-delay encoder that produces a sequence of independent, capacity achieving output symbols for any channel (and any cost measure).

3.4.1 Exploiting Feedback Via Posterior Matching

Example 3.2 showed how a simple transmission scheme can achieve the optimal distortion for a Gaussian source and channel if noiseless feedback is available. As an aside, we now show how it is possible to encode one source symbol into n channel inputs for *any* channel and *any* cost measure (under the condition that the source is continuous-valued). The underlying idea is well known and was used by Gastpar and Rimoldi [13] to develop various examples of optimal uncoded transmission with feedback (including the Gaussian example given in this chapter). Later, it was used by Shayevitz and Feder in 2007 [36, 37] (who baptized it *posterior matching*) to generalize the capacity achieving channel coding schemes of Schalkwijk and Kailath [30] and Horstein [18] to arbitrary channels with feedback.

Before continuing we prove some properties of cumulative distribution functions (CDFs).

Lemma 3.1. *Let X be a continuous random variable with density $f(x)$ and* CDF *F_X, i.e.,*

$$F_X(x) = \Pr[X \le x].$$

Then the random variable $Y = F_X(X)$ is uniformly distributed on $[0, 1]$.

Proof. Let $Y = F_X(X)$. Then $\Pr[Y \le y] = \Pr[F_X(X) \le y]$. Hence if $y < 0$ then $\Pr[Y \le y] = 0$, and if $y > 1$ then $\Pr[Y \le y] = 1$. If $y \in [0, 1]$ then

$$\begin{aligned} \Pr[F_X(X) \le y] &= \Pr[X \le F_X^{-1}(y)] \\ &= F_X(F_X^{-1}(y)) = y \end{aligned}$$

The CDF of Y is therefore that of a uniform random variable on $[0, 1]$. □

[3] It might not be obvious at first glance why the channel outputs are independent, given that Y_i depends on X_i, which is computed itself as a function of the past outputs. Note, however, that the encoder (3.4) is specifically chosen such that each X_i is Gaussian with zero mean and variance P, regardless of the past channel outputs. Thus, $p(x_i|y_1, \dots, y_{i-1})$ has the same distribution whatever the values of $y_1, \dots, y_{i-1}$. The key to understanding this is the distinction between statistical dependence and causal dependence: X_i is *causally* dependent on the past outputs, but this causal dependence is chosen such that it becomes *statistically* independent of them.

Lemma 3.2. *Let Y be a uniform random variable on $[0,1]$ and let F_X be the* CDF *of an arbitrary random variable X. If F_X is not invertible, define F_X^{-1} with a slight abuse of notation as*

$$F_X^{-1}(y) = \sup\{x : F_X(x) \le y\}. \tag{3.7}$$

Then the random variable $X' = F_X^{-1}(Y)$ has the same distribution as X.

Proof. The definition of F_X^{-1} according to (3.7) is such that $\{y : F_X^{-1}(y) \le x\} = \{y : y \le F_X(x)\}$. Thus,

$$\begin{aligned} F_{X'}(x) &= \Pr[F_X^{-1}(Y) \le x] \\ &= \Pr[Y \le F_X(x)] = F_X(x) \end{aligned}$$

since Y is uniformly distributed on $[0,1]$. □

Consider now a channel $P_{Y|X}$ and let $\pi(x)$ be the capacity achieving distribution at average cost P (or *a* capacity achieving distribution if there are multiple), i.e.,

$$\pi(x) = \arg \max_{p(x):\mathbb{E}[\rho(X)]\le P} I(X;Y)$$

for an arbitrary cost measure $\rho(x)$. The problem is to encode one source symbol of a continuous-valued source into n channel inputs, making use of the feedback.

Let F_π be the CDF of the distribution $\pi(x)$, and let F_S be the CDF of the source. In the first channel use, the encoder produces

$$X_1 = {F_\pi}^{-1}(F_S(S)), \tag{3.8}$$

where ${F_\pi}^{-1}$ is the inverse of F_π according to (3.7). By Lemma 3.1, if S is continuous then $F_S(S)$ has uniform distribution on $[0,1]$, and so by Lemma 3.2, ${F_\pi}^{-1}(F_S(S))$ is a random variable with CDF F_π.

After $i-1$ rounds of transmission, the encoder knows $y_1, \ldots, y_{i-1}$ and can compute the conditional CDF $F_{S|y_1,\ldots,y_{i-1}}$. It then sends

$$X_i = {F_\pi}^{-1}(F_{S|y_1,\ldots,y_{i-1}}(S)). \tag{3.9}$$

Again, since S is continuous, $F_{S|y_1,\ldots,y_{i-1}}(S)$ is uniform. For any $y_1, \ldots, y_{i-1}$, therefore,

$$p(x_i|y_1,\ldots,y_{i-1}) = \pi(x)$$

and so X_i is independent of $Y_1, \ldots, Y_{i-1}$.

Using this strategy the encoder produces an *iid* sequence of inputs X_i with the capacity achieving distribution $\pi(x)$, satisfying conditions 5 and 6 of Theorem 1.2; condition 3 of the theorem is trivially satisfied because the encoder is deterministic.

Let us now derive the posterior matching encoder for the communication system of Example 3.2.

Example 3.3. *First, a few properties of Gaussian* CDF*s are given. Let* $F_{\mathcal{N}(\mu,\sigma^2)}$ *be the* CDF *of a Gaussian random variable of mean* μ *and variance* σ^2 *and let* $F_{\mathcal{N}} \stackrel{def}{=} F_{\mathcal{N}(0,1)}$*. Then* $F_{\mathcal{N}(\mu,\sigma^2)}(x) = F_{\mathcal{N}}((x-\mu)/\sigma)$*. Consequently, the inverse* CDF *is*

$$F^{-1}_{\mathcal{N}(\mu,\sigma^2)}(y) = \sigma F^{-1}_{\mathcal{N}}(y) + \mu.$$

Let $\pi(x) = \mathcal{N}(0, P)$*. According to* (3.8)*, the first channel input is*

$$X_1 = \sqrt{P} F^{-1}_{\mathcal{N}}(F_{\mathcal{N}}(S/\sigma_S)) = \sqrt{\frac{P}{\sigma_S^2}} S,$$

which coincides with (3.4) *in Example 3.2 when* $i = 1$*.*

Given Y_1*,* S *is Gaussian with mean* $\mathbb{E}[S|Y_1]$ *and variance*

$$\mathrm{Var}(S|Y_1) = \mathbb{E}[(S - \mathbb{E}[S|Y_1])^2|Y_1] = \mathrm{Var}(S - \mathbb{E}[S|Y_1]),$$

since the error $S - \mathbb{E}[S|Y_1]$ *is orthogonal to* Y_1 *(according to the properties of the conditional mean). Following* (3.9)*, the second channel input is thus*

$$\begin{aligned} X_2 &= \sqrt{P} F^{-1}_{\mathcal{N}} \left(F_{\mathcal{N}} \left(\frac{S - \mathbb{E}[S|Y_1]}{\sqrt{\mathrm{Var}(S - \mathbb{E}[S|Y_1])}} \right) \right) \\ &= \sqrt{P} \frac{S - \mathbb{E}[S|Y_1]}{\sqrt{\mathrm{Var}(S - \mathbb{E}[S|Y_1])}}. \end{aligned}$$

Continuing this way, the i^{th} *channel input is found to be*

$$X_i = \sqrt{P} \frac{S - \mathbb{E}[S|Y_1^{i-1}]}{\sqrt{\mathrm{Var}(S - \mathbb{E}[S|Y_1^{i-1}])}}. \tag{3.10}$$

That this is equal to (3.4) *can be seen as follows. In Example 3.2, write*

$$\begin{aligned} S &= E_0 + (E_1 - E_1) - (E_2 - E_2) + \cdots \pm (E_{i-2} - E_{i-2}) \\ &= (E_0 + E_1) - (E_1 + E_2) + (E_2 + E_3) - \cdots - E_{i-2} \\ &= \hat{E}_0 - \hat{E}_1 + \hat{E}_2 - \cdots - E_{i-2}. \end{aligned}$$

Since $\mathbb{E}[\hat{E}_j|Y_1^{i-1}] = \hat{E}_j$ *for* $j = 1, \ldots, i-2$*, and* $\mathbb{E}[E_{i-2}|Y_1^{i-1}] = \hat{E}_{i-2}$*,*

$$\mathbb{E}[S|Y_1^{i-1}] = \hat{E}_0 - \hat{E}_1 + \cdots - \hat{E}_{i-2}$$

and so $S - \mathbb{E}[S|Y_1^{i-1}] = \hat{E}_{i-2} - E_{i-2} = E_{i-1}$*. Plugging this into* (3.10) *yields exactly the encoder* (3.4) *of Example 3.2.*

This example shows that the Gaussian example is nothing but a special case of posterior matching, with the particular property that the posterior matching encoder is linear.

3.4.2 Can Posterior Matching Help for Source Coding?

Using posterior matching, one can turn an arbitrary source distribution into the capacity achieving distribution. Can the same trick be used to make the conditional distribution of $\hat{S}$ given S achieve the rate distortion function?

For simplicity assume $n = 1$ (whether there is feedback or not is irrelevant). For a fixed D, let

$$\Phi_s(\hat{s}) = \arg \min_{p(\hat{s}|s):\mathbb{E}[d(S,\hat{S})]\leq D} I(S;\hat{S}),$$

i.e., $\Phi_s(\hat{s})$ is the conditional distribution of $\hat{S}$ given S that achieves the rate distortion function at expected distortion D. Let the decoder be

$$g(y) = F^{-1}_{\Phi_s}(F_{Y|S=s}(y)). \tag{3.11}$$

Given $S = s$, $g(Y)$ thus has the distribution Φ_s, and the resulting joint distribution of $\hat{S} = g(Y)$ and S satisfies $I(S;\hat{S}) = R(D)$.

It is immediately clear that this approach cannot work – both CDFs needed to implement this decoder depend on the actual value of s, which is obviously not known at the decoder (there would not really be a communication problem otherwise). Interestingly, though, in the Gaussian case the dependence on s of F_{Φ_s} and of $F_{Y|S=s}$ cancel each other out, and the decoder (3.11) yields the MMSE decoder, as the following example shows.[4]

Example 3.4. *Let the source S be distributed as $\mathcal{N}(0,1)$ and let the channel be AWGN with noise variance 1 and input constraint $\mathbb{E}[X^2] \leq P$. The distortion is the squared error. The smallest achievable distortion is*

$$D_{\min} = \frac{1}{1+P}. \tag{3.12}$$

The capacity-achieving input distribution is $\mathcal{N}(0,P)$, and the conditional distribution of $\hat{S}$ given $S = s$ that achieves the rate-distortion function at distortion D is $\mathcal{N}((1-D)s, D(1-D))$ (see e.g. [6]). Let $X = \sqrt{P}S$. The decoder from (3.11) is

$$\begin{aligned} g(y) &= F^{-1}_{\Phi_s}(F_{Y|S=s}(y)) \\ &= \sqrt{D(1-D)}F^{-1}_{\mathcal{N}}\left(F_{\mathcal{N}}\left(y-\sqrt{P}s\right)\right) + (1-D)s \\ &= \sqrt{D(1-D)}\left(y-\sqrt{P}s\right) + (1-D)s. \end{aligned}$$

This expression still depends on s. If we plug in the optimal distortion $D_{\min}$ from (3.12), however, the decoder becomes

$$\begin{aligned} g(y) &= \frac{\sqrt{P}}{P+1}(y-\sqrt{P}s) + \frac{P}{P+1}s \\ &= \frac{\sqrt{P}y}{P+1}, \end{aligned}$$

[4] Simulations for other sources and channels have confirmed that the dependence on s is really only cancelled out in the Gaussian case.

which no longer depends on s. Furthermore, this decoder is the MMSE decoder.

3.5 Lessons for the Case Without Feedback

In the previous sections we have seen that if more than one channel use is available per source symbol then a Gaussian source can be transmitted optimally over a Gaussian channel at minimal delay only when the encoder has feedback from the receiver. What lessons can we draw from the feedback case that help us in the case without feedback?

The first observation is that if the encoder knows the state of the receiver then it can transmit the receiver's current estimation error without coding. Indeed, suppose that after $n-1$ channel uses the receiver has a preliminary estimate $\hat{S}'$ and suppose the transmitter knows $\hat{S}'$ (ignore for now the question *how* the transmitter comes to know this estimate). Then the transmitter can use uncoded transmission to send the error $E \stackrel{\text{def}}{=} \hat{S}' - S$ in the last channel use. Upon receiving the corresponding channel output, the receiver computes an estimate $\hat{E}$ of E and sets the final estimate of S to be $\hat{S} = \hat{S}' - \hat{E}$. This results in the overall error $\hat{S} - S = \hat{S}' - E - S + E - \hat{E} = E - \hat{E}$ and an SDR of

$$\text{SDR} = \frac{\sigma_S^2}{\mathbb{E}[(\hat{S}-S)^2]} = \frac{\sigma_S^2}{\mathbb{E}[(\hat{S}'-S)^2]} \cdot \frac{\mathbb{E}[(\hat{S}'-S)^2]}{\mathbb{E}[(\hat{E}-E)^2]}.$$

The first term on the right hand side is SDR', the SDR after $n-1$ channel uses. The second term, equal to $\mathbb{E}[E^2]/\mathbb{E}[(\hat{E}-E)^2]$, is the SDR resulting from the uncoded transmission of E, which, as seen in Example 3.1, scales linearly with the SNR. Hence, the overall SDR scales as $\text{SDR}' \cdot \text{SNR}$ and so uncoded transmission in the last channel use boosts the SDR by a factor SNR.

In Example 3.1 the transmitter trivially knows the state of the receiver before transmitting anything (one can assume the receiver's initial estimate is $\hat{S}' = \mathbb{E}[S] = 0$, with a corresponding SDR' of σ_S^2). In Example 3.2, the transmitter knows the receiver's estimate at each step via the feedback link.

Feedback is not the only option for the transmitter to know the receiver's state after $n-1$ transmissions, however. Using coding, one can transform an unreliable channel into a reliable connection. Suppose a perfect source code combined with a perfect channel code is used to transmit S across the first $n-1$ channel uses. This implies a decomposition of S as $S = Q + E$, where Q is transmitted error free and the associated distortion $\mathbb{E}[E^2]$ scales as $\text{SNR}^{-(n-1)}$ (cf. Equation 3.2 on page 33). If E is transmitted uncoded in the n^{th} channel use and the receiver sets $\hat{S} = Q + \hat{E}$, the overall SDR is

$$\text{SDR} = \frac{\sigma_S^2}{\mathbb{E}[(\hat{S}-S)^2]} = \frac{\sigma_S^2}{\mathbb{E}[E^2]} \cdot \frac{\mathbb{E}[E^2]}{\mathbb{E}[(\hat{E}-E)^2]},$$

which scales as $\text{SNR}^{n-1} \cdot \text{SNR} = \text{SNR}^n$.

Naturally, the constraint that a single source symbol must be encoded at a time prevents the use of a perfect source and channel code, which would

require large block lenghts. Suppose thus that the receiver decodes not Q after $n-1$ channel uses, but that it has only an estimate $\hat{Q}$ and that it sets $\hat{S} = \hat{Q} + \hat{E}$. In that case the overall SDR is

$$\text{SDR} = \frac{\sigma_S^2}{\mathbb{E}[(\hat{Q}-Q)^2] + \mathbb{E}[(\hat{E}-E)^2]} = \frac{\sigma_S^2}{\mathbb{E}[E^2]} \cdot \frac{\mathbb{E}[E^2]}{\mathbb{E}[(\hat{Q}-Q)^2] + \mathbb{E}[(\hat{E}-E)^2]}. \tag{3.13}$$

If now $\mathbb{E}[(\hat{Q}-Q)^2] \in O(\mathbb{E}[(\hat{E}-E)^2])$ as $\text{SNR} \to \infty$, i.e., if the error in estimating Q is dominated by that of estimating E then the SDR scales as

$$\frac{\sigma_S^2}{\mathbb{E}[E^2]} \cdot \frac{\mathbb{E}[E^2]}{\mathbb{E}[(\hat{E}-E)^2]}. \tag{3.14}$$

The second term is again the SDR from uncoded transmission of E and scales as SNR; overall, the SDR behaves thus as

$$\text{SDR} \approx \frac{\sigma_S^2 \,\text{SNR}}{\mathbb{E}[E^2]}. \tag{3.15}$$

There is a tradeoff in how Q (and E) are chosen as a function of S. If Q is such that $\mathbb{E}[(\hat{Q}-Q)^2]$ decreases too slowly with increasing SNR (for example because Q is obtained by quantizing the source with too fine a resolution), the approximation (3.15) is not valid and (3.13) scales instead as $\sigma_S^2/\mathbb{E}[(\hat{Q}-Q)^2]$, so the SDR gain from the uncoded transmission is lost. If, on the other hand, $\mathbb{E}[(\hat{Q}-Q)^2]$ decreases too fast then $\mathbb{E}[E^2]$ decreases only slowly and the overall SDR, $\text{SNR}/\mathbb{E}[E^2]$, grows only slowly with SNR. It is clear, then, that the best SDR scaling is obtained when $\mathbb{E}[(\hat{Q}-Q)^2]$ scales the same as $\mathbb{E}[(\hat{E}-E)^2]$.

The conclusion from all this is that to take advantage of uncoded communication in the last channel use, error free communication in the first $n-1$ channel uses is not necessary. All that is needed is that the error from the first $n-1$ channel uses be dominated by that of the last channel use as $\text{SNR} \to \infty$.

The traditional way to analyze minimal-delay transmission strategies is through a geometric analysis of the signal curve, which was first suggested by Shannon [34] and treated in much detail by Wozencraft and Jacobs [47]. It turns out that such a geometric analysis leads to the same tradeoff as that represented by (3.13), as the next section shows.

3.6 Connection to the Geometric Point of View

If $f(s)$ is an encoding function that maps a single source symbol into n channel inputs, one can gain insights about its performance by studying the set $\{f(s) : s \in \mathcal{S}\}$ (where $\mathcal{S}$ is the support set of the source). This set is called the *signal curve* or *signal locus* corresponding to $f(s)$. A sample signal locus is shown in

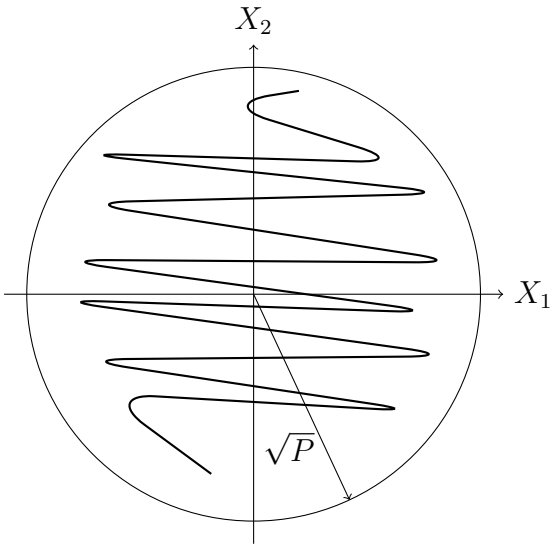


Figure 3.3: The signal locus corresponding to a nonlinear encoder from $\mathbb{R}$ to $\mathbb{R}^2$. A power constraint P signifies that the signal must roughly be contained within a sphere of radius $\sqrt{P}$.

Figure 3.3. This section, which is largely drawn from Wozencraft & Jacobs [47], reviews the basics of the geometric signal curve analysis.

The estimate that minimizes the mean squared error is the conditional mean $\mathbb{E}[S|Y^n]$. For nonlinear encoders, closed-form evaluation of the corresponding estimation error is in general hopelessly complicated. A decoder whose performance is easier to evaluate is the *maximum likelihood* (ML) decoder. It computes $\hat{s} = \arg\max_s f(\mathbf{y}|s)$, where $f(\mathbf{y}|s)$ is the conditional pdf of $\mathbf{Y} = (Y_1, \ldots, Y_n)$ given S. For the AWGN channel, $\mathbf{Y} = f(S) + \mathbf{Z}$, where $\mathbf{Z}$ is circularly symmetric Gaussian noise whose orthogonal components have variance ${\sigma_Z}^2$. In this case $f(\mathbf{y}|s)$ is a decreasing function of $\|\mathbf{y} - f(s)\|$, so the ML decoder can equivalently be described by $\hat{s} = \arg\min_s \|\mathbf{y} - f(s)\|$. It decides thus for $\hat{s}$ such that $f(\hat{s})$ is the point on the signal curve closest to $\mathbf{y}$.

Suppose now that the transmitted point is $f(s_0)$ and suppose further that $f(s)$ is differentiable in s_0. Then the signal curve can be linearly approximated in a small neighborhood around s_0 as

$$f(s_0 + \Delta) \approx f(s_0) + \Delta \left.\frac{df(s)}{ds}\right|_{s=s_0}. \tag{3.16}$$

If the noise is small, a valid approximation of the ML estimate is therefore the projection of the received point onto the tangent through $f(s_0)$ (see Figure 3.4). Writing $\hat{s} = s_0 + \Delta$, the estimation error is $\hat{s} - s_0 = \Delta$ and by (3.16) satisfies

$$\|\Delta\|^2 \approx \frac{\|f(s_0 + \Delta) - f(s_0)\|^2}{\left\| \left.\frac{df(s)}{ds}\right|_{s=s_0} \right\|^2}.$$

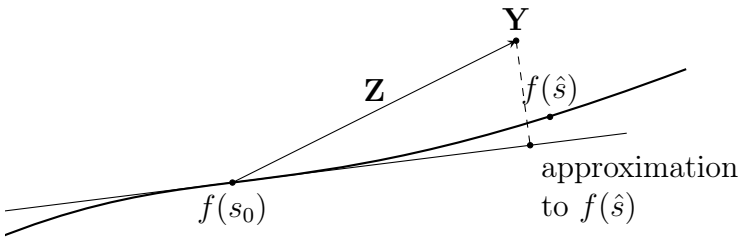


Figure 3.4: If the noise level is small, the ML estimate can be approximated by projecting the noise vector $\mathbf{Z}$ onto a tangent through $f(s_0)$.

The term $\|f(s_0+\Delta)-f(s_0)\|$ is the length of the noise vector $\mathbf{Z}$ projected onto the tangent through $f(s_0)$ (cf. Figure 3.4). It is Gaussian with variance ${\sigma_Z}^2$, so the average squared error given $S=s_0$ is approximately

$$\mathbb{E}[(\hat{S}-S)^2 \mid S=s_0] \approx \frac{{\sigma_Z}^2}{\left\| \frac{df(s)}{ds}\Big|_{s=s_0} \right\|^2}. \tag{3.17}$$

Expressing $f(s)=\sqrt{P}\tilde{f}(s)$, with $\mathbb{E}[\|\tilde{f}(S)\|^2] \le 1$, its derivative is $df(s)/ds = \sqrt{P}d\tilde{f}(s)/s$. The quantity $l(s) \stackrel{\text{def}}{=} d\tilde{f}(s)/ds$ is called the *stretch factor* or simply the *stretch* of the signal locus. If the stretch does not depend on s, the overall MSE simplifies to

$$\mathbb{E}[(\hat{S}-S)^2] \approx (\mathrm{SNR}\, l^2)^{-1} \tag{3.18}$$

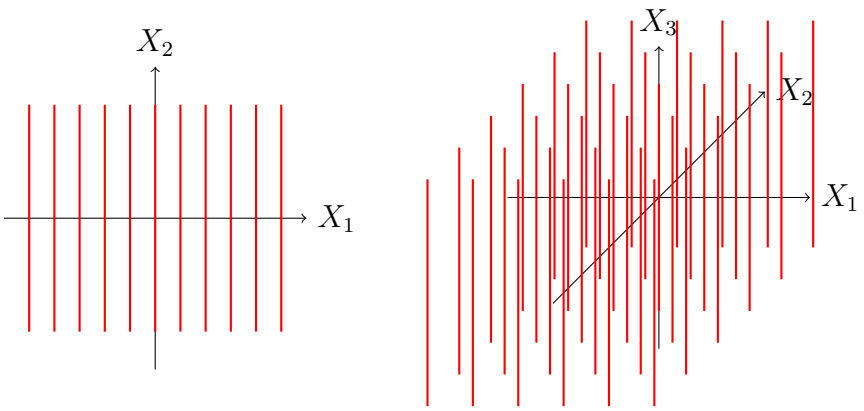
or equivalently

$$\mathrm{SDR} \approx \sigma_S^2 \,\mathrm{SNR}\, l^2. \tag{3.19}$$

As long as the noise level is small with respect to the distance between different folds of the signal curve, (3.17) is a valid approximation for the achieved MSE, and the latter decreases with growing stretch. It is clear, though, that the MSE cannot decrease arbitrarily: a larger stretch implies a longer signal curve, so the folds of the curve must be placed closer together to satisfy the power constraint. If the stretch becomes too large, the small noise assumption will no longer be valid. On the other hand, if the SDR is to scale more than linearly with the SNR, then by (3.17) the stretch must increase with the SNR.

The choice of the stretch represents a tradeoff: a bigger stretch results in a smaller error, provided that the correct fold of the signal curve is decoded, but a bigger stretch also increases the probability that the wrong fold of the curve is decoded. Optimally, thus, the stretch is such that the two kinds of errors contribute equally to the overall error.

There are clear parallels to the previous point of view. In the feedback-inspired discussion, the error from decoding Q had to be dominated by that



(a) $n = 2$ (b) $n = 3$

Figure 3.5: If uncoded transmission is used only in the last channel use, the signal locus consists of parallel straight lines, aligned with the axis of X_n. This is shown here for $n = 2$ and $n = 3$, respectively.

from decoding E in order for the approximation (3.15) to work. Here, the error due to decoding the wrong fold of the curve must be dominated by that from decoding the wrong point on the same fold, in order for the approximation (3.17) to be valid. Note in particular the striking similarity between (3.19) and (3.15): both equations express the SDR as the product of the SNR and a factor that should grow with the SNR but whose growth rate is limited by the conditions of the underlying approximation.

Remark 3.1. The geometric perspective provides a simple argument why a linear encoder cannot achieve an MSE scaling better than SNR^{-1}, regardless of the number of channel uses. The image of a linear function from $\mathbb{R}$ to $\mathbb{R}^n$ is a straight line (a subspace of dimension 1). Applying invertible transforms at the encoder and decoder, this straight line can be rotated to lie on the X_1 axis without changing the performance. The resulting constellation is such that all X_i for $i = 2, \ldots, n$ are zero. Effectively, thus, only a single transmission is made on the channel. The same reasoning implies that a linear encoder from $\mathbb{R}^k$ to $\mathbb{R}^n$, where $k < n$, achieves a MSE scaling of only SNR^{-k}, regardless of n.

3.7 Towards a Hybrid Communication Strategy

Translated to the geometric perspective, the principle that uncoded transmission be used only in the last channel use means that the signal curve should consist of parallel straight line segments, aligned along the axis of X_n. This is illustrated in Figure 3.5.

One of the simplest ways to obtain such a signal curve is by means of hierarchical quantization. The first channel input is obtained by passing the

source through a uniform quantizer. The resulting quantization error is scaled up and quantized itself to yield the second channel input. This procedure is repeated a total of $n-1$ steps. Finally, the remaining quantization error is again scaled up and transmitted uncoded in the n^{th} channel use. The next chapter formalizes this idea and provides an exact characterization of the resulting asymptotic performance.

3.A Asymptotic Notation

Definition 3.3. *Let $f(x) \geq 0$ and $g(x) \geq 0$ be two functions defined on $\mathbb{R}$. The set $O(g(x))$ is defined as*

$$f(x) \in O(g(x))$$

if and only if there exists an x_0 and a constant c such that

$$f(x) \leq cg(x)$$

for all $x > x_0$. Similarly, $f(x) \in \Omega(g(x))$ if $\leq$ is replaced by $\geq$ in the above definition. Finally, $\Theta(g(x)) \stackrel{def}{=} O(g(x)) \cap \Omega(g(x))$.

A Hybrid Communication Strategy for Gaussian Channels

4

Based on the ideas outlined in the previous chapter, this section introduces a simple communication strategy. The strategy consists of splitting a single source symbol into $n-1$ discrete parts and a continous part (hence the term "hybrid"), to be transmitted, respectively, in the first $n-1$ and in the last channel use. While this strategy emerged as a natural way to take advantage of the lessons learned from the feedback case, it turns out that the method of combining quantization and uncoded transmission as done here is not new. It was first suggested by McRae [24] 1971; for more detailed historical references see Section 4.7 at the end of the chapter. [1]

4.1 Transmission Strategy

The communication strategy described hereafter is displayed schematically in Figure 4.1. To encode a single source letter S into n channel input symbols $X_1, \ldots, X_n$, it proceeds as follows. Define $E_0 = S$ and recursively compute the pairs (Q_i, E_i) as

$$Q_i = \frac{1}{\beta} \operatorname{int}(\beta E_{i-1}) \tag{4.1a}$$

$$E_i = \beta(E_{i-1} - Q_i) \tag{4.1b}$$

for $i = 1, \ldots, n-1$ where $\operatorname{int}(x)$ is the unique integer i satisfying

$$x \in \left[i - \frac{1}{2}, i + \frac{1}{2} \right).$$

[1] The main results of this chapter have been published in [20] and [22].

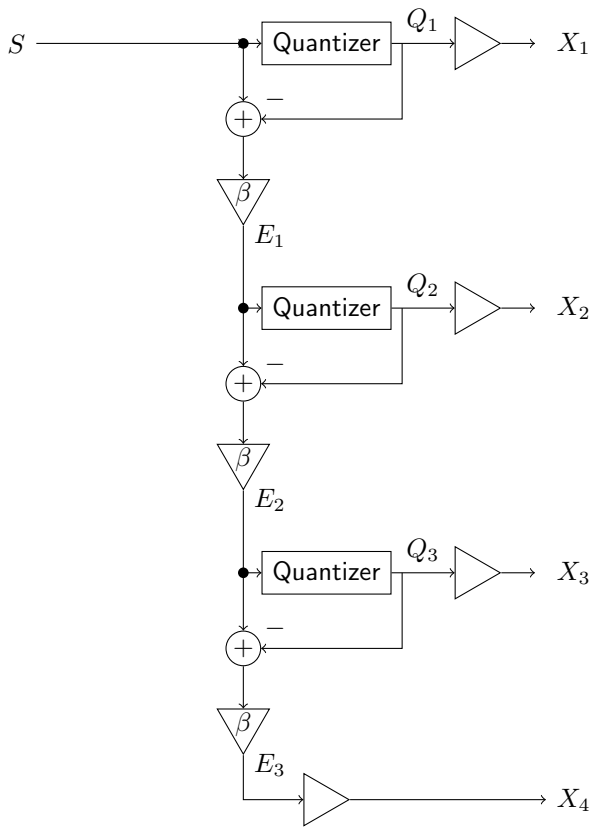


Figure 4.1: Schematic display of the encoder described in Section 4.1 for $n = 4$. The triangles represent the scaling operations of (4.1b) and (4.4).

The equations 4.1 define a *hierarchical quantization* of the source. Q_1 is the quantized source symbol and E_1 the associated quantization error. Q_2 is the quantized version of the previous quantization error E_1, and so on. E_{n-1} is the remaining quantization error after $n-1$ steps.

The parameter $\beta \in \mathbb{N}$ determines the quantization resolution; the larger β, the finer the quantization. Equation 4.1 implies a partition of the source space into intervals of length $1/\beta^{n-1}$, as illustrated in Figure 4.2 on the next page. The Q_i determine the interval that contains S, and E_{n-1} determines the position of S within an interval.

The following result will be useful in the sequel.

Lemma 4.1. *For all $i = 1, \ldots, n-1$, the variance of Q_i satisfies*

$$\mathbb{E}[Q_i^2] \leq \mathbb{E}[E_{i-1}^2] + \frac{\sqrt{\mathbb{E}[E_{i-1}^2]}}{\beta} + \frac{1}{4\beta^2}.$$

Proof. See Appendix 4.A. □

Proposition 4.2. *The Q_i and E_i satisfy the following properties:*

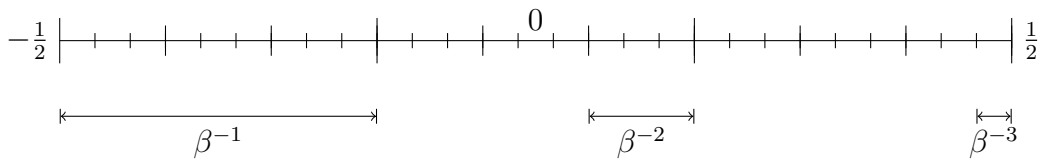


Figure 4.2: Example partition of the source space $[-1/2, 1/2]$ for $\beta = 3$ and $n = 4$.

1. *The map* $S \mapsto (Q_1, \ldots, Q_{n-1}, E_{n-1})$ *is one-to-one, with the inverse given by*

$$S = \sum_{i=1}^{n-1} \frac{1}{\beta^{i-1}} Q_i + \frac{1}{\beta^{n-1}} E_{n-1}. \tag{4.2}$$

2. *There exists a constant* $\gamma > 0$ *such that* $\operatorname{Var} Q_i \leq \gamma^2$ *and* $\operatorname{Var} E_i \leq \gamma^2$ *for all* i*, regardless of the value of* β*.*

Proof. 1. From the definition (4.1b), with $E_0 = S$,

$$E_{i-1} = \frac{1}{\beta} E_i + Q_i. \tag{4.3}$$

Repeated use of this relationship leads to the given expression for S.

2. First, $\operatorname{Var} E_0 = \operatorname{Var} S = \sigma_S^2$, which doesn't depend on β. For $i = 1, \ldots, n-1$, $E_i \in [-1/2, 1/2)$ and so its variance is upper bounded as well. Finally, since $\beta \geq 1$ and by Lemma 4.1,

$$\operatorname{Var} Q_i \leq \mathbb{E}[Q_i^2] \leq \mathbb{E}[E_{i-1}^2] + \sqrt{\mathbb{E}[E_{i-1}^2]} + \frac{1}{4}.$$

which completes the proof.

□

To complete the description of the transmission strategy, the Q_i and E_{n-1} are scaled to satisfy the power constraint, resulting in the channel inputs

$$\begin{aligned} X_i &= (\sqrt{P}/\gamma) Q_i \quad \text{for } i = 1, \ldots, n-1 \text{ and} \\ X_n &= (\sqrt{P}/\gamma) E_{n-1}. \end{aligned} \tag{4.4}$$

Following Proposition 4.2, this ensures that $\mathbb{E}[X_i^2] \leq P$ for all i.

Remark 4.1. The signal locus resulting from this encoding scheme is precisely the one illustrated in Figure 3.5. Each of the segments of length $\beta^{-(n-1)}$ of the source (cf. Figure 4.2) is mapped into a line segment of length $\sqrt{P}/\gamma$; the stretch of the signal curve, as defined in Section 3.6, is therefore β^{n-1}/γ.

4.2 Lower Bound on the Mean Squared Error

When an analog source is transmitted across a Gaussian channel using the scheme just presented, two types of decoding errors can occur. Either the decoded point is on the same line segment as the transmitted point (i.e., all the Q_i are decoded correctly) or it is on a different segment. Which one of these errors dominates the overall error behavior depends on how the parameter β is chosen as a function of the SNR: if β grows fast with the SNR the quantization resolution quickly becomes very fine, and the line segments of the constellation move close together fast, so the probability of decoding the wrong segment is high. On the other hand, since the stretch of the signal curve is proportional to β^{n-1} (Remark 4.1), the error decreases fast in case the correct segment is decoded. Conversely, if β grows only slowly with the SNR then decoding the wrong segment is unlikely. But because the stretch also only grows slowly, the error when the correct segment is decoded decreases only slowly.

The following results make these reflections precise. The results apply to any source; provided that there is an interval on which the source distribution admits a density. Based on these results one can choose β as a function of the SNR in order to optimize the MSE scaling.

Remark 4.2. Throughout this section it is assumed that $\beta = \lceil \mathrm{SNR}^{(1-\epsilon)/2} \rceil$, where $\epsilon = \epsilon(\mathrm{SNR})$ is a positive function of SNR. This results in no loss of generality, since for an arbitrary positive function f one can set $\epsilon(\mathrm{SNR}) = 1 - 2\log(f(\mathrm{SNR}))/\log \mathrm{SNR}$ to get $\beta(\mathrm{SNR}) = \lceil f(\mathrm{SNR}) \rceil$. Writing β in this form will slightly simplify the mathematical derivations to follow. Note that one can bound β by $\mathrm{SNR}^{(1-\epsilon)/2} \leq \beta \leq \mathrm{SNR}^{(1-\epsilon)/2} + 1$, which implies $\beta \in \Theta(\mathrm{SNR}^{(1-\epsilon)/2})$ (the Θ-notation is defined in Appendix 3.A). Choosing an optimal $\beta(\mathrm{SNR})$ is therefore equivalent to choosing an optimal $\epsilon(\mathrm{SNR})$.

Remark 4.3. Note that by (4.1) the Q_i are completely determined by S. With a slight abuse of notation, $Q_i(s)$ is therefore used in the sequel to denote the value of Q_i when $S = s$. $E_i(s)$ and $X_i(s)$ are defined in an analogous manner. Furthermore, $\mathbf{X}(s) \stackrel{\text{def}}{=} (X_1(s), \ldots, X_n(s))$.

To obtain a lower bound on the MSE that holds for all possible decoders, the obvious thing to do is to assume a minimum mean squared error (MMSE) decoder. As mentioned in this chapter's introduction, though, the MMSE decoder is in general hard to evaluate mathematically. This is no different for the scheme at hand. Fortunately the following lemma, due to Ziv [48], presents a way around the MMSE decoder. Moreover, unlike the MMSE decoder, it does not depend on the source distribution.

Lemma 4.3. *Consider a communication system where a continuous-valued source S is encoded into an n-dimensional vector $\mathbf{X}(S)$, sent across n independent parallel AWGN channels with noise variance ${\sigma_Z}^2$, and decoded at the receiver to produce an estimate $\hat{S}$. If the density p_S of the source is such that there exists an interval $[A, B]$ and a number $p_{\min} > 0$ such that $p_S(s) \geq p_{\min}$*

whenever $s \in [A, B]$, then for any $\Delta \in [0, B - A)$ the mean squared error incurred by the communication system satisfies

$$\mathbb{E}[(\hat{S} - S)^2] \geq p_{\min} \left(\frac{\Delta}{2}\right)^2 \int_A^{B-\Delta} Q(d(s, \Delta)/2\sigma_Z) ds, \tag{4.5}$$

where $d(s, \Delta) \stackrel{def}{=} \|\mathbf{X}(s) - \mathbf{X}(s + \Delta)\|$ and

$$Q(x) = \int_x^\infty (1/\sqrt{2\pi}) \exp\{-\xi^2/2\} d\xi.$$

Proof. See Appendix 4.B. □

The next two lemmas provide two different asymptotic lower bounds on the mean squared error of the hybrid transmission strategy considered. They hold regardless of the decoder used. (The Ω-notation is defined in Appendix 3.A.)

Lemma 4.4. *Given a hybrid transmission strategy characterized by $\epsilon(\text{SNR}) \geq 0$, the mean squared error satisfies*

$$\mathbb{E}[(\hat{S} - S)^2] \in \Omega(\text{SNR}^{-n+(n-1)\epsilon}).$$

Lemma 4.5. *Given a hybrid transmission strategy characterized by $\epsilon(\text{SNR}) \geq 0$, the mean squared error satisfies*

$$\mathbb{E}[(\hat{S} - S)^2] \in \Omega(\text{SNR}^{-1+\epsilon/2} \exp\{-c\,\text{SNR}^\epsilon\}),$$

where $c > 0$ does not depend on SNR.

Discussion: An immediate consequence of the lemmas is that the scaling SNR^{-n} is not achievable with the given encoding strategy: by Lemma 4.4 this would require $\epsilon = 0$, but following Lemma 4.5 the scaling is at best SNR^{-1} if $\epsilon = 0$. More generally, which one of the two lower bounds decays more slowly and is therefore tighter depends on the scaling of $\epsilon(\text{SNR})$. How to choose $\epsilon(\text{SNR})$ optimally will be the subject of Theorem 4.7.

Proof of Lemma 4.4. Assume $\Delta \in [0, \beta^{-(n-1)})$ and define for $j \in \mathbb{Z}$

$$\mathcal{I}_j^\Delta = \left[(j - \frac{1}{2})\beta^{-(n-1)}, (j + \frac{1}{2})\beta^{-(n-1)} - \Delta\right).$$

These intervals will be used to partition the source space, as illustrated in Figure 4.3.

It can be verified from (4.1) that if $s \in \mathcal{I}_j^\Delta$ for some j, the following properties hold: 1) $Q_i(s) = Q_i(s+\Delta)$ for $i = 1, \ldots, n-1$, and 2) $E_{n-1}(s+\Delta) - E_{n-1}(s) = \beta^{n-1}\Delta$. Geometrically this means that s and $s + \Delta$ are mapped to the same straight line segment; see Figure 4.4a. From (4.4) it follows that $s \in \mathcal{I}_j^\Delta$ implies

$$d(s, \Delta) = \|\mathbf{X}(s) - \mathbf{X}(s + \Delta)\| = \sqrt{P/\gamma^2}\beta^{n-1}\Delta.$$

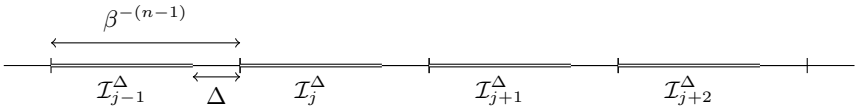


Figure 4.3: Illustration of the intervals $\mathcal{I}_j^{\Delta}$ in the proof of Lemma 4.4. The horizontal line represents a subset of the source space. The length of each $\mathcal{I}_j^{\Delta}$ is $\beta^{-(n-1)} - \Delta$, and they are placed such that within an interval of length ℓ there are approximately $\ell\beta^{n-1}$ of them.

Now apply Lemma 4.3 and restrict the integral to the set $\psi(\Delta) \stackrel{\text{def}}{=} [A, B - \Delta) \cap \bigcup_{j\in\mathbb{Z}} \mathcal{I}_j^{\Delta}$. The lower bound is then relaxed to give

$$\mathbb{E}[(\hat{S} - S)^2] \geq \frac{p_{\min}}{4}\Delta^2 Q(\sqrt{\mathrm{SNR}\,/\gamma^2}\beta^{n-1}\Delta/2) \int_{\psi(\Delta)} ds.$$

Letting $\Delta = 1/(\sqrt{\mathrm{SNR}}\beta^{n-1})$ and using $\beta^2 \in \Theta(\mathrm{SNR}^{1-\epsilon})$ (see Remark 4.2) yields (for sufficiently large SNR)

$$\mathbb{E}[(\hat{S} - S)^2] \geq c\,\mathrm{SNR}^{-n+(n-1)\epsilon}\, Q\left(1/2\gamma\right) \int_{\psi(\Delta)} ds.$$

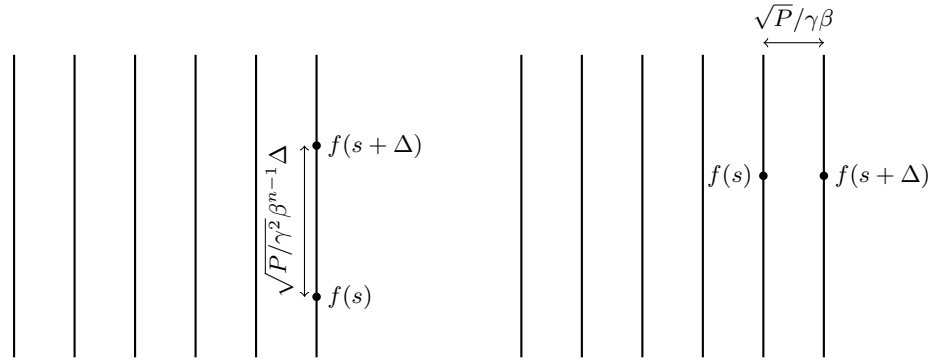
The proof is almost complete; it only remains to show that $\int_{\psi(\Delta)} ds$ can be lower bounded by a constant for large SNR. The length of a single interval $\mathcal{I}_j^{\Delta}$ is $\beta^{-(n-1)} - \Delta$. Within $[A, B - \Delta)$ there are $(B - A - \Delta)\beta^{n-1}$ such intervals (see Figure 4.3). The total length of all intervals $\mathcal{I}_j^{\Delta}$ in $[A, B - \Delta)$ is therefore

$$\int_{\psi(\Delta)} ds = (B - A - \Delta)(1 - \beta^{n-1}\Delta),$$

which, for the given values of β and Δ, converges to $B - A$ for $\mathrm{SNR} \to \infty$ and thus can be lower bounded by a constant for SNR greater than some SNR_0. With this, the proof is complete. □

Proof of Lemma 4.5. Observe first that (4.1) implies $Q_1(s+\beta^{-1}) = Q_1(s)+\beta^{-1}$ and $E_1(s+\beta^{-1}) = E_1(s)$. Since all Q_i and E_i for $i \geq 2$ are by recursion a function of E_1 only, $Q_i(s) = Q_i(s+\beta^{-1})$ for $i = 2, \ldots, n-1$, and $E_{n-1}(s) = E_{n-1}(s+\beta^{-1})$. Consequently, $X_i(s) = X_i(s+\beta^{-1})$ for all $i = 2, \ldots, n$. Geometrically speaking, s and $s+\beta^{-1}$ are mapped to the same position on two adjacent segments of the signal locus; see Figure 4.4b. By (4.4) and the above, the Euclidean distance between $\mathbf{X}(s)$ and $\mathbf{X}(s + \beta^{-1})$ is therefore

$$d(s, \beta^{-1}) = \frac{\sqrt{P}}{\gamma}|Q_1(s) - Q_1(s + \beta^{-1})| = \frac{\sqrt{P}}{\gamma\beta}. \tag{4.6}$$



(a) $\Delta \in [0, \beta^{-(n-1)})$ and $s \in \mathcal{I}_j^{\Delta}$ (b) $\Delta = \beta^{-1}$

Figure 4.4: Geometric view of the proofs of Lemma 4.4 (left) and Lemma 4.5 (right) for $n = 2$. In the former case, s and $s + \Delta$ are mapped to the same segment of the signal locus, whereas in the latter case they are mapped to adjacent segments but at the same "height".

Apply now Lemma 4.3 with $\Delta = \beta^{-1}$. The parameter β will be chosen to increase with the SNR, therefore $\Delta \in [0, B - A)$ holds for sufficiently large values of SNR. Using (4.6), the resulting bound on the mean squared error is

$$\mathbb{E}[(\hat{S} - S)^2] \geq \frac{p_{\min}}{4} \beta^{-2} Q\left(\frac{\sqrt{\mathrm{SNR}}}{2\gamma\beta}\right)(B - A - \beta^{-1}).$$

Because $\beta^2 \in \Theta(\mathrm{SNR}^{1-\epsilon})$ (see Remark 4.2), $\sqrt{\mathrm{SNR}}/\beta \in \Theta(\mathrm{SNR}^{\epsilon/2})$. If $\epsilon(\mathrm{SNR})$ is such that $\lim_{\mathrm{SNR}\to\infty} \mathrm{SNR}^{\epsilon(\mathrm{SNR})} = \infty$, use the fact that $Q(x)$ converges to $e^{-x^2/2}/\sqrt{2\pi}x$ (cf. [1, §26.2.12]). Otherwise $\mathrm{SNR}^{\epsilon(\mathrm{SNR})}$ is upper bounded by a constant, in which case $Q(x)$ is bounded above and below by constants for $x = \mathrm{SNR}^{\epsilon(\mathrm{SNR})}$, and $e^{-x^2/2}/x$ is equally upper and lower bounded. In any case, for sufficiently large values of SNR,

$$\mathbb{E}[(\hat{S} - S)^2] \geq c_1 \,\mathrm{SNR}^{-1+\epsilon/2} \exp\{-c_2 \,\mathrm{SNR}^{\epsilon}\},$$

with c_1 and c_2 positive constants that do not depend on SNR, thus proving the lemma. □

Remark 4.4. Figure 4.4 illustrates the connection to the geometric argument given in Section 3.6. The MSE is lower bounded by Lemma 4.4 in case the correct fold of the signal curve is decoded and by Lemma 4.5 in case the wrong fold is decoded. The parameter ϵ represents the tradeoff between the two kinds of errors.

The following lemma will be used to prove Theorem 4.7, the main result of this section.

Lemma 4.6. *For* $\mathrm{SNR} > 1$ *and arbitrary real constants* a, $b > 0$, *and* $c > 0$, *it holds that*

$$\mathrm{SNR}^{a+b\epsilon} = \exp\{-c\,\mathrm{SNR}^{\epsilon}\}, \tag{4.7}$$

if and only if

$$\mathrm{SNR}^{\epsilon} = (b/c)W(c\,\mathrm{SNR}^{-a/b}/b), \tag{4.8}$$

where $W(x)$ *is the function that satisfies* $W(x)e^{W(x)} = x$ *for* $x > 0$. *This function is well defined and is sometimes called the* Lambert W-function [5].

Proof. Let $\mathrm{SNR} > 1$. Since $\mathrm{SNR}^{a+b\epsilon}$ is strictly increasing and $\exp\{-c\,\mathrm{SNR}^{\epsilon}\}$ is strictly decreasing in SNR^{ϵ}, there is at most one solution to (4.7) in SNR^{ϵ}. Assume now SNR^{ϵ} is as in (4.8). Then

$$\exp\{-c\,\mathrm{SNR}^{\epsilon}\} = \exp\{-bW(c\,\mathrm{SNR}^{-a/b}/b)\}.$$

On the other hand,

$$\begin{aligned}\mathrm{SNR}^{a+b\epsilon} &= \mathrm{SNR}^{a}\left((b/c)W(c\,\mathrm{SNR}^{-a/b}/b)\right)^{b} \\ &= \left(W(c\,\mathrm{SNR}^{-a/b}/b)/(c\,\mathrm{SNR}^{-a/b}/b)\right)^{b}.\end{aligned}$$

By definition, $W(x)/x = e^{-W(x)}$, so the above is equal to

$$\mathrm{SNR}^{a+b\epsilon} = \exp\{-bW(c\,\mathrm{SNR}^{-a/b}/b)\},$$

which proves the claim. □

Lemmas 4.4 and 4.5 provide conflicting objectives for the choice of ϵ: according to Lemma 4.4, ϵ should be small for the MSE to decay fast, but according to Lemma 4.5 it should be large to have a fast exponential decay of the error. The following theorem, which is the main result of this section, is obtained by finding the ϵ for which the two lower bounds scale the same.

Theorem 4.7. *For any parameter* β *and for any decoder, the mean squared error of the hybrid transmission strategy described in this section satisfies*

$$\mathbb{E}[(\hat{S} - S)^2] \in \Omega(\mathrm{SNR}^{-n}(\log \mathrm{SNR})^{n-1}).$$

Proof. For notational simplicity define

$$\begin{aligned}l_1(\mathrm{SNR}, \epsilon) &= \mathrm{SNR}^{-n+(n-1)\epsilon} \quad \text{and} \\ l_2(\mathrm{SNR}, \epsilon) &= \mathrm{SNR}^{-1+\epsilon/2}\exp\{-c\,\mathrm{SNR}^{\epsilon}\}.\end{aligned}$$

By Lemmas 4.4 and 4.5,

$$\mathbb{E}[(\hat{S} - S)^2] \in \Omega\big(\max\left(l_1(\mathrm{SNR}, \epsilon), l_2(\mathrm{SNR}, \epsilon)\right)\big).$$

The optimal parameter $\epsilon(\mathrm{SNR})$ is therefore such that for any SNR

$$\max\left(l_1(\mathrm{SNR}, \epsilon), l_2(\mathrm{SNR}, \epsilon)\right) \tag{4.9}$$

is minimized. Now for any fixed SNR, $l_1(\text{SNR}, \epsilon)$ is increasing in ϵ and $l_2(\text{SNR}, \epsilon)$ is increasing in ϵ for $0 \leq \epsilon < \xi = \log(1/2c)/\log \text{SNR}$ and decreasing in ϵ for $\epsilon \geq \xi$. The maximum in (4.9) is therefore minimized either for $\epsilon = 0$ or for $\epsilon \geq \xi$ such that $l_1(\epsilon) = l_2(\epsilon)$. As remarked before, $\epsilon = 0$ leads to $\mathbb{E}[(\hat{S} - S)^2] \in \Omega(\text{SNR}^{-1})$. In the following, let thus $\epsilon(\text{SNR})$ be such that $l_1(\text{SNR}, \epsilon) = l_2(\text{SNR}, \epsilon)$, to see whether this gives a better lower bound. Inserting the definitions of l_1 and l_2 and rearranging the terms yields

$$\text{SNR}^{-(n-1)+(n-3/2)\epsilon} = \exp\{-c\,\text{SNR}^{\epsilon}\},$$

which is of the form (4.7) with $a = -(n-1)$ and $b = n - 3/2$. By Lemma 4.6, for $\text{SNR} > 1$,

$$\text{SNR}^{\epsilon} = \frac{n-3/2}{c} W\left(\frac{c\,\text{SNR}^{\frac{2(n-1)}{2n-3}}}{n-3/2}\right).$$

Using L'Hôpital's rule and because the derivative of $W(x)$ is $W(x)/[x(1+W(x))]$ (cf. [5]), it is straightforward to check that $W(x)/\log x$ converges to 1 for $x \to \infty$. For sufficiently large SNR, therefore, there exists a constant $c_1 > 0$ such that

$$\text{SNR}^{\epsilon} \geq c_1 \frac{n-3/2}{c}\left[\frac{2(n-1)}{2n-3}\log \text{SNR} - \log\left(\frac{n-3/2}{c}\right)\right],$$

and so $\text{SNR}^{\epsilon} \in \Omega(\log \text{SNR})$. Plugging this into the bound of Lemma 4.4 finally results in[2]

$$\mathbb{E}[(\hat{S} - S)^2] \in \Omega(\text{SNR}^{-n}(\log \text{SNR})^{n-1}),$$

and no choice of $\epsilon(\text{SNR})$ can improve this bound. □

4.3 Asymptotic Achievability of Lower Bound

The previous section showed that the hybrid transmission strategy achieves an SDR scaling of at best $\text{SNR}^n/(\log \text{SNR})^{n-1}$. Using a simple decoder this scaling is in fact achievable, as will now be shown.

4.3.1 A Suboptimal Decoder

The X_i are transmitted across the channel, producing at the channel output the symbols

$$Y_i = X_i + Z_i, \quad i = 1, \ldots, n,$$

where the Z_i are iid Gaussian random variables of variance ${\sigma_Z}^2$. To estimate S from $Y_1, \ldots, Y_n$, the decoder first computes separate estimates $\hat{Q}_1, \ldots, \hat{Q}_{n-1}$ and $\hat{E}_{n-1}$, and then combines them to obtain the final estimate $\hat{S}$. While this strategy is suboptimal in terms of achieving a small MSE, it will turn out to be good enough to achieve the desired SDR scaling.

[2] If $a(x) \in \Omega(f(x))$ and $b(x) \in \Omega(g(x))$, then $a(x)b(x)^m \in \Omega(f(x)g(x)^m)$.

The Q_i are estimated using a maximum likelihood (ML) decoder, which yields the minimum distance estimate

$$\hat{Q}_i = \frac{1}{\beta} \arg\min_{j \in \mathbb{Z}} \left| \frac{j\sqrt{P}}{\gamma\beta} - Y_i \right|. \tag{4.10}$$

To estimate E_{n-1}, a linear minimum mean-square error (LMMSE) estimator is used (see e.g. [31, Section 8.3]), which computes

$$\hat{E}_{n-1} = \frac{\mathbb{E}[E_{n-1}Y_n]}{\mathbb{E}[Y_n^2]} Y_n. \tag{4.11}$$

Finally, using (4.2), the estimate of S is computed as

$$\hat{S} = \sum_{i=1}^{n-1} \frac{1}{\beta^{i-1}} \hat{Q}_i + \frac{1}{\beta^{n-1}} \hat{E}_{n-1}. \tag{4.12}$$

4.3.2 Error Analysis

The overall MSE $\mathbb{E}[(\hat{S} - S)^2]$ can be broken up into contributions due to the errors in decoding Q_i and E_{n-1} as follows. From (4.2) and (4.12), the difference between $\hat{S}$ and S is

$$\hat{S} - S = \sum_{i=1}^{n-1} \frac{1}{\beta^{i-1}} (\hat{Q}_i - Q_i) + \frac{1}{\beta^{n-1}} (\hat{E}_{n-1} - E_{n-1}).$$

The error terms $\hat{Q}_i - Q_i$ depend only on the noise of the respective channel uses and are therefore independent of each other and of $\hat{E}_{n-1} - E_{n-1}$, so the error variance can be written componentwise as

$$\mathbb{E}[(\hat{S} - S)^2] = \sum_{i=1}^{n-1} \frac{1}{\beta^{2(i-1)}} \mathcal{E}_{Q,i} + \frac{1}{\beta^{2(n-1)}} \mathcal{E}_E, \tag{4.13}$$

where $\mathcal{E}_{Q,i} \overset{\text{def}}{=} \mathbb{E}[(\hat{Q}_i - Q_i)^2]$ and $\mathcal{E}_E \overset{\text{def}}{=} \mathbb{E}[(\hat{E}_{n-1} - E_{n-1})^2]$.

Lemma 4.8. *For each $i = 1, \ldots, n-1$,*

$$\mathcal{E}_{Q,i} \in O\left(\exp\{-c\,\mathrm{SNR}\,/\beta^2\}\right), \tag{4.14}$$

where $c > 0$ is a constant. (The O-notation is defined in Appendix 3.A.)

Proof. Define the interval

$$\mathcal{I}_j = \left[\frac{(j - \frac{1}{2})\sqrt{P}}{\gamma\beta}, \frac{(j + \frac{1}{2})\sqrt{P}}{\gamma\beta} \right).$$

According to the minimum distance decoder (4.10), $\hat{Q}_i - Q_i = j/\beta$ whenever $Z_i \in \mathcal{I}_j$. The error $\mathcal{E}_{Q,i}$ satisfies thus

$$\begin{aligned}\mathbb{E}[(\hat{Q}_i - Q_i)^2] &= \frac{1}{\beta^2}\sum_{j\in\mathbb{Z}} j^2 \Pr[Z_i \in \mathcal{I}_j] \\ &= \frac{2}{\beta^2}\sum_{j=1}^{\infty} j^2 \Pr[Z_i \in \mathcal{I}_j], \end{aligned} \tag{4.15}$$

where the second equality follows from the symmetry of the distribution of Z_i. Now,

$$\Pr[Z_i \in \mathcal{I}_j] = Q\left(\frac{(j-\frac{1}{2})\sqrt{\text{SNR}}}{\gamma\beta}\right) - Q\left(\frac{(j+\frac{1}{2})\sqrt{\text{SNR}}}{\gamma\beta}\right),$$

where

$$Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^{\infty} e^{-\xi^2/2} d\xi,$$

which can be bounded from above for $x \geq 0$ as

$$Q(x) \leq \frac{1}{2} e^{-x^2/2}.$$

Since $\beta \geq 1$, (4.15) is then upper bounded by

$$\mathcal{E}_{Q,i} \leq \sum_{j=1}^{\infty} j^2 \exp\left\{-\frac{(j-1/2)^2\,\text{SNR}}{2\gamma^2\beta^2}\right\}.$$

Note that for $j \geq 2$, $(j-1/2)^2 > j$. Thus

$$\begin{aligned}\mathcal{E}_{Q,i} \;&\leq\; \exp\left\{-\frac{\text{SNR}}{8\gamma^2\beta^2}\right\} \\ &\quad + \sum_{j=2}^{\infty} j^2 \exp\left\{-\frac{j\,\text{SNR}}{2\gamma^2\beta^2}\right\}. \end{aligned} \tag{4.16}$$

To bound the infinite sum, use

$$\sum_{j=2}^{\infty} j^2 p^j \leq \sum_{j=1}^{\infty} j^2 p^j = \frac{p^2+p}{(1-p)^3} \tag{4.17}$$

with $p = \exp\{-\,\text{SNR}\,/2\gamma^2\beta^2\}$. The first term of (4.16) thus dominates for large values of $\text{SNR}\,/\beta^2$ and

$$\mathcal{E}_{Q,i} \leq c_1 \exp\left\{-\frac{\text{SNR}}{c_2\beta^2}\right\}$$

for some $c_1 > 0$ and $c_2 = 8\gamma^2$, which completes the proof. □

Lemma 4.9. $\mathcal{E}_E \in O(\text{SNR}^{-1})$.

Proof. The mean-squared error that results from the LMMSE estimation (4.11) is

$$\mathcal{E}_E = {\sigma_E}^2 - \frac{(\mathbb{E}[E_{n-1}Y_n])^2}{\mathbb{E}[Y_n^2]}. \tag{4.18}$$

Since

$$Y_n = X_n + Z_n = \frac{\sqrt{P}}{\gamma} E_{n-1} + Z_n,$$

$\mathbb{E}[E_{n-1}Y_n] = \sqrt{P}{\sigma_E}^2/\gamma$. Moreover, $\mathbb{E}[Y_n^2] = \mathbb{E}[X^2] + \mathbb{E}[Z^2] = P{\sigma_E}^2/\gamma^2 + {\sigma_Z}^2$. Inserting this into (4.18) yields

$$\begin{aligned}
\mathcal{E}_E &= {\sigma_E}^2 - \frac{P{\sigma_E}^4/\gamma^2}{P{\sigma_E}^2/\gamma^2 + {\sigma_Z}^2} \\
&= {\sigma_E}^2 \left(1 - \frac{P{\sigma_E}^2/\gamma^2}{P{\sigma_E}^2/\gamma^2 + {\sigma_Z}^2}\right) \\
&= \frac{{\sigma_E}^2}{1 + \mathrm{SNR}\,{\sigma_E}^2/\gamma^2} \\
&< \frac{\gamma^2}{\mathrm{SNR}}.
\end{aligned}$$

Since γ is independent of the SNR (cf. Proposition 4.2), $\mathcal{E}_E \in O(\mathrm{SNR}^{-1})$ as claimed. □

4.3.3 Optimizing the Quantization Resolution

Recall the formula for the overall error

$$\mathbb{E}[(S - \hat{S})^2] = \sum_{i=1}^{n-1} \frac{1}{\beta^{2(i-1)}} \mathcal{E}_{Q,i} + \frac{1}{\beta^{2(n-1)}} \mathcal{E}_E.$$

According to Lemma 4.8, $\mathcal{E}_{Q,i}$ decreases exponentially when SNR/β^2 goes to infinity. This happens for increasing SNR if β is set e.g. to

$$\beta = \lceil \mathrm{SNR}^{(1-\epsilon)/2} \rceil$$

for some $\epsilon > 0$, in which case $\mathcal{E}_{Q,i} \in O\left(\exp(-c\,\mathrm{SNR}^{\epsilon})\right)$. From this and Lemma 4.9, the overall error satisfies

$$\mathbb{E}[(S - \hat{S})^2] \in O(\mathrm{SNR}^{-(n-\epsilon')}), \tag{4.19}$$

where $\epsilon' = (n-1)\epsilon$ can be made as small as desired.

As already mentioned in Remark 4.4, the choice of ϵ represents a tradeoff: for small ϵ the error due to the "discrete" part vanishes only slowly, but the scaling exponent in the limit is larger. For larger ϵ, $\mathcal{E}_Q$ vanishes quickly but the resulting exponent is smaller. This is illustrated by the simulation results in Figure 4.5. The remainder of this section shows how to choose ϵ as a function of SNR in order to achieve the SDR scaling upper bound of Theorem 4.7.

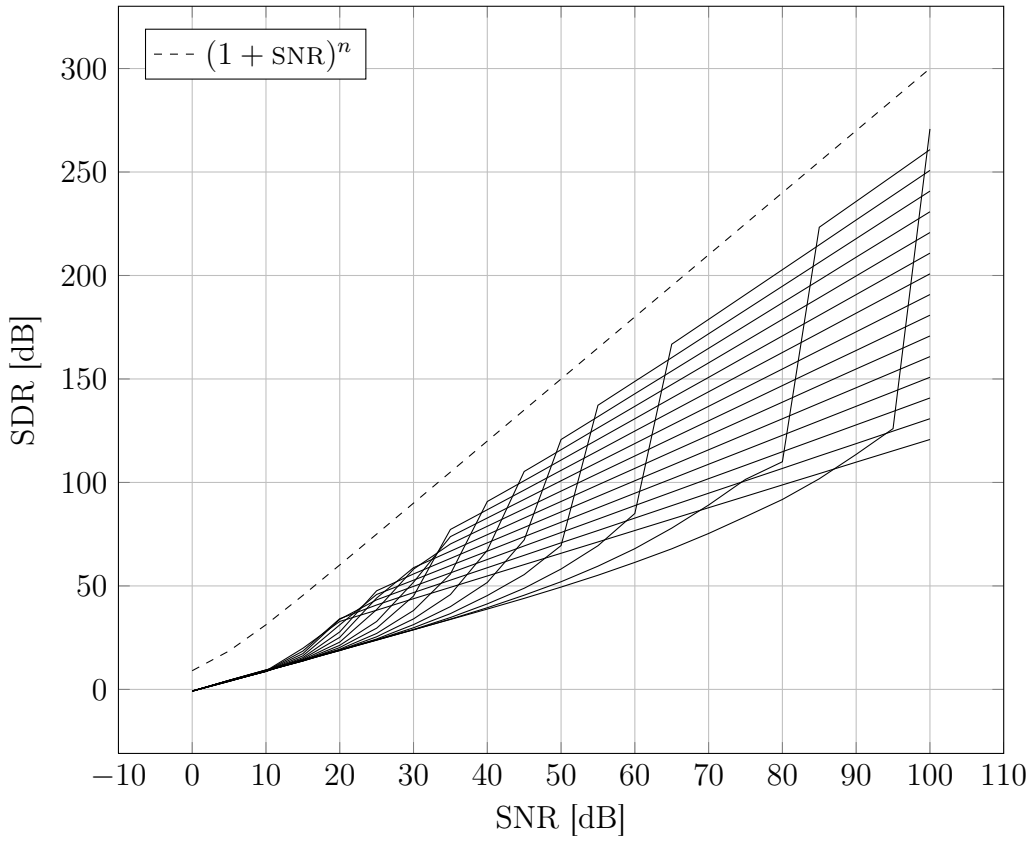


Figure 4.5: Simulation results illustrating the tradeoff represented by the choice of ϵ for $n = 3$. For larger ϵ, the error from decoding the discrete signal part decays quickly but the final scaling is worse, while for smaller ϵ the opposite holds. If ϵ is chosen optimally as a function of SNR, the resulting performance is the convex hull of the collection of all curves.

Let

$$\epsilon = \epsilon(\mathrm{SNR}) = \frac{\log(n \log \mathrm{SNR}\,/c)}{\log \mathrm{SNR}}, \tag{4.20}$$

where c is the constant indicating the decay of $\mathcal{E}_{Q,i}$ in (4.14). With this choice of ϵ,

$$\begin{aligned} \mathcal{E}_{Q,i} &\in O\left(\exp\left(-c\,\mathrm{SNR}^{\epsilon}\right)\right) \\ &= O(\mathrm{SNR}^{-n}), \end{aligned}$$

hence the overall error is still dominated as in (4.19). Inserting (4.20) in (4.19) leads to the following achievability result, which coincides with the converse result of Theorem 4.7, asserting that separately decoding the Q_i and E_{n-1} is asymptotically optimal.

Theorem 4.10. *Setting* $\beta = \lceil \mathrm{SNR}^{(1-\epsilon)/2} \rceil$ *with* ϵ *as in* (4.20)*, the decoder described by* (4.10)–(4.12) *achieves a mean squared error that scales as*

$$\mathbb{E}[(\hat{S} - S)^2] \in O(\mathrm{SNR}^{-n}(\log \mathrm{SNR})^{n-1}).$$

4.4 Encoding Blocks of Source Symbols using Lattices

The scalar quantizer scheme described in the previous section can be extended quite easily to treat blocks of m source symbols using lattices for the hierarchical quantization. For the reader unfamiliar with lattices or in need of a refresher, a concise presentation of the necessary concepts and results is provided in Appendix 4.C.

To anticipate the conclusion of this section, it turns out that the SDR scaling achieved for $m = 1$ cannot be improved upon by choosing a larger m. Choosing a larger m and thus a lattice of larger dimension can however result in faster convergence to the asymptotic scaling and thus increase the SDR for low SNR values (see Figure 4.6 for a preview).

4.4.1 Transmission Strategy

The procedure to encode a vector $\mathbf{S}$ of m source symbols into mn channel input vectors $\mathbf{X}_1, \ldots, \mathbf{X}_n$ using an m-dimensional lattice for quantization is analog to that in Section 4.1, except that now all involved quantities are m-dimensional vectors.

Let Λ be some fixed lattice of dimension m and define $\mathbf{E}_0 = \mathbf{S}$. For $i = 1, \ldots, n-1$ define

$$\begin{aligned} \mathbf{Q}_i &= Q_{\Lambda/\beta}(\mathbf{E}_{i-1}) \\ \mathbf{E}_i &= \beta(\mathbf{E}_{i-1} - \mathbf{Q}_i), \end{aligned} \tag{4.21}$$

where $\beta \in \mathbb{N}$ and where $Q_{\Lambda/\beta}(\cdot)$ denotes quantization with respect to the lattice Λ/β. According to Lemma 4.18 (in the appendix), $\mathbb{E}[\|\mathbf{Q}_i\|^2]$ is upper bounded by a constant independent of β. Moreover, $\mathbf{E}_i$ is contained within the Voronoi region of Λ around the origin, so by Lemma 4.19 $\mathbb{E}[\|\mathbf{E}_i\|^2]$ is also upper bounded independent of β. Let γ denote the common upper bound on $\mathbb{E}[\|\mathbf{Q}_i\|^2]/m$ and $\mathbb{E}[\|\mathbf{E}_i\|^2]/m$. The channel inputs are then computed as

$$\begin{aligned} \mathbf{X}_i &= \frac{\sqrt{P}}{\gamma}\mathbf{Q}_i \quad \text{for } i = 1, \ldots, n-1 \text{ and} \\ \mathbf{X}_n &= \frac{\sqrt{P}}{\gamma}\mathbf{E}_{n-1}. \end{aligned}$$

By the above argument, this ensures that $\mathbb{E}[\|\mathbf{X}_i\|^2]/m \leq P$ for all i.

4.4.2 Error Lower Bound

Ziv's lower bound on the mean squared error introduced in Section 4.2 allows a straightforward extension to vector sources. With this, essentially the same argument sequence as in Section 4.2 can be used to lower bound the mean squared error, independent of the particular decoder used. (Again, the quantization resolution in terms of SNR is assumed to be $\beta = \lceil \mathrm{SNR}^{(1-\epsilon)/2} \rceil$ without loss of generality (cf. Remark 4.2).)

Lemma 4.11 (Extension of Lemma 4.3 to vector sources)**.** *Consider a communication system where a continuous-valued source vector* $\mathbf{S}$ *is encoded into a vector* $\mathbf{X}(\mathbf{S})$*, sent across independent parallel scalar AWGN channels with noise variance* ${\sigma_Z}^2$*, and decoded at the receiver to produce an estimate* $\hat{\mathbf{S}}$*. If the density* $f_{\mathbf{S}}(\mathbf{s})$ *of the source is such that there exists a set* Ξ *and a number* $p_{\min} > 0$ *such that* $f_{\mathbf{S}}(\mathbf{s}) \geq p_{\min}$ *whenever* $s \in \Xi$*, then for any vector* $\mathbf{\Delta}$ *the mean squared error incurred by the communication system satisfies*

$$\mathbb{E}[\|\hat{\mathbf{S}} - \mathbf{S}\|^2] \geq p_{\min} \left(\frac{\|\mathbf{\Delta}\|}{2} \right)^2 \int_{\Xi \cap (\Xi - \mathbf{\Delta})} Q(d(\mathbf{s}, \mathbf{\Delta})/2\sigma_Z) d\mathbf{s},$$

where $d(\mathbf{s}, \mathbf{\Delta}) = \|\mathbf{X}(\mathbf{s}) - \mathbf{X}(\mathbf{s} + \mathbf{\Delta})\|$*. (The addition of a set* A *and a vector* $\mathbf{x}$ *is defined as* $A + \mathbf{x} = \{\mathbf{a} + \mathbf{x} : \mathbf{a} \in A\}$*.)*

Remark 4.5. The set $\Xi \cap (\Xi - \mathbf{\Delta})$ is the equivalent of the interval $[A, B - \Delta]$ in the scalar case of Lemma 4.3. It has the property that for every $\mathbf{s} \in \Xi \cap (\Xi - \mathbf{\Delta})$, $\mathbf{s} + \mathbf{\Delta} \in \Xi$. To get a meaningful lower bound, $\mathbf{\Delta}$ should of course be chosen such that $\Xi \cap (\Xi - \mathbf{\Delta})$ is nonempty. Note also that when $\|\mathbf{\Delta}\| \to 0$, the volume (or area) of $\Xi \cap (\Xi - \mathbf{\Delta})$ converges to the volume of Ξ.

Proof of Lemma 4.11. The proof is essentially the same as that for the scalar case (Lemma 4.3). The only difference is that the integrals $\int_A^{B-\Delta}$ and $\int_{A+\Delta}^{B}$ are replaced, respectively, with $\int_{\Xi \cap (\Xi - \mathbf{\Delta})}$ and $\int_{\Xi \cap (\Xi + \mathbf{\Delta})}$. □

Using Lemma 4.11, Lemmas 4.4 and 4.5 can be rederived for the case of lattices; the statements are in fact identical to the scalar case.

Lemma 4.12. *For an arbitrary function* $\epsilon(\mathrm{SNR}) \geq 0$*, the mean squared error of the lattice communication scheme characterized by this function satisfies*

$$\mathbb{E}[\|\hat{\mathbf{S}} - \mathbf{S}\|^2] \in \Omega(\mathrm{SNR}^{-n+(n-1)\epsilon}).$$

Proof. See Appendix 4.D. □

Lemma 4.13. *For an arbitrary function* $\epsilon(\mathrm{SNR}) \geq 0$*, the mean squared error of the lattice communication scheme characterized by this function satisfies*

$$\mathbb{E}[\|\hat{\mathbf{S}} - \mathbf{S}\|^2] \in \Omega(\mathrm{SNR}^{-1+\epsilon/2} \exp\{-c\,\mathrm{SNR}^{\epsilon}\}),$$

where $c > 0$ *does not depend on* SNR*.*

Proof. See Appendix 4.D. □

Since Lemmas 4.12 and 4.13 are identical to Lemmas 4.4 and 4.5, it is a direct consequence that Theorem 4.7 also applies to lattice quantizers. It is restated here for completeness.

Theorem 4.14. *For any choice of the parameter β (as a function of the SNR) and for any decoder, the mean squared error of the lattice quantizer transmission strategy of Section 4.4 satisfies*

$$\mathbb{E}[\|\hat{\mathbf{S}} - \mathbf{S}\|^2] \in \Omega(\mathrm{SNR}^{-n}(\log \mathrm{SNR})^{n-1}).$$

Proof. The proof is identical to that of Theorem 4.7. □

4.4.3 Achievability

The MSE scaling lower bound of Theorem 4.14 is trivially achievable with the integer lattice $\mathbb{Z}$: quantization with this lattice can be performed independently in each dimension and is equivalent to repeated application of the scalar scheme of Section 4.1. Hence Theorem 4.10 applies to lattice quantizers as well.

Figure 4.6 compares SDR curves achieved with a scalar quantizer with those achieved using the 24-dimensional Leech lattice for quantization. While the scaling at high SNR is indeed the same, confirming the results of this section, the error due to decoding the discrete part of the signal decreases faster if the quantizing lattice has higher dimension. The following calculations can be used to quantify the performance.

Like in Section 4.3, a combination of ML decoder (for the lattice points) and LMMSE decoder (for the quantization error) may be used to estimate $\mathbf{S}$. Consider first the ML decoder. Since $\mathbf{Q}_i \in \Lambda/\beta$ and $\mathbf{X}_i = \sqrt{P/\gamma^2}\mathbf{Q}_i$, the ML decoder divides $\mathbf{Y}_i$ by $\sqrt{P/\gamma^2}$ and then sets $\hat{\mathbf{Q}}_i$ to be the closest point in Λ/β. The resulting error therefore satisfies $\hat{\mathbf{Q}}_i - \mathbf{Q}_i = \mathbf{p} \in \Lambda/\beta$ if $\sqrt{\gamma^2/P}\mathbf{Z}_i \in V(\mathbf{p})$, where $V(\mathbf{p})$ is the Voronoi region of $\mathbf{p}$ (with respect to Λ/β). Averaging over the noise, the average squared error is then

$$\mathbb{E}[\|\hat{\mathbf{Q}}_i - \mathbf{Q}_i\|^2] = \sum_{\mathbf{p}\in(\Lambda/\beta)\setminus\{\mathbf{0}\}} \|\mathbf{p}\|^2 \int_{V(\mathbf{p})} \xi(\mathbf{z})d\mathbf{z},$$

where $\xi(\mathbf{z})$ is the pdf of $\sqrt{\gamma^2/P}\mathbf{Z}_i$. Since $\mathbf{z} \in V(\mathbf{p})$ implies $\|\mathbf{p}\| \le \|\mathbf{z}\| + R/\beta$, where R/β is the covering radius of Λ/β, $\mathbb{E}[\|\hat{\mathbf{Q}}_i - \mathbf{Q}_i\|^2]$ can be upper bounded by

$$\begin{aligned}\mathbb{E}[\|\hat{\mathbf{Q}}_i - \mathbf{Q}_i\|^2] &\le \sum_{\mathbf{p}\in(\Lambda/\beta)\setminus\{\mathbf{0}\}} \int_{V(\mathbf{p})} (\|\mathbf{z}\| + R/\beta)^2 \xi(\mathbf{z})d\mathbf{z} \\ &\le \int_{\mathbb{R}^m\setminus B(0,\rho/\beta)} (\|\mathbf{z}\| + R/\beta)^2 \xi(\mathbf{z})d\mathbf{z},\end{aligned}$$

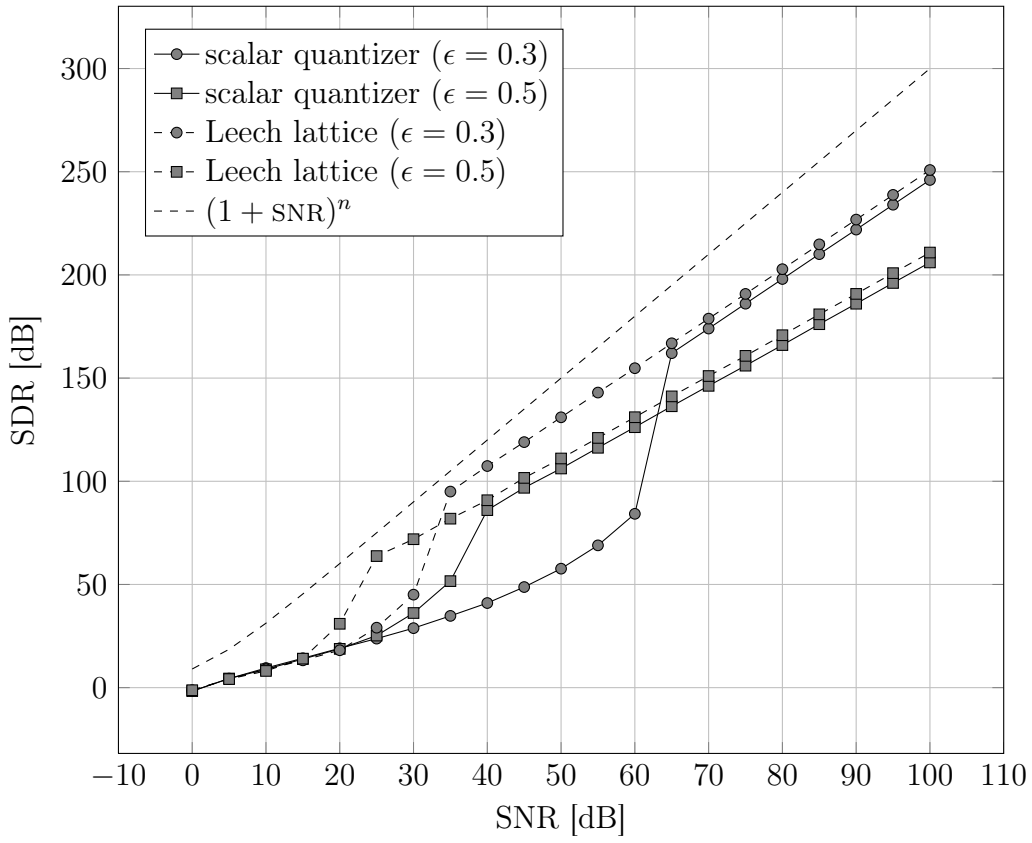


Figure 4.6: If the 24-dimensional Leech lattice is used for quantization, the scaling at large SNR remains the same but the MSE decreases quicker. This is useful for applications at low SNR. (Here $n = 3$.)

where $B(0, r)$ denotes a ball of radius r around the origin and ρ/β is the packing radius of Λ/β.

Transforming the above integral to spherical coordinates, one obtains $\|\mathbf{z}\| = r$ and $d\mathbf{z} = r^{m-1}S^{(m-1)}(1)dr$, where $S^{(m-1)}(1)$ is the surface of a unit sphere in $\mathbb{R}^m$. With this,

$$\mathbb{E}[\|\hat{\mathbf{Q}}_i - \mathbf{Q}_i\|^2] \leq S^{(m-1)}(1) \int_{\rho/\beta}^{\infty} (r + R/\beta)^2 r^{m-1} \phi(r) dr, \tag{4.22}$$

where $\phi(r)$ is the pdf of a real Gaussian random variable with variance $(\gamma^2/P)\sigma_Z{}^2 = \gamma^2/\,\text{SNR}$. To get explicit bounds for finite SNR that depend on R and ρ (and thus on the particular lattice used), this integral can be evaluated using the formula

$$\int_{\mu}^{\infty} r^n e^{-cr^2} dr = \frac{1}{2} c^{-(n+1)/2} \Gamma\left(\frac{1+n}{2}, c\mu^2\right), \tag{4.23}$$

where $\Gamma(s, x) = \int_x^{\infty} t^{s-1} e^{-t} dt$ is the upper incomplete gamma function.[3]

[3]This can be shown using the formula $\alpha_n(z) = z^{-(n+1)}\Gamma(n + 1, z)$, where $\alpha_n(z) =$

As $\text{SNR} \to \infty$, the bound (4.22) behaves as[4]

$$\begin{aligned}\mathbb{E}[\|\hat{\mathbf{Q}}_i - \mathbf{Q}_i\|^2] &\in O\left(\frac{1}{\beta^2}\int_{\rho/\beta}^{\infty}\phi(r)dr\right)\\ &= O\left(\frac{1}{\beta^2}Q\left(\frac{\sqrt{\text{SNR}}\rho}{\beta\gamma}\right)\right)\\ &= O\left(\frac{1}{\beta\sqrt{\text{SNR}}}\exp\left\{-\frac{\text{SNR}\,\rho^2}{2\beta^2\gamma^2}\right\}\right),\end{aligned}$$

where the last equality is because of the approximation

$$Q(x) \approx \exp\{-x^2/2\}/2\pi x$$

(see [1, §26.2.12]). Letting $\beta = \lceil \text{SNR}^{(1-\epsilon)/2} \rceil$ yields

$$\mathbb{E}[\|\hat{\mathbf{Q}}_i - \mathbf{Q}_i\|^2] \in O(\text{SNR}^{-1+\epsilon/2}\exp\{-c\,\text{SNR}^{\epsilon}\}),$$

which is the same scaling as the lower bound of Lemma 4.13.

As for the LMMSE decoder, it is a straightforward consequence of (4.21) that $\mathbb{E}[\|\hat{\mathbf{E}}_{n-1} - \mathbf{E}_{n-1}\|]/\beta^{2(n-1)}$ scales as $\text{SNR}^{-n+(n-1)\epsilon}$. Together, these two results again confirm that Theorem 4.7 applies to lattices as well.

4.5 General Bandwidth Expansion

If the source S has bounded support, the transmission strategy of Section 4.1 can easily be adapted to encode not one but k source symbols into n channel inputs (where $k < n$). For simplicity it is first assumed that the source support is contained in the interval $[-1/2, 1/2]$; a generalization follows at the end of this section.

4.5.1 Transmission Strategy

The strategy used to encode the source symbols $S_1, \ldots, S_k$ into the channel inputs $X_1, \ldots, X_n$ is displayed schematically on Figure 4.7 on the facing page for $k = 3$ and $n = 5$. The encoder consists of k parallel scalar encoders that encode each S_i into $n-k$ quantizer outputs $Q_{i,1}, \ldots, Q_{i,n-k}$ and a quantization error $E_{i,n-k}$, just like for the case $k = 1$. The quantizer outputs at each level are then combined into a single value $Q_{\text{tot},j}$ (cf. Figure 4.7). More precisely,

$\int_1^\infty t^n e^{-zt}dt$ [1, §5.1.5, §5.1.56] and using variable substitution to find that $\int_\mu^\infty r^m e^{-cr^2}dr = \frac{1}{2}\mu^{m+1}\alpha_{(m-1)/2}(c\mu^2)$.

[4] An exact derivation of this statement fell victim to the delay constraints of thesis writing. For now, the author puts his trust in Mathematica; a rigorous proof is left to the investigative reader (Chapter 5 of [1] may be of help).

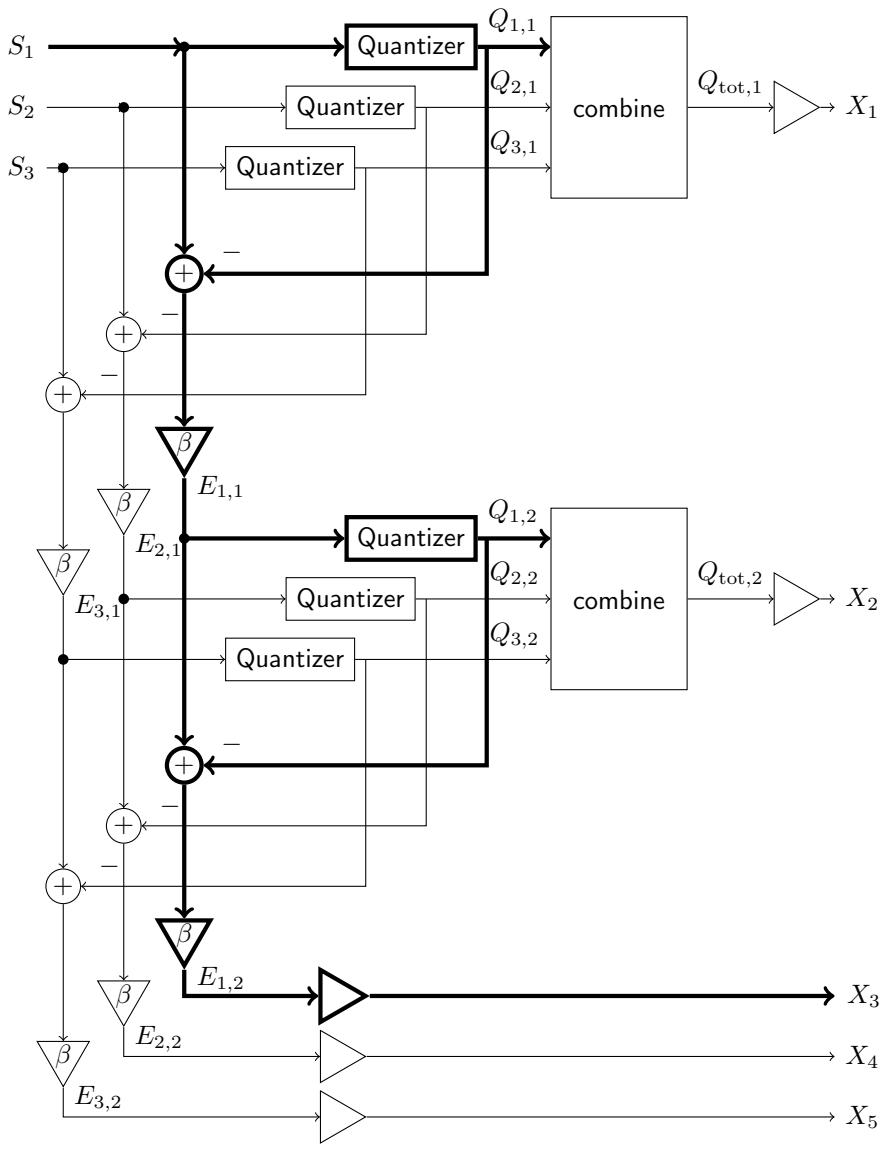


Figure 4.7: Schematic display of the transmission strategy of Section 4.5 for $k = 3$ and $n = 5$. The boxes labeled Q are uniform scalar quantizers with resolution β as described in (4.24). The boxes marked "combine" implement the operation (4.25). The part marked in thick lines corresponds exactly to the encoding scheme described in Section 4.1 to encode one source symbol into three channel uses; compare this with Figure 4.1.

letting $i \in \{1, \ldots, k\}$ be the index of the source symbol, the encoder defines $E_{i,0} = S_i$ for each i and computes

$$Q_{i,j} = \frac{1}{\beta} \operatorname{int}(\beta E_{i,j-1}) \quad \text{and}$$
$$E_{i,j} = \beta(E_{i,j-1} - Q_{i,j}) \tag{4.24}$$

for $j = 1, \ldots, n-k$. (This is exactly the same as (4.1), except for the addition of the subscript i.) For each quantization level j, the k parallel quantizer outputs are combined into

$$Q_{\text{tot},j} = \sum_{i=1}^{k} \frac{1}{\beta^{i-1}} Q_{i,j}. \tag{4.25}$$

Since each $Q_{i,j}$ is a multiple of $1/\beta$ and is assumed to lie in $[-1/2, 1/2]$, the mapping from the $Q_{i,j}$ into $Q_{\text{tot},j}$ is invertible, and the inverse can be recursively computed as

$$Q_{k,j} = \beta^{k-1} Q_{\text{tot},j} \bmod 1$$
$$Q_{k-i,j} = \beta^{k-1-i} Q_{\text{tot},j} - \sum_{l=1}^{i} \beta^{-l} Q_{k-i+l} \bmod 1, \quad i = 1, \ldots, k-1, \tag{4.26}$$

where $x \bmod 1 \stackrel{\text{def}}{=} x - \operatorname{int}(x)$.

Point 2 of Proposition 4.2 from Section 4.1 applies here as well, so there exists a constant γ^2 that upper bounds the variances of all $Q_{i,j}$ and $E_{i,j}$ for all β. The channel inputs are thus

$$X_j = (\sqrt{P}/\gamma) Q_{\text{tot},j} \qquad \text{for } j = 1, \ldots, n-k \text{ and}$$
$$X_j = (\sqrt{P}/\gamma) E_{j-n+k,n-k} \qquad \text{for } j = n-k+1, \ldots, n.$$

4.5.2 Decoder

Just like when $k = 1$, the decoder first computes separate estimates $\hat{Q}_{i,j}$ and $E_{i,n-k}$ for $i = 1, \ldots, k$ and then combines them into the estimates $\hat{S}_i$. The $\hat{Q}_{i,j}$ are obtained using a maximum likelihood (minimum distance) estimate of $Q_{\text{tot},j}$ and then by breaking it down according to (4.26). The estimate of $Q_{\text{tot},j}$ is

$$\hat{Q}_{\text{tot},j} = \frac{1}{\beta^k} \arg\min_{l \in \mathbb{Z}} \left| \frac{l\sqrt{P}}{\gamma \beta^k} - Y_j \right|,$$

and the quantization errors $E_{i,n-k}$ are estimated as

$$E_{i,n-k} = \frac{\mathbb{E}[E_{i,n-k} Y_{n-k+i}]}{\mathbb{E}[Y_{n-k+i}^2]}.$$

Using (4.2) and (4.26), the final estimate for $i = 1, \ldots, k$ is

$$\hat{S}_i = \sum_{j=1}^{n-k} \frac{1}{\beta^{j-1}} \hat{Q}_{i,j} + \frac{1}{\beta^{n-k}} \hat{E}_{i,n-k}.$$

4.5.3 Error Analysis

As in the case $k = 1$, the overall mean squared error can be written as

$$\mathbb{E}[(\hat{S} - S)^2] = \sum_{j=1}^{n-k} \frac{1}{\beta^{2(j-1)}} \mathcal{E}_{Q,i,j} + \frac{1}{\beta^{2(n-k)}} \mathcal{E}_{E,i}$$

where $\mathcal{E}_{Q,i,j} \stackrel{\text{def}}{=} \mathbb{E}[(\hat{Q}_{i,j} - Q_{i,j})^2]$ and $\mathcal{E}_{E,i} \stackrel{\text{def}}{=} \mathbb{E}[(\hat{E}_{i,n-k} - E_{i,n-k})^2]$. The behavior of the $\mathcal{E}_{E,i}$ is exactly the same as when $k = 1$; the following lemma is therefore given without proof.

Lemma 4.15. *The estimation error of the $E_{i,n-k}$ satisfies*

$$\mathcal{E}_{E,i} \in O(\mathrm{SNR}^{-1})$$

for all $i = 1, \ldots, k$.

The main difference to the case $k = 1$ concerns the behavior of $\mathcal{E}_{Q,i,j}$. Because k quantizer outputs are packed into a single channel input as described by (4.25), β^2 is raised to the exponent k in the following lemma (compare with Lemma 4.8).

Lemma 4.16. *For each $i = 1, \ldots, k$ and for each $j = 1, \ldots, n - k$,*

$$\mathcal{E}_{Q,i,j} \in O(\exp\{-c\,\mathrm{SNR}\,/\beta^{2k}\})$$

where $c > 0$ is a constant.

Proof. For any i and j, $|\hat{Q}_{i,j} - Q_{i,j}| \neq 0$ only if $|Z_j| \geq \sqrt{P}/2\gamma\beta^k$. Furthermore, since $\hat{Q}_{i,j}, Q_{i,j} \in [-1/2, 1/2)$, $|\hat{Q}_{i,j} - Q_{i,j}| \leq 1$. The error $\mathcal{E}_{Q,i,j}$ is therefore upper bounded by

$$\begin{aligned}
\mathbb{E}[(\hat{Q}_{i,j} - Q_{i,j})^2] &\leq \Pr\left[|Z_j| \geq \frac{\sqrt{P}}{2\gamma\beta^k}\right] \\
&= 2Q\left(\frac{\sqrt{\mathrm{SNR}}}{2\gamma\beta^k}\right) \\
&\leq \exp\{-c\,\mathrm{SNR}\,/\beta^{2k}\}
\end{aligned}$$

with $c = 1/8\gamma$. □

Let now $\beta = \lceil \mathrm{SNR}^{(1-\epsilon)/2k} \rceil$. Then $\beta^{2k} \in O(\mathrm{SNR}^{1-\epsilon})$, and the bound from Lemma 4.16 becomes

$$\mathcal{E}_{Q,i,j} \in O(\exp\{-c\,\mathrm{SNR}^{\epsilon}\}) \tag{4.27}$$

(where $c > 0$ is not necessarily the same constant as in Lemma 4.16). Moreover, using Lemma 4.15,

$$\frac{\mathcal{E}_{E,i}}{\beta^{2(n-k)}} \in O(\mathrm{SNR}^{\frac{n}{k} - \epsilon\frac{n-k}{k}}). \tag{4.28}$$

The final step is to choose ϵ as a function of SNR (again just like for $k = 1$). Let

$$\epsilon(\text{SNR}) = \frac{\log(n \log \text{SNR}\,/c)}{\log \text{SNR}}.$$

Inserting this in (4.27) and (4.28),

$$\mathcal{E}_{Q,i,j} \in O(\text{SNR}^{-n}) \quad \text{and}$$
$$\mathcal{E}_{E,i} \in O(\text{SNR}^{-n/k}(\log \text{SNR})^{(n-k)/k}).$$

The overall MSE scales thus as

$$\mathbb{E}[(\hat{S} - S)^2] \in O(\text{SNR}^{-n/k}(\log \text{SNR})^{(n-k)/k}).$$

4.5.4 Extension to General Sources

The assumption in Section 4.5 has so far been that the support of the source is limited to $[-1/2, 1/2]$. If a source S has support outside this interval but its support still lies within a bounded set, just define $S' = S/\alpha$, with $\alpha > 1$ such that $S' \in [-1/2, 1/2]$. Then use the described scheme to transmit S' and let $\hat{S} = \alpha\hat{S}'$. The incurred distortion is $\mathbb{E}[(S - \hat{S})^2] = \alpha^2\mathbb{E}[(S' - \hat{S}')^2]$; the SDR therefore still scales in the same way as when $S \in [-1/2, 1/2]$.

For sources with unbounded support, some form of compander must be used to bring them into a bounded interval; this problem is left as future work.

4.6 Towards a General SDR Upper Bound

None of the communication strategies studied in this chapter achieve the SDR scaling of $\text{SNR}^{n/k}$ that is achievable without a delay limit. Instead, in each case the scaling is divided by $(\log \text{SNR})^{(n-k)/k}$, i.e., there is a "penalty factor" of $(\log \text{SNR})^{1/k}$ for each of the $n - k$ channel inputs that carry quantized information about the source.

At first sight this may just be due to the particular nature of the constellations used here. Because a significant fraction of channel inputs is from discrete alphabets, a certain decrease from the theoretically optimal performance is likely to result. On the other hand, the arguments brought forward in Chapter 3 make the use of discrete channel inputs plausible for a good minimal-delay code. What is more, there seems to be no known minimal-delay code that has been shown to achieve a better scaling than the one found here. The obvious question to ask, then, is whether the upper bounds found in this chapter do not only apply to the type of schemes studied, but may be of a more general nature.

Lower Bounding the Mean Squared Error for More General Constellations

To evaluate the best achievable SDR for a given transmission strategy, one has to assume that the best possible decoder is used. The decoder that minimizes the mean squared error is the posterior mean decoder, which computes $\hat{S}^k = \mathbb{E}[S^k|Y^n]$. As mentioned earlier in this chapter, the error resulting from this decoder is in general hard to evaluate mathematically.

This chapter uses a trick by Ziv to lower bound the MSE by the error probability of binary decoding, regardless of the particular decoder used, thus avoiding the need to analyze the posterior mean decoder. However, Ziv's bound can only be applied to a signal constellation that has some regularity: there must be a Δ such that for any source value s (in an interval with strictly positive probability density), the source points s and $s + \Delta$ are mapped to two channel inputs whose Euclidean distance is upper bounded. For example, in the proof of Lemma 4.5 we found that when $\Delta = \beta^{-1}$, the distance between $X(s)$ and $X(s + \Delta)$ is always $\sqrt{P}/\gamma\beta$. If there does not exist such a Δ, then the MSE cannot be bounded as in Equation 4.29 in the proof of Ziv's lemma (see page 73).

This does not mean that less regular constellations necessarily perform better. It only means that Ziv's bound cannot be readily applied to constellations that do not have the same regularity properties. One direction for future work is thus to extend Ziv's result such that the MSE resulting from more general constellations can be upper bounded.

Properties of a Scheme With Optimal SDR Scaling

Assuming that there exists a minimal-delay code that achieves the optimal SDR scaling $\text{SNR}^{n/k}$, such a code should have good minimal distance properties in the following intuitive sense. In conventional channel coding, a code has good minimum distance properties if every codeword is not too close to another codeword. In joint source-channel coding with a mean squared error criterion for a Gaussian channel, not all constellation points must be far from each other. In fact, two constellation points may be close to each other provided that the corresponding points in the source are close (except possibly for a set of source points of vanishing probability). Conversely, the farther two points of the source are from each other, the farther away the corresponding constellation points should be. This ensures that decoding errors resulting in a large squared error occur less frequently than those resulting in a smaller squared error.

In order to find bounds on the performance of minimal-delay source-channel codes, one could thus try to characterize the "best" minimum distance behavior that an arbitrary map from $\mathbb{R}^k$ to $\mathbb{R}^n$ (under a power constraint on its image) can have.

Essentially, a map $f : \mathbb{R}^k \to \mathbb{R}^n$ is good for joint source-channel coding if its inverse f^{-1} is "almost surely" *continuous*. A map from $\mathbb{R}^n$ to $\mathbb{R}^k$ is continuous,

roughly speaking, if any two points close in $\mathbb{R}^n$ are mapped to points close in $\mathbb{R}^k$. Here we say "almost surely" continuous, meaning that there may be pairs of points in $\mathbb{R}^n$ that violate the continuity property, as long as a decoding error between these points occurs almost never.

The preceding reflections are admittedly quite vague. The problem is that as long as it is not clear how one can bound the MSE for an arbitrary map (cf. above), it is also not clear how to formally specify the properties that such a map should have. In any case, how to formulate an exact problem and come up with good answers is definitely a topic that should be investigated further.

Piecewise Continuous Maps from $\mathbb{R}$ to $\mathbb{R}^n$

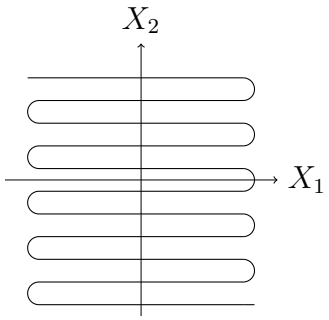
Of particular interest are maps that are (at least piecewise) continuous, since they can be implemented more easily than less structured maps. We can make a few general observations about such maps. First note that no loss of performance is incurred by using a constant stretch. The argumentation is similar to the "minimax considerations" by Wozencraft and Jacobs [47, p. 620]: using Ziv's bound, we can obtain a lower bound on the MSE by choosing the interval $[A, B]$ such that it contains the source section corresponding to the smallest stretch. It is therefore the *minimum* stretch that determines the MSE scaling, and by making the stretch constant (while preserving the shape of the constellation) this minimum can only be increased, leading to a better MSE scaling.

On the other hand, by a similar argument as that used in the proof of Lemma 4.4, if the stretch is constant and has value ℓ, then the SDR scales at most as $\ell^2 \,\mathrm{SNR}$. Thus, if the squared stretch is less than SNR^{n-1}, the optimal scaling cannot be achieved.

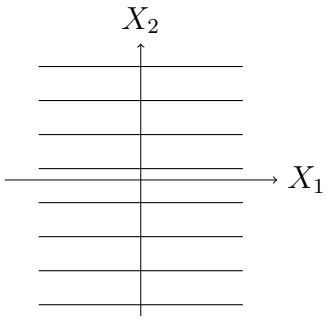
This implies that to achieve the optimal SDR scaling, a piecewise continuous map must have a constant squared stretch of at least SNR^{n-1}. For the example of a uniform source with support $[-1/2, 1/2]$, the problem of designing a good encoder then becomes the problem of how to arrange segments of a line of total length $\mathrm{SNR}^{(n-1)/2}$ in $\mathbb{R}^n$ such that the resulting signal locus has good minimum distance properties (as explained before) while simultaneously satisfying the power constraint. The maps presented in this chapter are one such way; they arrange the line as parallel segments. Another possibility is e.g. the Archimedes spiral [26].

Summary

Summarizing the above considerations, the problem of upper bounding the performance of minimal-delay codes can be broken down into two aspects. On one hand, it is about finding maps with good minimum distance properties, i.e., maps whose inverse is "almost surely" continuous (see above). On the other hand, a way must be found to lower bound the MSE for general maps. As we have seen, Ziv's bound only works if the constellation fulfills certain regularity



(a) Shannon's original proposition.



(b) The mapping proposed in Section 4.1 (for $n = 2$).

Figure 4.8: A minimum-delay source-channel code for $n = 2$ can be visualized as a curve in $\mathbb{R}^2$ parametrized by the source. Here the mapping presented in Section 4.1 is compared to Shannon's original suggestion (left).

properties. For more general bounds, either Ziv's bound must be extended or a wholly different bounding method must be found.

4.7 Historical Remarks

Schemes similar to the ones proposed here have been considered before. Indeed, one of the first schemes to transmit an analog source across two uses of a Gaussian channel was suggested by Shannon [34]. In fact, such joint source-channel mappings are sometimes called Shannon mappings or Shannon-Kotel'nikov mappings after Shannon and V. E. Kotel'nikov, who studied the problem independently of Shannon in his 1947 doctoral dissertation [23]. Notice the resemblance of the constellation resulting from the communication scheme of Section 4.1 to Shannon's original suggestion, both shown in Figure 4.8. The contribution of this thesis is to specify exactly how the distance between the segments must behave as the SNR increases in order to optimize the SDR scaling.

Wozencraft and Jacobs devoted a whole section of their 1965 textbook to the study of minimal-delay source-channel codes as curves in n-dimensional space [47, Section 8.2]. Sakrison's monograph [29] derived the optimal compander for sources with unbounded support (such as Gaussian sources) under the low noise assumption.

The particular communication strategy introduced in Section 4.1 was first explicitly mentioned by McRae in 1971 [24] as a modulation technique for bandspreading communication. Much of the more recent work on minimal-delay joint source-channel codes is due to Ramstad and his coauthors (see [26], [10], [9, 7], [45], [17]). For $n = 2$, the scheme of Section 4.1 is almost identical to the HSQLC scheme by Coward [8], which uses a numerically optimized quantizer, transmitter and receiver to minimize the mean-squared error (MSE) for finite

values of the SNR. Coward conjectured that the right strategy for $n > 2$ would be to repeatedly quantize the quantization error from the previous step, which is exactly what we do here.

Another closely related communication scheme is the *shift-map* scheme due to Chen and Wornell [3]. Vaishampayan and Costa [41] showed in their analysis that it achieves an SDR that scales as $\text{SNR}^{n-\epsilon}$ for any $\epsilon > 0$ if the relevant parameters are chosen correctly as a function of the SNR. Up to rotation and a different constellation shaping, the shift-map scheme is in fact virtually identical to the one presented here, a fact that was pointed out recently by Taherzadeh and Khandani [40]. In their own paper they develop a scheme that achieves almost the same SDR scaling the scheme presented here and is in addition robust to SNR estimation errors; their scheme, however, is based on rearranging the digits of the binary expansion of the source and requires greater implementation complexity.

Shamai, Verdú and Zamir [32] used Wyner-Ziv coding to extend an existing analog system with a digital code when additional bandwidth is available. Mittal and Phamdo [25] (see also the paper by Skoglund, Phamdo and Alajaji [38]) split up the source into a quantized part and a quantization error, much like we do here, but they use a separation-based code (or "tandem" code) to transmit the quantization symbols. Reznic et al. [28] use both quantization and Wyner-Ziv coding, and their scheme includes Shamai et al. and Mittal & Phamdo as extreme cases. All three schemes, however, use long block codes for the digital phase and incur correspondingly large delays, so they are not directly comparable with minimum delay schemes.

The bound used to lower bound the MSE scaling in Section 4.2 first occurred in a simpler form in a paper by Ziv and Zakai [49]. The version used here is based on the version that Ziv developped for his 1970 paper [48]. In that paper, Ziv found important theoretical limitations of source-channel mappings if the encoder can depend on the SNR only through a scaling factor.

4.A Proof of Lemma 4.1

Proof. Let $f(\xi)$ be the probability density function of E_{i-1}. Then

$$\begin{aligned}\mathbb{E}[Q_i^2] &= \frac{1}{\beta^2}\int_{\mathbb{R}} \operatorname{int}(\beta E_{i-1})^2 f(\xi)d\xi \\ &= \frac{1}{\beta^2}\sum_{i\in\mathbb{Z}} i^2 \int_{\frac{i-1/2}{\beta}}^{\frac{i+1/2}{\beta}} f(\xi)d\xi.\end{aligned}$$

Now, $|i| \leq \beta|\xi| + 1/2$ whenever $\xi \in [(i-1/2)/\beta, (i+1/2)/\beta)$, so

$$\begin{aligned}\mathbb{E}[Q_i^2] &\leq \frac{1}{\beta^2}\sum_{i\in\mathbb{Z}} \int_{\frac{i-1/2}{\beta}}^{\frac{i+1/2}{\beta}} (\beta|\xi| + 1/2)^2 f(\xi)d\xi \\ &= \frac{1}{\beta^2}\int_{\mathbb{R}} (\beta|\xi| + 1/2)^2 f(\xi)d\xi \\ &= \mathbb{E}[E_{i-1}^2] + \frac{1}{4\beta^2} + \frac{1}{\beta}\int_{\mathbb{R}} |\xi| f(\xi)d\xi.\end{aligned}$$

To bound the last integral, use the fact that $\mathbb{E}[|E_{i-1}|] \leq \sqrt{\mathbb{E}[E_{i-1}^2]}$ to finally obtain

$$\mathbb{E}[Q_i^2] \leq \mathbb{E}[E_{i-1}^2] + \frac{\sqrt{\mathbb{E}[E_{i-1}^2]}}{\beta} + \frac{1}{4\beta^2}.$$

□

4.B Proof of Ziv's Lower Bound (Lemma 4.3)

Conditioning the mean squared error on S and using the assumption on p_S one obtains

$$\mathbb{E}[(\hat{S} - S)^2] \geq p_{\min}\int_A^B \mathbb{E}[(\hat{S} - S)^2|s]ds.$$

For $\Delta \in [0, B - A]$ one can further bound this in two ways:

$$\begin{aligned}\mathbb{E}[(\hat{S} - S)^2] &\geq p_{\min}\int_A^{B-\Delta} \mathbb{E}[(\hat{S} - S)^2|s]ds \\ \mathbb{E}[(\hat{S} - S)^2] &\geq p_{\min}\int_{A+\Delta}^{B} \mathbb{E}[(\hat{S} - S)^2|s]ds \\ &= p_{\min}\int_A^{B-\Delta} \mathbb{E}[(\hat{S} - S)^2|s + \Delta]ds.\end{aligned}$$

Averaging the two lower bounds yields

$$\mathbb{E}[(\hat{S} - S)^2] \geq \frac{p_{\min}}{2}\int_A^{B-\Delta} \Big(\mathbb{E}[(\hat{S} - S)^2|s] + \mathbb{E}[(\hat{S} - S)^2|s + \Delta]\Big)ds, \quad (4.29)$$

and applying Markov's inequality to the expectation terms leads to

$$\mathbb{E}[(\hat{S}-S)^2|s] \geq \left(\frac{\Delta}{2}\right)^2 \Pr[|\hat{S}-S| \geq \Delta/2 \mid s] \tag{4.30}$$

and

$$\mathbb{E}[(\hat{S}-S)^2|s+\Delta] \geq \left(\frac{\Delta}{2}\right)^2 \Pr[|\hat{S}-S| \geq \Delta/2 \mid s+\Delta]. \tag{4.31}$$

Now suppose that the communication system in question is used for binary signaling. One wants to send either s or $s+\Delta$; at the decoder the estimate $\hat{S}$ is used to decide for s or $s+\Delta$ depending on which one $\hat{S}$ is closer to. When s is sent, the decoder makes an error only if $|\hat{S}-s| \geq \Delta/2$; when $s+\Delta$ is sent, it makes an error only if $|\hat{S}-s-\Delta| \geq \Delta/2$. The conditional error probabilities therefore satisfy $\Pr[\text{error}|s] \leq \Pr[|\hat{S}-S| \geq \Delta/2 \mid s]$ and $\Pr[\text{error}|s+\Delta] \leq \Pr[|\hat{S}-S-\Delta| \geq \Delta/2 \mid s+\Delta]$. Applying this to (4.30) and (4.31) and inserting the result in (4.29) yields

$$\mathbb{E}[(\hat{S}-S)^2] \geq p_{\min}\left(\frac{\Delta}{2}\right)^2 \int_A^{B-\Delta} P_e(s,\Delta)ds, \tag{4.32}$$

where $P_e(s,\Delta) = (\Pr[\text{error}|s] + \Pr[\text{error}|s+\Delta])/2$ is the average error probability.

If s and $s+\Delta$ are picked with equal probability and transmitted across n parallel Gaussian channels as $\mathbf{X}(s)$ and $\mathbf{X}(s+\Delta)$, and if $d(s,\Delta) = \|\mathbf{X}(s) - \mathbf{X}(s+\Delta)\|$, then the error probability of the MAP decoder is $Q(d(s,\Delta)/2\sigma_Z)$, a standard result of communication theory (see e.g. [47, Section 4.5]). Because the MAP decoder minimizes the error probability, $Q(d(s,\Delta)/2\sigma_Z) \leq P_e(s,\Delta)$, which, when inserted into (4.32), completes the proof. □

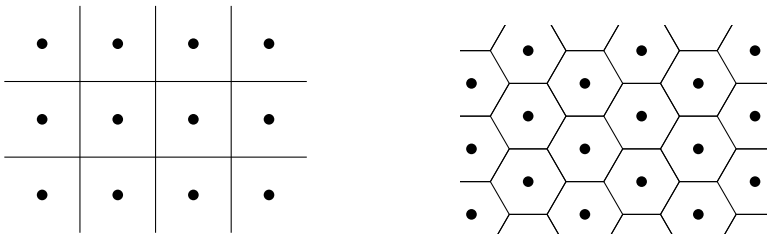
4.C Lattice Basics

This appendix contains the very basics on lattices and lattice quantization needed in Section 4.4. For a comprehensive treatment of lattices and/or quantization the reader is referred to the books by Conway and Sloane [4] and by Gersho and Gray [15].

Definition 4.1. *An n-dimensional* lattice Λ *is a discrete subgroup of $\mathbb{R}^n$ that spans $\mathbb{R}^n$. A sublattice of Λ is a subset $\Lambda' \subseteq \Lambda$ that is itself a lattice.*

Example 4.1. *In $\mathbb{R}$ there exists only a single lattice (up to scaling), the scalar lattice $\mathbb{Z}$. Two examples of lattices in $\mathbb{R}^2$ are displayed in Figure 4.9.*

Proposition 4.17. *If Λ is a lattice and $\beta \in \mathbb{N}$, then $\beta\Lambda$ is a sublattice of Λ. (The set $\beta\Lambda$ is defined as $\{\beta\mathbf{x} : \mathbf{x} \in \Lambda\}$.)*



(a) Rectangular lattice. (b) Hexagonal lattice.

Figure 4.9: Two lattices in $\mathbb{R}^2$ and the corresponding partition into Voronoi regions.

Proof. By definition of a lattice, $\beta\Lambda \subseteq \Lambda$. Moreover, if $\mathbf{x}, \mathbf{y} \in \beta\Lambda$, then $\mathbf{x} = \beta\mathbf{x}'$ and $\mathbf{y} = \beta\mathbf{y}'$ with $\mathbf{x}', \mathbf{y}' \in \Lambda$. It follows that $\mathbf{x} + \mathbf{y} = \beta(\mathbf{x}' + \mathbf{y}') \in \beta\Lambda$, so $\beta\Lambda$ is itself a lattice. □

Definition 4.2. *The* Voronoi region $V(\mathbf{p})$ *of a lattice point* $\mathbf{p} \in \Lambda$ *is defined as*

$$V(\mathbf{p}) = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{p}\| \leq \|\mathbf{x} - \mathbf{q}\|, \forall \mathbf{q} \in \Lambda\},$$

i.e., $V(\mathbf{p})$ *is the set of points in* $\mathbb{R}^n$ *that at least as close to* $\mathbf{p}$ *as to any other lattice point.*

See Figure 4.9 for an illustration of the Voronoi region.

Definition 4.3. *The* packing radius ρ *of a lattice is half the minimal distance between lattice points. Thus,* ρ *is the largest radius of spheres that can be packed in* $\mathbb{R}^n$ *by placing them at the lattice points.*

Definition 4.4. *The* covering radius R *of a lattice* Λ *is the least upper bound for the distance from any point of* $\mathbb{R}^n$ *to the closest point* $\mathbf{x} \in \Lambda$*. Thus, spheres of radius* ρ *around each lattice point will cover* $\mathbb{R}^n$*, and no smaller radius will do. [4]*

The packing radius and the covering radius are illustrated on Figure 4.10.

Definition 4.5. *A* lattice quantizer $Q_\Lambda : \mathbb{R}^n \to \Lambda$ *maps each point of* $\mathbb{R}^n$ *to the closest lattice point. Thus, for any* $\mathbf{x} \in \mathbb{R}^n$*,* $\mathbf{y} \in \Lambda$*,*

$$\|\mathbf{x} - Q_\Lambda(\mathbf{x})\| \leq \|\mathbf{x} - \mathbf{y}\|.$$

Remark 4.6. Definition 4.5 does not unambiguously specify $Q_\Lambda(\mathbf{x})$ if $\mathbf{x}$ lies on the boundary between the Voronoi regions of two adjacent lattice points. Since quantization is only applied to continuous-valued random variables in this chapter, however, the probability of this happening is zero, and this ambiguity can be left alone without causing any trouble.

Example 4.2. *Let* $\Lambda = \mathbb{Z}/\beta$ *for some* $\beta > 0$*. The associated quantizer* Q_Λ *maps each real number to the closest multiple of* $1/\beta$*. This is exactly the quantizer used in Section 4.1.*

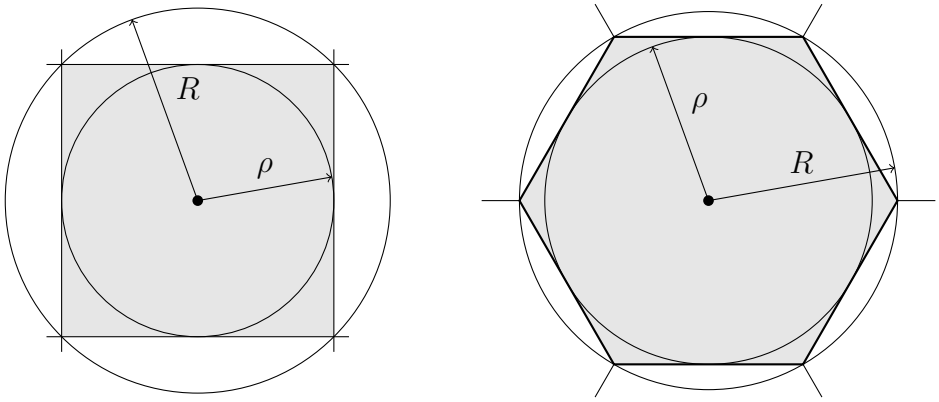


Figure 4.10: The packing radius ρ and the covering radius R for the rectangular lattice and the hexagonal lattice.

The next two lemmas are useful to bound the transmit power when transmitting a quantized random vector.

Lemma 4.18. *Let* $\mathbf{X}$ *be a random vector satisfying* $\mathbb{E}[\|\mathbf{X}\|^2] = \sigma^2 < \infty$. *Let* $\mathbf{Y} = Q_\Lambda(\mathbf{X})$. *Then* $\mathbb{E}[\|\mathbf{Y}\|^2] \leq \sigma^2 + 2R\sigma + R^2$, *where* R *is the covering radius of* Λ.

Proof. The power of $\mathbf{Y}$ is given by

$$\mathbb{E}[\|Q_\Lambda(\mathbf{X})\|^2] = \int_{R^n} \|Q_\Lambda(\mathbf{x})\|^2 f_{\mathbf{X}}(\mathbf{x})d\mathbf{x} = \sum_{\mathbf{p}\in\Lambda} \|\mathbf{p}\|^2 \int_{V(\mathbf{p})} f_{\mathbf{X}}(\mathbf{x})d\mathbf{x}.$$

By definition of the covering radius, $\|\mathbf{p}\| \leq \|\mathbf{x}\| + R$ for all $\mathbf{x} \in V(\mathbf{p})$. Thus,

$$\begin{aligned}\mathbb{E}[\|Q_\Lambda(\mathbf{X})\|^2] &\leq \sum_{\mathbf{p}\in\Lambda} \int_{V(\mathbf{p})} (\|\mathbf{x}\| + R)^2 f_{\mathbf{X}}(\mathbf{x})d\mathbf{x} \\ &= \int_{R^n} (\|\mathbf{x}\| + R)^2 f_{\mathbf{X}}(\mathbf{x})d\mathbf{x}.\end{aligned}$$

By assumption, $\int_{\mathbb{R}^n} \|\mathbf{x}\|^2 f_{\mathbf{X}}(\mathbf{x})d\mathbf{x} = \sigma^2$. Moreover, by the positivity of the variance, $\mathbb{E}[\xi] \leq (\mathbb{E}[\xi^2])^{1/2}$, and so $\int_{\mathbb{R}^n} \|\mathbf{x}\| f_{\mathbf{X}}(\mathbf{x})d\mathbf{x} \leq \sigma$. Applying this to the above yields

$$\mathbb{E}[\|Q_\Lambda(\mathbf{X})\|^2] \leq \sigma^2 + 2R\sigma + R^2,$$

thus completing the proof. □

Example 4.3. *Lemma 4.18 can be used to derive Lemma 4.1. Indeed, let* X *be a scalar zero-mean random variable of variance* σ^2 *and let* $\Lambda = \mathbb{Z}/\beta$, *for some* $\beta > 0$. *The covering radius of this lattice is* $R = 1/2\beta$, *so* $\mathbb{E}[Q_\Lambda(X)^2] \leq \sigma^2 + \beta^{-1}\sigma + \beta^{-2}/4$.

Lemma 4.19. *Let* $\mathbf{X}$ *be a random vector whose support is limited to the Voronoi region* $V(\mathbf{0})$ *of a lattice* Λ*. Then* $\mathbb{E}[\|\mathbf{X}\|^2] \leq R^2$*, where* R *is the covering radius of* Λ*.*

Proof.

$$\begin{aligned}\mathbb{E}[\|\mathbf{X}\|^2] &= \int_{V(\mathbf{0})} \|x\|^2 f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} \\ &\leq R^2 \int_{V(\mathbf{0})} f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} = R^2.\end{aligned}$$

□

4.D Proof of Lemmas 4.12 and 4.13

The following auxiliary result and its corollary will be useful for the proofs to come. See Figure 4.11 for an illustration.

Lemma 4.20. *Let* Λ *be an arbitrary* m*-dimensional lattice and let* $\Lambda' = \Lambda/\beta$*, where* $\beta \in \mathbb{N}$*. (By Proposition 4.17,* Λ *is a sublattice of* Λ'*.) Then the fraction of Voronoi cells of* Λ *that do not lie on the boundary between two Voronoi cells of* Λ' *is bounded away from zero as* β *grows large.*

Proof. Let $V(\Lambda)$ be a Voronoi region of Λ and let $V(\Lambda')$ be a Voronoi region of Λ'. Consider a sphere of radius $\rho - 2R'$ around the center of $V(\Lambda)$, where ρ is the packing radius of Λ and $R' = R/\beta$ is the covering radius of Λ'. By definition of the packing radius, this sphere is completely contained within $V(\Lambda)$. Furthermore, the distance from the border of the sphere to the border of $V(\Lambda)$ is at least $2R'$. Any Voronoi cell of Λ' that intersects with the boundary of $V(\Lambda)$ lies therefore outside the sphere.

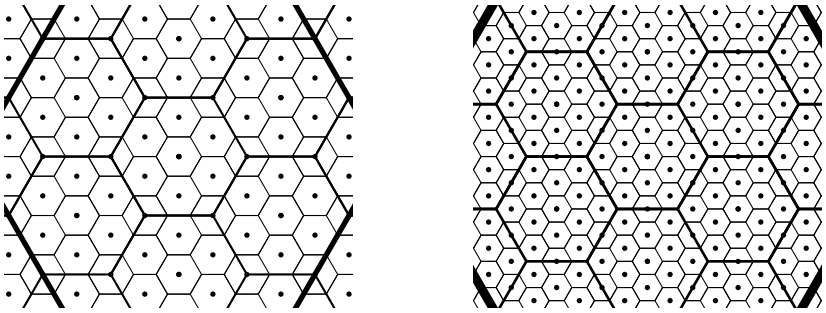
The fraction of Voronoi cells of Λ' that do not lie on the boundary of $V(\Lambda)$ is therefore lower bounded by

$$\frac{(\rho - 2R')^m V^{(m)}(1)}{\operatorname{Vol} V(\Lambda)} = \frac{(\rho - 2R/\beta)^n V^{(m)}(1)}{\operatorname{Vol} V(\Lambda)},$$

where $V^{(m)}(1)$ is the volume of the m-dimensional unit sphere. As beta grows large, this converges to $\rho V^{(n)}(1)/\operatorname{Vol} V(\Lambda) > 0$, and the proof is complete. □

Corollary. *Let* Λ *be a fixed lattice and consider the sequence of lattices* Λ/β*,* Λ/β^2*, ...,* Λ/β^{n-1}*, where* $\beta \in \mathbb{N}$*. Then the fraction of Voronoi regions of* Λ/β^{n-1} *that do not lie on the boundary between two Voronoi regions of any of its sublattices* Λ/β^i *is bounded away from zero as* β *grows large.*

Proof. Apply Lemma 4.20 successively to the pairs Λ/β^i, Λ/β^{i+1}, for $i = 1, \ldots, n-2$. □



(a) $\beta = 3$ (b) $\beta = 4$

Figure 4.11: Illustration for Lemma 4.20 and the corollary thereof. Note how in the right picture the fraction of small Voronoi cells that are not "cut" by a Voronoi boundary of the sublattice is larger.

Proof of Lemma 4.12. Consider the lattice $\Lambda_{n-1} \stackrel{\text{def}}{=} \Lambda/\beta^{n-1}$. For $\mathbf{p} \in \Lambda_{n-1}$, let $V(\mathbf{p})$ be the Voronoi region of $\mathbf{p}$. For some $\xi \in V(\mathbf{0})$ define $\mathbf{\Delta} = \Delta\xi$ for some $\Delta \leq 1$ and define $V_{\mathbf{\Delta}}(\mathbf{p}) = V(\mathbf{p}) \cap (V(\mathbf{p}) - \mathbf{\Delta})$. This set has the property that $\mathbf{x} + \mathbf{\Delta} \in V(\mathbf{p})$ whenever $\mathbf{x} \in V_{\mathbf{\Delta}}(\mathbf{p})$.[5]

If $\mathbf{p}$ is such that $V(\mathbf{p})$ does not lie on the boundary of the Voronoi region of a sublattice, then $V_{\mathbf{\Delta}}(\mathbf{p})$ defined this way has the property that if $\mathbf{s} \in V_{\mathbf{\Delta}}(\mathbf{p})$,

$$\begin{aligned} d(\mathbf{s}, \mathbf{\Delta}) &= \frac{\sqrt{mP}}{\gamma} \|\mathbf{E}_{n-1}(\mathbf{s}) - \mathbf{E}_{n-1}(\mathbf{s} + \mathbf{\Delta})\| \\ &= \frac{\sqrt{mP}}{\gamma} \|\xi\| \beta^{n-1} \Delta. \end{aligned}$$

In other words, the lattice quantizers at each level map $\mathbf{s}$ and $\mathbf{s} + \mathbf{\Delta}$ to the same lattice point.

One can now apply Lemma 4.11 and restrict the integral to the set

$$\Psi(\mathbf{\Delta}) \stackrel{\text{def}}{=} \Xi \cap (\Xi - \mathbf{\Delta}) \cap \bigcup_{\mathbf{p} \in {\Lambda_{n-1}}'} V_{\mathbf{\Delta}}(\mathbf{p}),$$

where ${\Lambda_{n-1}}'$ is defined to be the subset of those points in Λ_{n-1} whose Voronoi region does not lie on the boundary between Voronoi regions of a sublattice. By the corollary to Lemma 4.20 this set has positive probability.

The lower bound of Lemma 4.11 is then relaxed to give

$$\mathbb{E}[\|\hat{\mathbf{S}} - \mathbf{S}\|^2] \geq c_1 \Delta^2 Q(c_2 \sqrt{\text{SNR}} \beta^{n-1} \Delta) \int_{\Psi(\mathbf{\Delta})} d\mathbf{s}$$

for some positive constants c_1 and c_2. Letting $\Delta = 1/(\sqrt{\text{SNR}}\beta^{n-1})$ and $\beta^2 = \text{SNR}^{1-\epsilon}$ yields

$$\mathbb{E}[\|\hat{\mathbf{S}} - \mathbf{S}\|^2] \geq c_3 \, \text{SNR}^{-n+(n-1)\epsilon} \int_{\Psi(\mathbf{\Delta})} d\mathbf{s},$$

[5]For reference, the sets $V_{\mathbf{\Delta}}(\mathbf{p})$ are the equivalent of the intervals $\mathcal{I}_j^{\Delta}$ in the proof of Lemma 4.4.

with $c_3 > 0$ a constant independent of SNR.

It remains to prove the convergence of $\int_{\Psi(\mathbf{\Delta})} d\mathbf{s}$ to a constant. Since Δ goes to zero as SNR $\to \infty$, the set $\Psi(\mathbf{\Delta})$ converges to the set $\Xi \cap \bigcup_{\mathbf{p}\in\Lambda_{n-1}{}'} V(\mathbf{p})$. Since both Ξ and $\bigcup_{\mathbf{p}\in\Lambda_{n-1}{}'} V(\mathbf{p})$ have positive volume and because the points of $\Lambda_{n-1}{}'$ grow closer and closer as SNR and thus β increases, the overall set also has positive volume. □

Proof of Lemma 4.13. Let Λ be the normalized lattice used for quantization. Let $\xi \in \Lambda$ be arbitrary but fixed. Let $\mathbf{\Delta} = \beta^{-1}\xi$. Using the same reasoning as in the proof of Lemma 4.5, it follows that $\mathbf{Q}_1(\mathbf{s} + \mathbf{\Delta}) = \mathbf{Q}_1(\mathbf{s}) + \mathbf{\Delta}$, and also $\mathbf{Q}_i(\mathbf{s} + \mathbf{\Delta}) = \mathbf{Q}_i(\mathbf{s})$ for $i = 2, \ldots, n-1$, and $\mathbf{E}_{n-1}(\mathbf{s} + \mathbf{\Delta}) = \mathbf{E}_{n-1}(\mathbf{s})$. Consequently,

$$\|\mathbf{X}(\mathbf{s}) - \mathbf{X}(\mathbf{s} + \mathbf{\Delta})\| = \frac{\sqrt{mP}}{\gamma}\|\xi\|\beta^{-1}.$$

According to Lemma 4.11, therefore,

$$\mathbb{E}[\|\hat{\mathbf{S}} - \mathbf{S}\|^2] \geq c_1\beta^{-2}Q(c_2\sqrt{\text{SNR}}\beta^{-1}) \int_{\Xi\cap(\Xi-\mathbf{\Delta})} d\mathbf{s},$$

where c_1 and c_2 are positive constants independent of SNR. The rest of the proof is essentially identical to that of Lemma 4.5. □

JSCsim: A Joint Source–Channel Coding Simulator

5

For a communication engineer, simulations are an invaluable tool. They can not only help to verify the correctness of a theoretical result, they can also produce results about systems far too complex to be modeled mathematically.

Simulations are often seen as not more than ad-hoc tools. They are quick-and-dirty programming jobs that have served their purpose once the desired result has been obtained. According to the rule that the more a program is likely to change the better it should be structured, however, simulations should instead be among the most well-structured and thought through pieces of software. It takes time to plan and write well-structured code, but if the simulation is seen merely as a means to an end, communication engineers rarely take this time. As a result, most simulation code will eventually become so ugly that it is easier to write a new simulation from scratch rather than to modify the existing one.

This chapter presents JSCsim, an object-oriented simulator for joint source-channel coding schemes.[1] Its main design goal is to allow rapid testing of new ideas while simultaneously keeping the code free of redundancies.

JSCsim is implemented in MATLAB but could easily be translated to other languages that support object-oriented programming. Its power lies in its structure, which relies heavily on the twin paradigms of inheritance and polymorphism[2]. Moreover, this structure can be translated to communication problems other than joint source-channel coding.

This chapter is organized as follows. Section 5.2 makes the reader familiar with JSCsim by leading him through a series of hands-on examples and giving

[1] `http://ipg.epfl.ch/~kleiner/jscsim/`

[2] *Inheritance* is the capability of a class to "inherit" the properties and methods of another class while adding its own functionality. *Polymorphism* is the capability of a class to appear as and be used like another class.

```
function mse = run_simulation()
  snr = 10.^(0:.1:5);   % SNR range: 0 to 50 dB
  sv  = 1;              % Source variance
  N   = 100000;         % Sample size

  s = get_source_samples(N, sv);
  for k = 1:length(snr)
    x  = encode(s);
    y  = x + gaussian_noise(snr(k));
    sh = decode(y);

    mse(k) = mean((s - sh).^2);
  end
end
```

Listing 5.1: A hypothetical simulation of a joint source-channel communication scheme.

an overview of the implementation. Thereafter, Section 5.3 provides a detailed reference and usage manual.

Before introducing JSCsim, the chapter starts with a short example that illustrates why object oriented programming is so well suited for writing simulations.

Remark 5.1. The reader is assumed to be familiar with the fundamentals of object-oriented programming (classes, methods, encapsulation, etc.). For those unfamiliar with the subject, the Wikipedia article on object-oriented programming [46] provides a good introduction; otherwise there are numerous good language-specific introductions around. The examples throughout this chapter are in MATLAB's programming language.

5.1 Object Oriented Programming and Simulation: A Great Match

Imagine that you have written the MATLAB code in Listing 5.1 to simulate your freshly devised communication scheme. The function `run_simulation()` first defines a few relevant parameters and creates a vector of random source samples. Then, for each SNR value in the defined range, it calls `encode()` to encode the source symbols into the channel input `x`, adds noise, decodes the result `y` to produce an estimate `sh`. Finally it computes the empirical MSE and stores it in the vector `mse`.

Suppose now that you want to test an alternative decoding method. For example, you want to see how a maximum likelihood decoder compares to

an MMSE decoder. To this end you implement the alternative decoder in the function `alt_decode()`. How do you test this function? You have essentially four possibilities:

1. You replace the call to `decode()` in `run_simulation()` by a call to `alt_decode()`. The change to the existing code is only minimal. In doing so, however, you give up the old version of your code; this is not very good since you'll probably want to compare the performance of the new decoder to that of the old one and thus keep both versions around.

2. You copy the contents of `run_simulation()` into a new function called `run_alt_simulation()`, replacing the call to `decode()` with a call to `alt_decode()`. While this approach leaves the old code unchanged and again only requires little programming effort, it results in a lot of duplicate code, which makes your program error prone.

3. You do as in point 2, but then you eliminate duplicate code by putting all the common code into separate functions that will be called by both `run_simulation()` and `run_alt_simulation()`. This does indeed remove the redundancies, but it also requires significant programming effort. Moreover, if one day you decide to change the encoder as well, this will probably again require a similar amount of effort.

4. You add an argument to `run_simulation()` that specifies whether the old or the new decoding function should be used, and then you add a test as in

```
1 if decode_mode == 0
2   sh = decode(y);
3 elseif decode_mode == 1
4   sh = decode_alt(y);
5 end
```

   In the example given, `decode()` is called directly from `run_simulation()`, so the changes to the code are small (and no redundant code is created). However, in a practical simulator the decoding function may be under many layers of nested functions, and then all these functions would have to be modified to pass on the new parameter.

   A similar but more sophisticated way would be to set up a way for the simulator to read in configuration files and then add a configuration option for the decoding method used. This would require a (one-time) programming effort but it eliminates the need for options to be passed down from function to function.

   While this fourth method may be better than the first three, it still has the drawback that existing simulator code needs to be rewritten. After adding many such options and corresponding conditionals, the code may

well become difficult to read. Moreover, if someone else is the author of the original simulator and he or she releases a new version, all your changes will have to be merged into the new version.

As you can see, none of these options is quite satisfactory. You either lose old functionality, are left with a lot of duplicate code, or face a significant restructuring effort.

The Power of Inheritance

Suppose now that your programming language supports object-oriented programming (as MATLAB does[3]) and that you have implemented `run_simulation()`, along with the functions it calls, as *methods* of the class `Simulation`, as shown in Listing 5.2. Creating a new simulation with an alternative `decode()` function now couldn't be easier: just create a *derived class*[4] of `Simulation` and override the `decode()` method, as shown in Listing 5.3.

The two versions of the simulation can then be run as follows.

```
1 s = Simulation();                % Create class instance
2 mse1 = s.run_simulation();       % Call class method
3 as = AlternativeSimulation();    % Create class instance
4 mse2 = as.run_simulation();      % Call class method
```

Because the new class `AlternativeSimulation` inherits everything from `Simulation` except the overridden method `decode()`, it can be used just like the original simulation.

The object-oriented approach has none of the drawbacks of the previous example:

- The original simulation remains completely unchanged.
- There is no programming effort other than implementing the new method.
- There is no redundant code whatsoever.

The concept we have exploited here is called *inheritance*, since `AlternativeSimulation` *inherits* all those methods from `Simulation` which it doesn't explicitly redefine. The power of inheritance is that it allows *old code to call new code*: if `as` is an instance of `AlternativeSimulation` then a call to `as.run_simulation()` calls the "old" method `run_simulation()` (defined in `Simulation`), which in turn calls the *new* `decode()` method, without the need to make any changes to `run_simulation()`.

[3] at least the more recent releases

[4] In many object oriented programming languages, most notably Java, derived classes are called *subclasses* and base classes are called *superclasses*. As pointed out by Stroustrup [39, §12.2], this is somewhat confusing because the capabilities of a *sub*class form a *super*set of those of its superclass (and vice versa). Hence in this text we shall stick to the terms *base class* and *derived class*.

```
classdef Simulation
  methods
    function mse = run_simulation()
      ... % as before
    end

    function x = encode(s)
      ...
    end

    function sh = decode(y)
      ...
    end
  end
end
```

Listing 5.2: Object-oriented version of the simulator of Listing 5.1.

```
classdef AlternativeSimulation < Simulation
  methods
    function sh = decode(y)
      % Alternative decoder implementation goes here ...
    end
  end
end
```

Listing 5.3: A new simulation with an alternative decoder is easily implemented by deriving a new class from `Simulation` and overriding the `decode()` method.

In *procedural programming*[5], on the other hand, it is only possible for *new code to call old code*: a new function can call an existing one, but the opposite is not possible without changing the existing function. Furthermore, if you want two versions of an existing function to coexist such that one of them calls a new function (which is what we tried in our introductory example), this either requires code duplication (which leads to bugs) or significant programming effort.

To summarize: one can create two simulations that are identical except for a single function by the simple act of embedding the functions making up the simulation in a class and creating a derived class that overrides one of these functions.

[5]Procedural programming is programming using functions or *procedures*, as opposed to object-oriented programming.

5.2 A Short Overview of JSCsim

This section has two subsections. The first one is for those readers who just want to see how to *use* JSCsim and care less about what goes on behind the scenes. The second subsection, starting on page 94, presents JSCsim from the developer's point of view by explaining the architecture in a top-down fashion. Depending on the reader's interests, he or she may safely skip one or the other subsection.

The best way to read this section may be to read both parts in parallel. Start by going through the first example of the tutorial, then switch to reading about the implementation, and continue alternating between the two parts.

5.2.1 A Step-By-Step Tutorial

To demonstrate how JSCsim helps defining communication schemes and simulating them, this section presents a number of small hands-on examples.

Implementing a Communication Scheme

Example 3.1 of Chapter 1 showed that uncoded transmission is optimal to transmit a Gaussian source across a Gaussian channel if the source and channel bandwidths are matched. To verify this claim experimentally, let us implement uncoded transmission in JSCsim.

The code for this is shown in Listing 5.4 on the facing page. One can make the following observations.

1. The communication scheme is implemented as a *class* called `UncodedScheme` (line 1). It is derived from the class `PracticalScheme`. This is the class that all "practical" communication schemes are derived from, i.e., communication schemes that can be implemented in practice as opposed to merely "theoretical" schemes. The so called theoretical schemes, of which we will see an example shortly, compute a MSE from a mathematical formula (such as $\sigma_S^2/(1+\mathrm{SNR})^n$) without simulating transmission.

2. The constructor of `UncodedScheme` receives two parameters (line 4). `sv` is the source variance and `s` is the sequence of source symbols that are to be transmitted.[6]

   The constructor of the base class `PracticalScheme` has two additional arguments. They are, respectively, the number k of source symbols encoded at a time and the number n of channel inputs produced from every k source symbols. The scheme at hand is for $k = n = 1$, which is why the last two parameters passed to `PracticalScheme()` are both 1 (line 5).

[6] If you wonder why the constructor returns something called `obj`: this is simply how MATLAB's syntax specifies the constructor; the returned value is the object just created.

```
classdef UncodedScheme < PracticalScheme
    % UNCODEDSCHEME Implementation of uncoded transmission.
    %   This scheme implements uncoded transmission for the 1:1 case
    %   (no bandwidth expansion). The encoder scales the source
    %   symbol to satisfy the power constraint and the decoder is the
    %   LMMSE decoder.

    % $Id: UncodedScheme.m 812 2010-06-22 08:05:49Z kleiner $

    methods (Access = 'public')
        function obj = UncodedScheme(sv, s)
            obj@PracticalScheme(sv, s, 1, 1);
        end
    end

    methods (Access = 'protected')
        function x = encode(obj, s)
            x = sqrt(obj.P / obj.sv) * s;
        end

        function sh = decode(obj, y)
            sh = sqrt(obj.P * obj.sv) / (obj.P + obj.nv) * y;
        end
    end

end
```

Listing 5.4: Implementation of uncoded transmission.

3. Actual computation is only performed in two methods: `encode()` and `decode()`[7]. This makes sense, since a communication scheme is completely specified by its encoder and decoder. For the scheme at hand, these methods simply compute $X = \sqrt{P/\sigma_S^2}S$ and $\hat{S} = \sqrt{P\sigma_S^2}Y/(P + {\sigma_Z}^2)$.

Performance Analysis

Having implemented the class `UncodedScheme`, we would now like to plot its performance. For this, JSCsim has a class called `PerformanceProcessor`, which is used as follows.

[7]Again, the reason why the first argument to both `encode()` and `decode()` is `obj` has to do with MATLAB's syntax. All (non-static) methods of a class must have a first argument that refers to the object instance. Here we call it `obj` by convention, but it could in principle be given any other name. An object's properties are accessed from within its methods by prefixing them with '`obj.`' (cf. Listing 5.4). This is similar to the `this` pointer in C++ or Java, except that the latter doesn't have to be explicitly given as a method argument.

```
1 schemes = {'UncodedScheme'};
2 parameters = {[]};
3 pp = PerformanceProcessor();
4 pp.process(schemes, parameters);
```

The first two lines define two cell arrays[8] with a single entry each. The first contains the class names of the schemes to plot, and the second contains a list of parameters for each scheme. Here we only have a single scheme, `UncodedScheme`. It does not have any parameters, so the corresponding entry in the parameter list is an empty vector.

The third line creates an instance of the class `PerformanceProcessor`, whose `process()` method we invoke in the fourth line, passing the list of schemes and parameters. Figure 5.1 on the next page contains the resulting plot.

This was not much work at all. Behind the scenes, however, a lot more was going on:

1. A long sequence of random source symbols was generated.

2. For a range of SNR values and for each communication scheme, the source sequence was encoded using the `encode()` method of our new class, Gaussian noise of the appropriate variance was added, and the result was decoded using our new `decode()` method.

3. The average difference between the source sequence and the estimate sequence was computed, again for each scheme and for each value of SNR.

4. The resulting performance curves were plotted by the `PerformanceProcessor`.

All this work was done by `PerformanceProcessor` and the base class `PracticalScheme`, leaving us free to focus on the essential stuff. (Section 5.2.2 explains in detail how the above steps are performed.)

Theoretical Performance

At this point we have implemented uncoded communication, but we have not yet verified that it performs indeed optimally. From Chapter 1 we know that if there are n channel uses per source symbol then the optimal SDR is $(1+\mathrm{SNR})^n$. In JSCsim we can implement this as a "theoretical" communication scheme: this is a communication scheme that doesn't actually do any encoding or decoding, but simply computes a theoretical MSE for a given SNR.

[8] In MATLAB, *cell arrays* are special arrays whose elements can be of arbitrary types (scalars, vectors, matrices, strings, other cell arrays, etc.). Cell arrays are specified by listing their elements in curly braces, as in `{1, [2,3], 'foo'}`.

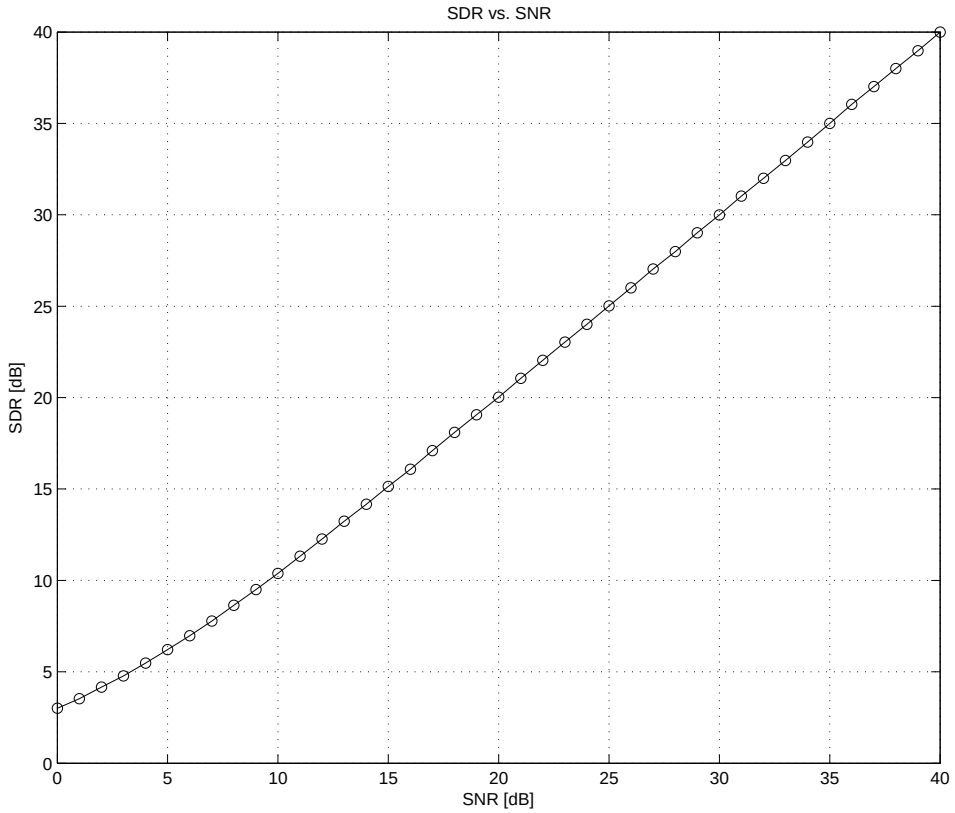


Figure 5.1: Plot of the SDR resulting from the `UncodedScheme` class, obtained using a `PerformanceProcessor`.

The resulting MATLAB code is given in Listing 5.5 on the following page. We can make the following observations.

1. The class `ShannonScheme` is derived from `TheoreticalScheme` rather than from `PracticalScheme` as in the previous example (line 1). This is because unlike practical schemes, theoretical schemes do not have `encode()` or `decode()` methods.

2. `ShannonScheme` has a single parameter `n`, the number of channel uses per source symbol, which it receives as an argument of its constructor (line 7).

3. The method `update_mse(obj)` (line 20) is the heart of this class. This method is called by the base class whenever the SNR is changed. Here it computes $\sigma_S^2/(1+\mathrm{SNR})^n$, which is the theoretically optimal MSE for the given SNR (cf. Chapter 1).

```
1  classdef ShannonScheme < TheoreticalScheme
2    properties
3      n         % The number of channel uses per source symbol.
4    end
5
6    methods (Access = 'public')
7      function obj = ShannonScheme(sv, s, n)
8
9        % Call base class constructor.
10       obj = obj@TheoreticalScheme(sv, s);
11
12       % Set class-specific parameters.
13       obj.n = n;
14     end
15   end
16
17   methods (Access = 'protected')
18     % This function is called whenever the SNR is changed and so
19     % the MSE needs to be recomputed.
20     function update_mse(obj)
21         obj.mse = obj.sv / (1 + obj.snr)^obj.n;
22     end
23   end
24 end
```

Listing 5.5: A "theoretical" communication scheme does not perform any actual encoding or decoding, but rather computes the theoretically optimal MSE for a given SNR.

4. `update_mse()` accesses the `snr` property (line 21), even though we never defined this property. This is because the property is set by the base class, so that all derived classes have access to the SNR.

To plot the performance of both `UncodedScheme` and `ShannonScheme` on the same plot, we can use the `PerformanceProcessor` as before:

```
1 schemes = {'UncodedScheme', 'ShannonScheme'};
2 parameters = {[], [1]};
3 pp = PerformanceProcessor();
4 pp.process(schemes, parameters);
```

The only difference to the previous example is that the `schemes` array now has two elements and that we need to specify the parameter $n = 1$ for `ShannonScheme`. This parameter will be passed to the constructor of `ShannonScheme` as its third argument.

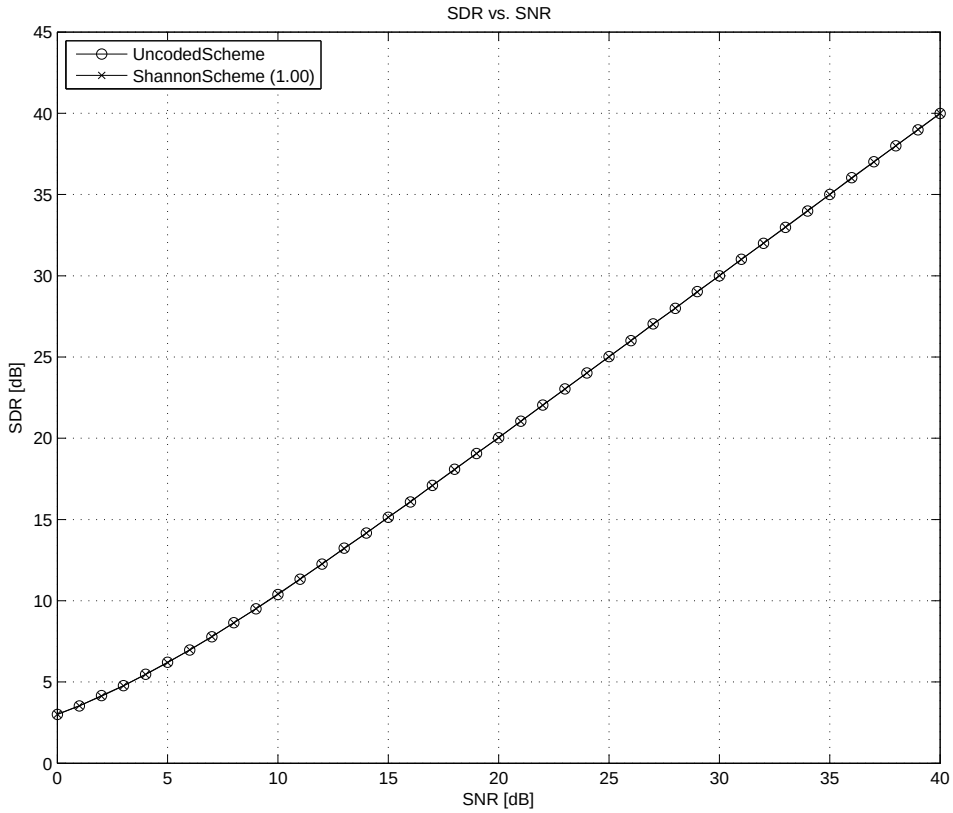


Figure 5.2: Comparing the theoretically optimal performance and the performance of uncoded transmission. The two curves coincide, experimentally confirming that uncoded transmission is optimal for the transmission of a Gaussian source across a Gaussian channel.

The resulting plot is shown on Figure 5.2 and confirms that uncoded transmission is indeed optimal. Note that a legend has automatically been added because we are plotting the performance of more than one communication scheme.

Alternative Output Formats

In the examples we saw so far, the performance processor just launched a standard MATLAB figure window with the performance plot. Often, you may not only want to look at the plot on the screen but also use it in a report or in a paper. For this, JSCsim has the concept of *output modules*. An output module is a class that implements a rudimentary set of plot capabilities.

The default output module is called `MatlabPlotModule` and uses MATLAB's plot command to display a figure window. Alternatively, to save the performance

plot in a PDF file, for instance, the default output module can be replaced by a `MatlabFilePlotModule`. Instead of displaying the plot in a window, this output module saves it in a file. Continuing our previous example, we can use it as follows.

```
1 ...  % Define list of schemes and parameters.
2 pp = PerformanceProcessor();
3 pp.output_module = MatlabFilePlotModule();
4 pp.output_module.fn = 'myplot.pdf';
5 pp.process(schemes, parameters);  % Saved to myplot.pdf.
```

In line 3 a new output module of type `MatlabFilePlotModule` is created and attached to the performance processor. Since this output module writes its output to a file, we have to specify a file name in line 4. Afterwards, we can call `process()` just as before.[9]

There is another output module, called `PGFPlotsOutputModule`. It produces a file containing LaTeX code to be used with the PGFPLOTS package[10]. To use it, just replace `MatlabFilePlotModule` in the above example by `PGFPlotsOutputModule` and set e.g.

```
1 pp.output_module.fn = 'myplot.tex';
```

The resulting file can then be included in a LaTeX file, provided that the `pgfplots` package has been loaded. The result is shown in Figure 5.3. Plots created by PGFPLOTS fit in nicer with the LaTeX layout, and their labels are more readable than those of Figures 5.1 and 5.2.

Analysis Using Scheme Processors

The performance processor we have used to plot the performance in the previous examples is a particular type of *scheme processor*. The idea behind scheme processors is to separate the implementation and the simulation of communication schemes.

A scheme processor takes a list of schemes along with the corresponding parameters, simulates these for a range of SNR values, and gathers data from each simulation run.

Each scheme processor is derived from the class `SchemeProcessor`. It must implement three methods:

- `initialize()` to allocate space for the data gathered;
- `save_scheme_data()` to collect data after each simulation run; and

[9] Unsurprisingly, Figures 5.1 and 5.2 have in fact been created using `MatlabFilePlotModule`.

[10] `http://pgfplots.sourceforge.net/`

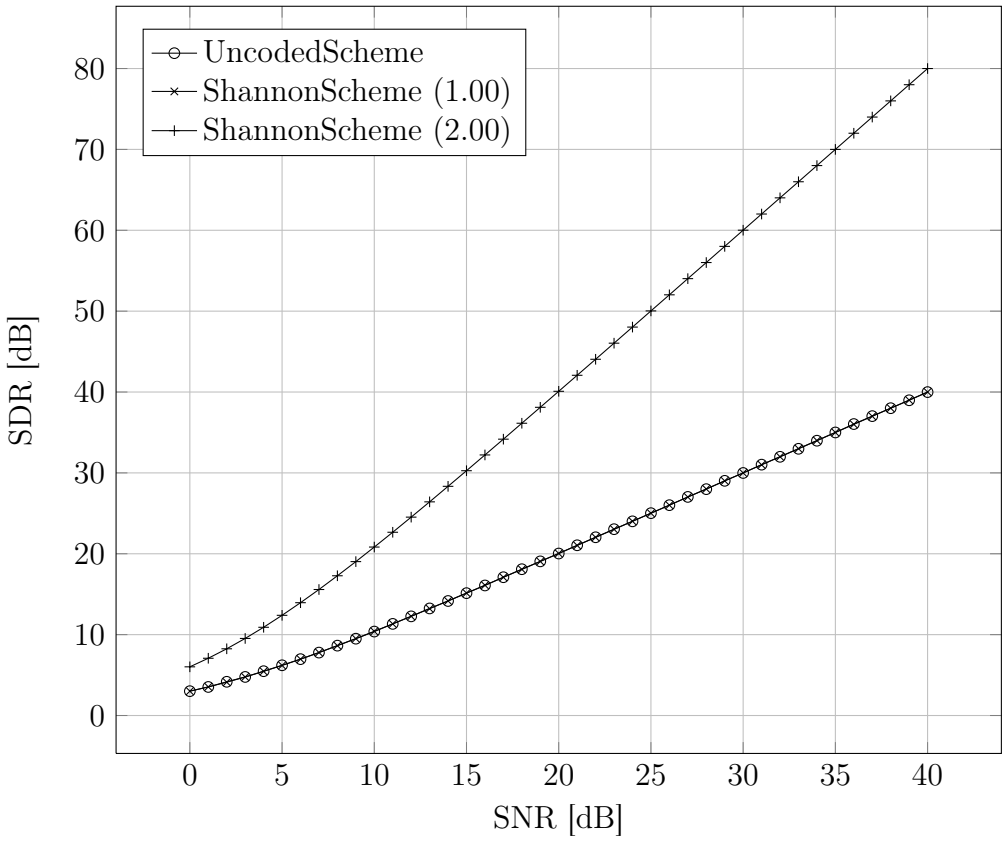


Figure 5.3: Using the `PGFPlotsOutputModule`, one can save the simulation output as TEX commands for the PGFPLOTS package, which can then be included in a LATEX source file.

- `post_process()` to post-process the gathered data, which usually just means to plot it.

This is illustrated by the simplified implementation of `PerformanceProcessor` in Listing 5.6 on the following page. This class has a single property, `mse` (line 4), where it stores the mean squared errors gathered in each simulation run. In its `initialize()` method, `mse` is initialized to a big all-zero matrix of the right size. In the method `save_scheme_data()`, the communication scheme just simulated (passed as the argument `scheme`) is queried by calling its `compute_mse()` method (line 13) and the result is stored in the matrix `mse`. Finally, `post_process()` plots the MSE of all processed schemes.

For another example of a performance processor, consider the hybrid communication scheme implemented as `Hybrid2DScheme` in Listing 5.7. In this communication scheme, each source symbol S is split into a discrete part Q and a continuous part E (lines 10–11), which are then transmitted across two channel uses (lines 13–14). The decoder computes the source estimate $\hat{S}$ from the separate estimates $\hat{Q}$ and $\hat{E}$ (lines 18–20). (This is similar to the

```
1  classdef PerformanceProcessor < SchemeProcessor
2
3      properties
4          mse
5      end
6
7      methods (Access = 'protected')
8          function initialize(obj)
9              obj.mse = zeros(nb_schemes(obj), length(obj.snr));
10         end
11
12         function save_scheme_data(obj, scheme, j, k)
13             obj.mse(j, k) = scheme.compute_mse();
14         end
15
16         function post_process(obj)
17           % Create the performance plot.
18           % ...
19         end
20     end
21 end
```

Listing 5.6: Simplified implementation of the `PerformanceProcessor` class.

communication schemes introduced in Chapter 4.)

A question of interest is the behavior of the average error $\mathbb{E}[(Q - \hat{Q})^2]$ as a function of the SNR. The scheme processor `Hybrid2DProcessor` in Listing 5.8 solves this task in a few small steps. It is very similar to the `PerformanceProcessor` of Listing 5.6, except that different variables are accessed in line 13. Since the corresponding communication scheme class, `Hybrid2DScheme`, saves `q` and `qh` as class properties and implements public `compute_*` methods for these properties, the `save_scheme_data()` method of the new scheme processor can easily access these and compute the corresponding error.

5.2.2 Implementation Overview

Design Philosophy

The design philosophy behind JSCsim is to separate the implementation of a communication strategy and the simulation thereof. The end user should be able to concentrate either on implementing the communication scheme or writing the code that processes it during simulation, but shouldn't have to think about both tasks at the same time. For this reason, the core classes that make up JSCsim are divided into two categories. In the first category are the *scheme classes* (or simply 'schemes'), which are classes that implement

```
classdef Hybrid2DScheme < PracticalScheme

    properties (Access = 'protected')
        q
        qh
    end

    methods (Access = 'protected')
        function x = encode(obj, s)
            obj.q = discrete_part(s);
            e = continuous_part(s);

            x(1, :) = scale_to_power_constraint(obj.q);
            x(2, :) = scale_to_power_constraint(e);
        end

        function sh = decode(obj, y)
            obj.qh = estimate_qh(y);
            eh = estimate_eh(y);
            sh = combine_estimates(obj.qh, eh);
        end

        % Other methods here ...
    end

    % Methods for scheme processors to access simulation data.
    methods (Access = 'public')
        function q = compute_q(obj)
            q = obj.q;
        end

        function qh = compute_qh(obj)
            qh = obj.qh;
        end
    end
end
```

Listing 5.7: Simplified implementation of a hybrid communication scheme. Note how `qh` and `eh` are saved as class properties, so that they can be accessed by a scheme processor through the respective `compute_*` methods.

```
classdef Hybrid2DProcessor < SchemeProcessor

    properties (Access = 'protected')
        qe
    end

    methods (Access = 'protected')
        function initialize(obj)
            obj.qe = zeros(obj.nb_schemes(), length(obj.snr));
        end

        function save_scheme_data(obj, scheme, j, k)
            obj.qe(j, k) = mean((scheme.compute_q() - ...
              scheme.compute_qh()).^2);
        end

        function post_process(obj)
            % Plot qe vs. SNR ...
        end
    end
end
```

Listing 5.8: A scheme processor to plot the estimation error of the discrete part in a hybrid communication scheme.

communication schemes and have the class `Scheme` as their common ancestor. Section 5.1 already covered why it makes sense to implement communication schemes as classes.

The second category consists of *scheme processors*, which are classes that implement the processing of a particular quantity (such as the MSE) during the simulation of a single or of a family of communication schemes. Scheme processors derive from the common ancestor class `SchemeProcessor`. What is the motivation behind organizing the processing code in this way? The "processing" of a communication scheme consists of two steps:

1. simulating the scheme by feeding the encoder with randomly generated source samples, transforming the encoder output by simulating a noisy channel (e.g., by adding Gaussian noise), and passing the channel output to the decoder

2. gathering relevant data, e.g., the mean squared error or other performance criteria, after the simulation has run, and displaying it

The base class `SchemeProcessor` essentially implements the first step. The second step is implemented in derived classes such as `PerformanceProcessor`

(which computes and plots the achieved SDR). This structure makes it possible to quickly implement new processor classes to analyze arbitrary quantities of a communication scheme or of a set of schemes, as the examples in the previous section show.

Based on the reasonable assumption that you implement a particular communication scheme in JSCsim because you eventually want to simulate it, you never directly instantiate a scheme class. Rather, you create a particular scheme processor (by instantiating a class derived – directly or indirectly – from `SchemeProcessor`) and call its `process()` method. Note that you cannot create an instance of the base class `SchemeProcessor` since the latter contains abstract methods[11] and serves only as a template for scheme processors. The example on page 87 shows how a performance processor, which is a particular type of scheme processor, is used to plot the SDR achieved by a communication scheme.

The interaction between scheme processors and scheme classes is illustrated in Figure 5.4 on the next page. When you first create an instance of a scheme processor, it generates and saves a random source sequence of a default variance.[12] Next you call the `process()` method of the scheme processor, passing a list of schemes (i.e., names of scheme classes) and a list of parameters for each scheme (cf. the example on page 87). (A scheme can have any number of parameters; for instance the number of channel input symbols produced per source symbol.) Internally, the scheme processor then creates an instance of each specified scheme, passing the source sample sequence to each one's constructor. Because all schemes operate on the same source sequence, a fair comparison is guaranteed.

A scheme processor has a list of SNR values for which each scheme is to be processed. By default, it runs from 0dB to 40dB with increments of 1dB. For each SNR value and for each scheme, the scheme processor calls the scheme's `set_snr()` method, passing the SNR value as an argument (cf. line 13 in Listing 5.9 on page 99).

When a scheme's `set_snr()` method is called, it stores the new SNR in its `snr` property and calls the `snr_updated()` method. The job of this method is to update the state of the scheme object to reflect the new SNR. For example, after a call to a scheme object's `set_snr()` method, its `mse` property should be set to the mean square error that the scheme achieves with the given SNR. How exactly a scheme updates its state when `snr_updated()` is called is up to the implementation. As we will see below, though, the procedure is very similar for a large class of schemes, of which JSCsim takes advantage.

After calling a scheme's `set_snr()` method, a scheme processor can access

[11] An *abstract method* is a method that is declared in a class but not implemented. For example, a class `Shape` might declare an abstract method `compute_area()`, which must be implemented by the derived classes `Circle`, `Square`, and `Triangle`. A class that has abstract methods cannot be instantiated.

[12] By default, the source samples are normally distributed, but this behavior may easily be changed by a derived class.

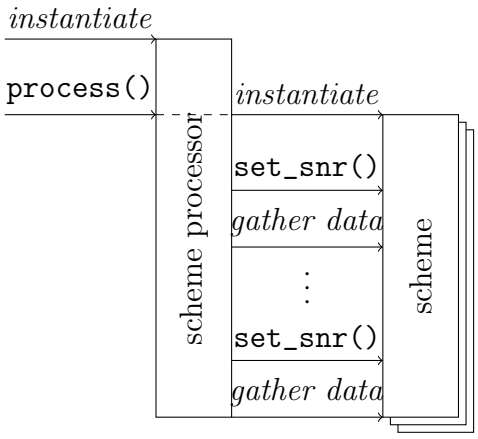


Figure 5.4: Interaction between a scheme processor and scheme classes. For each scheme, the scheme processor repeatedly calls the scheme's `set_snr()` method with different SNR values and then gathers the resulting simulation data.

the scheme's `mse` property (for instance) via the scheme's `compute_mse()` method (which we will see shortly). A scheme processor whose purpose it is to track the MSE achieved by a particular scheme thus alternatingly calls `set_snr()` and `compute_mse()` and stores the MSE for each SNR. This is precisely what the class `PerformanceProcessor` does.

Other scheme processors collect different data. For instance, the `Hybrid-2DProcessor` in Listing 5.8 on page 96 separately computes and stores the errors from the discrete and the continuous transmission rounds for the hybrid scheme in Listing 5.7.

The remainder of this section looks in detail at the implementation of the scheme classes and of the performance processors.

Scheme Classes

The common ancestor of all scheme classes is called `Scheme` and is shown in Listing 5.9 on the facing page. It defines the most rudimentary features that all communication scheme implementations must have.

The class definition starts in line 1. It is a MATLAB convention that any class whose internal state can change over time (as is the case for all classes considered here) must be derived from the `handle` class, but this is only a technical detail and of no importance in the sequel.

As we will shortly see in detail, JSCsim divides communication schemes into two types: theoretical schemes and practical schemes. Schemes of both types are (indirectly) derived from `Scheme`, so `Scheme` defines only the behavior common to both types, which consists essentially of the methods to set the SNR and to retrieve the MSE.

Every communication scheme has access to the three properties defined in

```
1  classdef Scheme < handle
2      properties (Access = 'protected')
3          sv    % Source variance.
4          snr   % SNR (power per channel input symbol).
5          mse   % Mean squared error resulting from the simulation.
6      end
7
8      methods (Access = 'public')
9          function obj = Scheme(sv, s)
10             obj.sv = sv;
11         end
12
13         function set_snr(obj, snr)
14             obj.snr = snr;
15             snr_updated(obj);
16         end
17
18         function mse = compute_mse(obj)
19             mse = obj.mse;
20         end
21     end
22
23     methods (Access = 'protected', Abstract = true)
24         snr_updated(obj)
25     end
26 end
```

Listing 5.9: The `Scheme` class is the base class from which all communication scheme classes are derived.

lines 2 to 6: the source variance, the SNR, and the incurred mean squared error. The source variance is passed via the constructor whenever a class derived from `Scheme` is instantiated (line 9). The SNR is set by `set_snr()` (line 13). The MSE property is not set by any of the class' methods; it will be the job of the derived classes to update this property in their `snr_updated()` method.

You may wonder why the constructor in line 9 receives an argument `s` but ignores it. The reason is that a scheme processor always passes the list of source symbols to a scheme it processes, even if it is a theoretical scheme that does not use the source symbols. More about this is explained further below.

We have already mentioned the method `set_snr()` (line 13). It is called by a scheme processor whenever the communication scheme is to be simulated for a new SNR. This method first saves the new SNR in its `snr` property (line 14) and then calls the abstract method `snr_updated()` to inform derived classes about the changed SNR.

The method `compute_mse()` allows scheme processors to access a scheme's `mse` property. (Because the property is declared as *protected* (cf. line 2), it cannot be directly accessed from outside the class, only through a method.)

Line 24 declares the abstract method `snr_updated()`. Because `Scheme` itself does not implement this method, the class cannot be instantiated directly. Instead, any class derived from `Scheme` must implement `snr_updated()`.[13]

Theoretical Schemes

Theoretical schemes could also be called "virtual" schemes. They are not actual communication strategies, they rather compute theoretical values for a given SNR. The purpose of a theoretical scheme is to compare the behavior of a communication scheme to a theoretically expected behavior. For an example, see the class `ShannonScheme` in Listing 5.5 on page 90.

The implementation of `TheoreticalScheme` is given in Listing 5.10 on the facing page. The class directly derives from `Scheme` (line 1); its constructor does nothing except call the base class constructor, passing on the arguments (line 4).

As mentioned before, classes derived from `Scheme` must implement the method `snr_updated()`. The implementation in `TheoreticalScheme` is quite simple: `snr_updated()` just calls `update_mse()`. The latter is itself an abstract method of `TheoreticalScheme`. This way of structuring the class is just a programmatical way of saying "the only thing a theoretical scheme needs to do when a new SNR is set is to compute the MSE as a function of the SNR". Listing 5.5 provides a sample implementation of `compute_mse()`.

Practical Schemes

As opposed to theoretical schemes, practical schemes are classes that actually *implement* a communication scheme. When the SNR is updated, a practical scheme must simulate encoding, transmission, and decoding of the source symbols. Since this procedure (except the particularities of encoding and decoding) is the same for all practical schemes, it is implemented in the class `PracticalScheme`, from which implementations of particular communication schemes can then be derived. The actual encoding and decoding functions are declared abstract in `PracticalScheme`, i.e., classes that implement communication strategies need only implement these two methods.

A simplified version of `PracticalScheme` is shown in Listing 5.11. Its `snr_updated()` method performs exactly the steps mentioned above: the source

[13] It is not quite true to say that any class derived from `Scheme` *must* implement the abstract method `snr_updated()`. A derived class is free not to implement the method; if it does not implement it, however, it remains itself an abstract class and cannot be instantiated either. A class can only be instantiated if it implements all abstract methods defined by any of its ancestors. Of course, if a class `X` has already implemented an abstract class declared in one of its ancestors, then classes derived from `X` no longer need to implement the abstract method themselves because they inherit the implementation from `X`.

```
1  classdef TheoreticalScheme < Scheme
2
3      methods (Access = 'public')
4          function obj = TheoreticalScheme(sv, s)
5              obj@Scheme(sv, s);
6          end
7      end
8
9      methods (Access = 'protected')
10         function snr_updated(obj)
11             update_mse(obj);
12         end
13     end
14
15     methods (Access = 'protected', Abstract = true)
16         update_mse(obj)
17     end
18
19 end
```

Listing 5.10: Simplified implementation of the class `TheoreticalScheme`.

sequence `s` is encoded into a channel input sequence `x` (line 5), transmission is simulated by adding Gaussian noise to `x` (line 8), and the channel output `y` is decoded to yield the estimate sequence `sh` (line 11). Finally, the MSE is computed and stored in the `mse` property (line 14). This is the precisely the property that a performance processor accesses when it calls the scheme's `compute_mse()` method (which we saw in the base class `Scheme`).

The methods `encode()` and `decode()` are the defining feature of a communication scheme. There is no "default" encoder and decoder, hence `PracticalScheme` does not provide an implementation of them but declares them as abstract. They must be implemented by derived classes (such as `UncodedScheme` in Listing 5.4).

Scheme Processors

Scheme processors instantiate a list of scheme classes and process them for a range of SNR values. All scheme processors derive from `SchemeProcessor`, a simplified implementation of which is given in Listing 5.12.

All scheme processors have at the three properties defined in lines 3–5. The vector `s` stores the source sequence, which is created in the constructor upon instantiation (line 16). The other two properties, `schemes` and `parameters`, hold the list of schemes to process and the corresponding parameters, respectively.

`SchemeProcessor` also has three *public* properties. Because they are declared

```
1  classdef PracticalScheme < Scheme
2      methods (Access = 'protected', Sealed = true)
3          function snr_updated(obj)
4              % Encode the source.
5              obj.x = encode(obj, obj.s);
6
7              % Transmit across AWGN channel.
8              obj.y = transmit(obj, obj.x);
9
10             % Decode the channel output.
11             obj.sh = decode(obj, obj.y);
12
13             % And compute the empirical MSE.
14             obj.mse = mean((obj.s - obj.sh).^2);
15         end
16     end
17
18     methods (Access = 'protected', Abstract = true)
19         x = encode(obj, s)
20         sh = decode(obj, y)
21     end
22
23 end
```

Listing 5.11: A simplified implementation of `PracticalScheme`. The methods `encode()` and `decode()` are declared abstract and must be implemented by derived classes.

public, they can be changed from outside the class; they allow the user of a scheme processor to control the latter's behavior. The meaning of these three properties should be self-evident from the code; for details see Section 5.3.

A scheme processor is launched by calling its `process()` method. This method has two arguments, `schemes` and `parameters`. Both are cell arrays that specify the schemes to be processed (i.e., the names of the respective classes) and the parameters for each scheme, respectively. See page 87 for an example of how to call `process()`.

`process()` first saves the schemes and parameters in the respective class properties (line 20 resp. lines 4–5). Then it calls the `initialize()` method (line 21). This abstract method allows scheme processor implementations to do things before the actual processing starts. The most common use of this function is to allocate a data structure that will hold the data gathered from the schemes (for an example, see the implementation of `PerformanceProcessor` in Listing 5.6 on page 94). The `do_processing()` method (line 22 resp. 28) then performs the actual processing of the schemes, we will look at it in detail

```
classdef SchemeProcessor < handle
    properties (Access = 'protected')
        s           % Source sample sequence.
        schemes     % Cell array of schemes to process.
        parameters  % Parameters for each scheme (if any).
    end
    properties (Access = 'public')
        sv = 1;               % Source variance.
        snr = 10.^(0:.1:4);   % SNR range.
        N = 100000;           % Sample size.
    end

    methods (Access = 'public')
        function obj = SchemeProcessor()
            obj.s = create_source_samples(obj);
        end

        function process(obj, schemes, parameters)
            set_schemes(obj, schemes, parameters);
            initialize(obj);
            do_processing(obj);
            post_process(obj);
        end
    end
    methods (Access = 'protected')
        function do_processing(obj)
            for j = 1:length(obj.schemes)
                scheme = create_scheme(obj, j);
                % Run the scheme for all SNR values and save data for
                % each run.
                for k = 1:length(obj.snr)
                    scheme.set_snr(obj.snr(k));
                    save_scheme_data(obj, scheme, j, k);
                end
            end
        end
    end

    methods (Access = 'protected', Abstract = true)
        initialize(obj);
        save_scheme_data(obj, scheme, j, k)
        post_process(obj)
    end
end
```

Listing 5.12: Simplified implementation of the `SchemeProcessor` class.

shortly. Finally, the abstract method `post_process()` is called (line 23). In this method, derived classes can for example display the data gathered from the schemes, or save it in a file, etc.

The function `do_processing()`, which starts in line 28, is the core of `SchemeProcessor`. In two nested *for* loops it traverses the list of schemes and the range of SNR values. For each scheme and for each SNR it first calls `set_snr()` (we have already seen above what this method does), and then it calls the abstract method `save_scheme_data()` (line 35). In their implementation of this key method, derived classes determine which data to save about the scheme. For example, the performance processor on page 87 uses it to store the MSE. (The arguments of `save_scheme_data()` are described in detail in Section 5.3.)

Summary

The preceding paragraphs have hopefully given the reader a good overview of how JSCsim is implemented. Naturally, some details have been swept under the rug. For instance, `PracticalScheme` also helps in arranging the source sequence in blocks of k source symbols if a scheme encodes more than one source symbol at once, and scheme processors use a slightly more involved method to store schemes and their parameters internally than the one shown. For these details, the reader so inclined is invited to peruse the source code, available at `http://ipg.epfl.ch/~kleiner/jscsim/`.

5.3 Reference

This reference section explains in detail how to use JSCsim to implement new communication schemes, how to simulate them, and how to build custom scheme processors.

5.3.1 Communication Schemes

There are two kinds of communication schemes in JSCsim: *theoretical* schemes and *practical* schemes. Theoretical schemes compute the MSE for a given SNR by computing some function of it without simulating anything. They allow you to compare the performance of a particular communication scheme with a theoretical value, as for example the `ShannonScheme` in Listing 5.5. On the other hand, practical schemes (like the one in Listing 5.4) determine the MSE by actually simulating communication.

Theoretical Schemes

Theoretical communication schemes are implemented as classes derived from `TheoreticalScheme`. Any class derived from `TheoreticalScheme` must implement the following methods.

⟨*class name*⟩(sv, s [, ⟨*parameters*⟩]) (public)

This is the constructor of the class. The first two arguments are the source variance sv and the source sample sequence sv, which must be passed on to the base class constructor, i.e., by calling

```
1 obj@TheoreticalScheme(sv, s);
```

If a theoretical scheme has parameters, such as the bandwidth expansion factor n of ShannonScheme (cf. Listing 5.5), they must be specified as additional arguments to the constructor.

update_mse(⟨*obj*⟩) (protected)

This method is called by the base class whenever the SNR is changed. It must update the mse property based on the value of the snr property. For example, the update_mse() method of ShannonScheme sets mse to be $\sigma_S^2/(1+\mathrm{SNR})^n$.

Practical Schemes

Practical communication schemes are implemented as classes derived from PracticalScheme. Any class derived from PracticalScheme must implement the following methods.

⟨*class name*⟩(sv, s [, ⟨*parameters*⟩]) (public)

This is the constructor of the class. The first two arguments are the source variance sv and the source sample sequence s, which must be passed on to the base class constructor. The base class constructor has two more arguments, which are the number k of source symbols encoded at a time, and the number n of channel symbols produced for every k source symbols. Depending on the scheme, these can be fixed values (as in UncodedScheme in Listing 5.4) or variable parameters of the scheme itself.

Example: A scheme that only works for 1:2 bandwidth expansion would have a constructor similar to the following.

```
1 function obj = MyScheme1(sv, s)
2   obj@PracticalScheme(sv, s, 1, 2);
3   % rest of constructor ...
4 end
```

On the other hand, a scheme that encodes one source symbol into n channel inputs, where n is arbitrary, would define a constructor like this.

```
1 function obj = MyScheme2(sv, s, n)
2   obj@PracticalScheme(sv, s, 1, n);
3   % rest of constructor ...
4 end
```

`x = encode(`⟨*obj*⟩`, s)` (protected)

This method receives `s`, a matrix of source samples with k rows, and must return a matrix `x` with n rows and the same number of columns as `s`.

`sh = decode(`⟨*obj*⟩`, y)` (protected)

This method receives the channel output `y` as a matrix with n rows and must return a matrix `sh` with k rows of source estimates

In addition, derived classes may want to override `update_variable_parameters()`.

`update_variable_parameters(`⟨*obj*⟩`)` (protected)

In this method, which is called whenever the SNR changes, derived classes can update any of their own properties that depend on the SNR. It is important that if a derived class overrides this method, it must first call the base class method, i.e.,

```
function update_variable_parameters_(obj)
  % Call base class version.
  update_variable_parameters@PracticalScheme(obj);
  % Your code here ...
end
```

To allow a scheme processor to track a particular property of a communication scheme, the property must be made accessible. This is normally done by writing a `compute_*` method. We have already seen the example of the `compute_mse()` function, which is defined by the class `Scheme`. The class `PracticalScheme` in addition implements the following such methods.

`x = compute_x(`⟨*obj*⟩`)` (public)

Returns the channel input sequence.

`y = compute_y(`⟨*obj*⟩`)` (public)

Returns the channel output sequence.

`sh = compute_sh(`⟨*obj*⟩`)` (public)

Returns the sequence of source estimates.

Any derived class must thus implement a `compute_*` function for each property it wants to make accessible to a scheme processor.

5.3.2 Scheme Processors

Scheme classes are usually not handled directly; rather, they are processed by a *scheme processor*. Scheme processors evaluate a set of communication schemes

for a range of SNR values and store and process the data gathered during the simulations.

All scheme processors are implemented as classes derived from `SchemeProcessor`. To process a communication scheme (or a set of schemes) you never directly use a `SchemeProcessor` object. Instead, you either use an existing derived class such as `PerformanceProcessor`, or you implement a custom scheme processor.

General Aspects

The behavior of all scheme processors can be controlled through the following properties, defined in the class `SchemeProcessor`.

`snr` (public; default `10.^(0:.1:4)`)

A vector that specifies the SNR range over which the communication schemes are simulated. By default it is set to the range 0dB to 40dB with increments of 1dB.

`N` (public; default `100000`)

The length of the random source sample sequence.

`verbose` (public; default `false`)

If this parameter is set to true, various status and debugging messages are displayed during the simulations.

`output_module` (public)

This is the output module used by the scheme processor to display its results (see the section on output modules). The default output module is `MatlabPlotModule`.

`legendmode` (public, default `'auto'`)

This property determines the behavior of the `plot_vs_csnr()` and `plot_vs_csnr_db()` functions (cf. the definition of `post_process()` below). If `legendmode` is set to `'on'`, a legend is always displayed. If it is set to `'off'`, a legend is never displayed. If it is set to `'auto'`, a legend is only displayed if more than one scheme is plotted.

All scheme processors are launched using the `process()` method.

`process(`⟨*obj*⟩`, schemes, parameters)` (public)

Process the specified schemes for the specified parameters. The cell array `schemes` lists the schemes to be processed; each of its elements is a string equal to the name of a scheme class.

The cell array `parameters` has the same number of elements as `schemes`. For a scheme that does not have any parameters, the corresponding entry

of `parameters` must be an empty matrix. For a scheme that accepts m parameters (i.e., its constructor has m arguments other than `sv` and `s`), the corresponding entry of `parameters` must be a matrix with m rows; the scheme is then processed once for each column of the matrix, using the parameters from the respective column. This makes it easy to use MATLAB's colon operator (`:`) to specify a *range* of parameters.

Example: Suppose `process()` is called as follows.

```
1 pp = PerformanceProcessor();
2 pp.process({'MyScheme'}, {[1:4; 0.1:0.1:0.4]});
```

Then `MyScheme` will be processed 4 times: once with the parameters 1 and 0.1, once with the parameters 2 and 0.2, and so on. I.e., line 2 above has the same effect as

```
1 pp.process({'MyScheme', 'MyScheme', 'MyScheme', 'MyScheme'}, ...
2   {[1;0.1], [2;0.2], [3;0.3], [4;0.4]});
```

Internally, a scheme is counted as many number of times as its parameter matrix has columns. This means that for the above example, `nb_schemes()` (cf. below) returns 4.

The Performance Processor

This is the only scheme processor that is already implemented in JSCsim. It works for all communication schemes derived from `Scheme`. It simply plots the SDR vs SNR curve of the specified communication scheme, as illustrated by the examples in Section 5.2.1.

To have fine grained control over the plot of the simulation results, the plot functionality of a performance processor can be disabled by setting its `output_module` property to the empty matrix[14]. The simulation results can then be read from the performance processor's `mse` property, which is declared `public`, and plotted with a custom output module. This is the recommended strategy if you want to manually specify say the plot legend or the axis labels, as illustrated by the example in Listing 5.13. For this reason, it is also recommended that custom scheme processors (see the following paragraph) declare their simulation results data as `public`.

Section 5.3.3 has a detailed description of output modules.

Implementing a New Scheme Processor

Custom scheme processors are implemented by creating a new class derived from `SchemeProcessor`. Such a class must implement the three following methods.

[14] or any other value for which MATLAB's `isempty()` function returns `true`, such as the empty string `''`, the empty cell `{}`, etc.

```
pp = PerformanceProcessor();
pp.output_module = [];  % Disable built-in output module.
pp.process({'ShannonScheme', 'UncodedScheme'}, {1, []});

% Create own output module.
om = MatlabPlotModule();

% Access SNR and simulation result from PerformanceProcessor object.
om.x = 10*log10(pp.snr);  % converted to dB
om.y = 10*log10(pp.sv ./ pp.mse);  % SDR = source variance / mse

% Set custom labels, legend, etc.
om.legend = {'theoretical limit', 'uncoded communication'};
% ...

% Display plot.
om.do_plot();
```

Listing 5.13: This example shows how the internal output module of a performance processor can be disabled and the simulation results can be plotted on a custom plot.

`initialize`(⟨*obj*⟩) (protected)

This method is called before the actual processing starts. It can be used e.g. to allocate data structures to save simulation data.

`SchemeProcessor` provides useful helper function, `nb_schemes()`, which returns the number of schemes that have been passed to `process()`. In addition, `length(obj.snr)` gives you the number of values in the SNR range.

`save_scheme_data`(⟨*obj*⟩, `scheme`, `j`, `k`) (protected)

This method is called each time a scheme has been processed for a particular SNR and gives you the opportunity to save data about the simulation run. `scheme` is the scheme object that was just run; you can gather data about it by calling its public methods. For example, the `save_scheme_data()` method of `PerformanceProcessor` calls the `compute_mse()` method of `scheme` to store the MSE.

`j` is a number between 1 and `nb_schemes()` and `k` is a number between 1 and `length(obj.snr)`; they refer to the current scheme and the current SNR value, respectively. Listing 5.6 shows a typical example of how they are used.

`post_process`(⟨*obj*⟩) (protected)

This method is called after all schemes have been processed. Here you can

post process the data gathered by `save_scheme_data()`, for instance by plotting it.

`SchemeProcessor` provides the helper function `plot_vs_snr(obj, m)`, which plots the data in the vector `m` against the SNR in dB. (The SNR is in dB, not the data; if you also want the data to be plotted on a dB scale, use `plot_vs_snr_db()` instead.)

To change the default output module, override the following method.

`om = default_output_module(`⟨*obj*⟩`)` (protected)

This method is called by the constructor of `SchemeProcessor` to install the default output module.

In addition, a custom scheme processor can override `create_source_samples()` to change the source distribution.

`s = create_source_samples(`⟨*obj*⟩`)` (protected)

This method must return a sequence of `N` independent random source samples of variance `sv`. The default implementation in `SchemeProcessor` creates Gaussian source samples.

5.3.3 Output Modules

In some cases you might just want to see the results of a simulation on screen; in other cases you might want to save them in a file that you can include in a paper or report. In JSCsim it is easy to change the default behavior by changing the output module.

Output modules provide an abstraction of basic plot functionalities. You can think of them as a kind of output "plugins". The functionality they offer is rather orthogonal to the task of simulating; they may well be used in other programs as well.

In JSCsim, each class derived from `SchemeProcessor` has an output module associated to it. By default this is the `MatlabPlotModule`, which displays the plots in a regular MATLAB figure window, but it is easy to change the plot module in order to save the plot in a PDF file rather than on screen, for instance:

```
1 pp = PerformanceProcessor();
2 pp.output_module = MatlabFilePlotModule();
3 pp.output_module.fn = 'myplot.pdf';
```

The behavior of output modules is controlled by a set of parameters. Some of these, such as the axis labels or the legend entries, apply to all output modules. Other parameters apply only to certain categories of output modules: the `fn` property, for instance, which determines the name of the file in which to save a plot, only applies to those output modules that can save plots in a file.

General Parameters

All output modules support the following properties and methods.

`x` (public)

The range of x values. This must be a row vector.

`y` (public)

The data to plot against the x values. This must be either a row vector or a matrix with the same number of columns as `x`; each row then corresponds to a separate data series to plot.

`xlabel` (public)

The label of the x-axis. This is a string; it can also contain (limited) LaTeX code depending on the actual output module used.

`ylabel` (public)

The label of the y-axis. This is a string; it can also contain (limited) LaTeX code depending on the actual output module used.

`plottitle` (public)

The plot title. This is a string; it can also contain (limited) LaTeX code depending on the actual output module used.

`legend` (public)

The legend entries. This must be a cell array of strings. If the number of legend entries is smaller than the number of data series in `y`, a warning is issued.

`legendpos` (public)

A string determining where in the plot the legend is placed. The possible values are `NorthEast`, `NorthWest`, `SouthEast`, `SouthWest`, and `NorthEastOutside`. If `legendpos` is set to the empty string, no legend is created.

`grid` (public; default `false`)

This boolean parameter determines whether a grid is drawn (if set to `true`) or not (if set to `false`).

`set_color_mode(`⟨*obj*⟩`, c)` (public)

Set the color mode of the plot. `c` can either be `'color'`, in which case a line of a different color is drawn for each data series, or `'bw'`, in which case the plot uses black lines with a different marker for each data series.

To use an output module on its own, i.e., outside of a scheme processor, call the `do_plot()` method.

`do_plot(`⟨*obj*⟩`)` (public)

Plots the data set specified by `x` and `y`.

The MatlabPlotModule Output Module

This output module creates a standard MATLAB plot. It is the default output module for scheme processors. It has a single public property.

`ah` (public)

A handle to the axes that will contain the plot. By default, `MatlabPlotModule` creates a new figure window and sets `ah` to point to the axes of the new windows. You can have `MatlabPlotModule` to plot on an existing set of axes by changing the `ah` property.

Example: To create the plot in a subplot of an existing figure window:

```
1 figure;
2 ah1 = subplot(2,1,1);
3 pp = PerformanceProcessor();
4 pp.output_module.ah = ah1;
```

Subsequent plots now appear in the specified subplot.

The MatlabFilePlotModule Output Module

The `MatlabFilePlotModule` works just like the `MatlabPlotModule`, except that the plot is not displayed in a window but saved in a file. The result is the same as if you had selected "File/Save as..." in a figure window.

The behavior of `MatlabFilePlotModule` is controlled by the following properties.

`fn` (public)

The name of the file in which to save the figure. By default an error occurs if a file of the given name already exist; this can be changed using the `force` property (see below).

When a file name is set, the class tries to determine the file type from the extension. The following extensions are recognized: `ps`, `eps`, `jpg`, `jpeg`, `png`, and `pdf`. If the file name has one of these extensions, it is not necessary to set the `type` property manually.

`type` (public)

A string denoting the file type. The set of valid file types is the same as for the MATLAB function `print`; see the help for that function for a complete list.

`pdfres` (public; default `600`)

An integer denoting the resolution (in dpi) of the created file if the file type is `pdf`. The default value is 600 dpi, which is a typical value for production quality.

`force` (public; default `false`)

If this option is set to `true`, existing files are overwritten; if it is set to `false` then an error occurs if a file of the given name already exists.

The PGFPlotsOutputModule Output Module

This output module writes the plot to a file in a format suitable for the PGFPLOTS package for LaTeX. A file generated by `PGFPlotsOutputModule` can be included in a LaTeX source file, provided that the PGFPLOTS package has been loaded.

The behavior of this module is controlled by the following properties.

`fn` (public)

The name of the file in which to save the figure. By default an error occurs if a file of the given name already exist; this can be changed using the `force` property (see below).

`force` (public; default `false`)

If this option is set to `true`, existing files are overwritten; if it is set to `false` then an error occurs if a file of the given name already exists.

Implementing Custom Output Modules

A new output module is implemented by creating a class derived from `OutputModule` (or from one of its derived classes). Any class derived from `OutputModule` must implement the following two methods.

`do_actual_plot(`⟨*obj*⟩`)` (protected)

This method creates the actual plot using the data set in the class properties. Before this method is called, it has already been verified by the base class that the data properties `x` and `y` have been set, that they have the right format, etc.

`set_color_mode(`⟨*obj*⟩`, c)` (public)

This method is called with the parameter `c` being either `'color'` or `'bw'`. Its task is to set up the state of the class such that the plot created by `do_actual_plot()` is in color or readable on black/white, respectively.

If an output module has other properties whose validity should be checked before plotting is done then it can override `check_parameters()`.

`check_parameters(`⟨*obj*⟩`)` (protected)

This method is called just before `do_actual_plot()`. If it is overriden by a derived class, the overriding function *must* call the base class version of method, since otherwise important checking is not done.

Example: `MatlabFilePlotModule` and `PGFPlotsOutputModule` both override this method to check whether the file to write the plot to already exists.

6 Conclusions and Outlook

The characterization of the achievable cost and distortion region of point-to-point communication systems under a delay constraint is an important unsolved problem in information theory. In this thesis we have looked at the particular case of minimal-delay transmission with bandwidth expansion across Gaussian channels. We have analyzed a hybrid transmission strategy based on quantization and uncoded transmission. This strategy is by no means new; it has appeared previously in various shapes (see the historical notes section in Chapter 4). Here we have established a justification for this strategy, inspired by the case with feedback, arguing that *any* minimal-delay bandwidth expansion scheme with uncoded components should use uncoded transmission only in the *last* channel use. Furthermore, we have exactly characterized the scaling behavior of the signal-to-distortion ratio (SDR) achieved when the signal-to-noise ratio (SNR) goes to infinity.

To date there is no known minimal-delay communication scheme for the Gaussian channel with bandwidth expansion that achieves an SDR that scales better than the hybrid strategy presented here, namely $\mathrm{SNR}^n / (\log \mathrm{SNR})^{n-1}$. Where does the $\log \mathrm{SNR}$ factor come from? Can it be explained by the particular nature of our hybrid scheme, or is there something more fundamental to it? Can the optimal SDR scaling of SNR^n be achieved at all with minimal delay for $n > 1$? These are open questions that should be investigated in the future, the ultimate goal being to completely characterize the achievable cost and distortion region.

A communication system achieves an optimal fidelity–cost tradeoff if the elements making up the system are properly matched, which requires that the cost and distortion measures are related in a certain way to the statistics of the communication system. This has been known before [14]. We have shown here that the set of cost and distortion measures for which a given communication

system is thus matched has a *subset* of measures for which the *ratio* of fidelity per cost is maximized, in the sense that no alternative encoder or decoder can increase this ratio. The set of communication systems operating at maximal fidelity per cost is thus a subset of those communication systems that achieve an optimal fidelity–cost tradeoff.

Whether achieving the maximal ratio of fidelity per cost is of practical value is not conclusively clear. The challenge lies in finding a fidelity measure for which one can quantitatively compare the performance of encoding many source symbols at low fidelity with encoding few source symbols at high fidelity. The conclusion to Chapter 2 mentions one potential application; it might be interesting to search for others.

The chapter on simulation was written with two goals in mind. The first was to show how object-oriented techniques are particularly helpful for simulations. The second goal was to make the JSCSIM simulator freely available to the public. The author hopes that it may save at least some students from the frustrating experience of simulator code gone chaotic.

It may yet take a long time until the fundamental results of information theory are extended to completely take into account delay constraints. Meanwhile, it is the author's hope that the results contained in this thesis indicate some of the avenues to follow and some of the questions to investigate. If one day a complete answer has been found, notify him – without delay!

Bibliography

[1] M. Abramowitz and I. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Dover Publications, 1964.

[2] R. Blahut and R. Koetter, Eds., *Codes, Graphs, and Systems.* Springer Verlag, 2002.

[3] B. Chen and G. Wornell, "Analog error-correcting codes based on chaotic dynamical systems," *IEEE Transactions on Communications*, vol. 46, no. 7, pp. 881–890, Jul 1998.

[4] J. Conway and N. J. A. Sloane, *Sphere packings, lattices and groups.* New York: Springer Verlag, 1988.

[5] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth, "On the Lambert W Function," *Advances in Computational Mathematics*, vol. 5, no. 1, pp. 329–359, September 1996.

[6] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* New York: John Wiley & Sons, 1991.

[7] H. Coward and T. A. Ramstad, "Quantizer optimization in hybrid digital-analog transmission of analog source signals," in *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2000, pp. 2637–2640.

[8] H. Coward, "Joint source–channel coding: Development of methods and utilization in image communications," Ph.D. dissertation, Norwegian University of Science and Technology, 2001.

[9] H. Coward and T. A. Ramstad, "Bandwidth doubling in combined source-channel coding of memoryless Gaussian sources," in *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Honolulu, HI, November 2000, pp. 571–576.

[10] P. Floor and T. Ramstad, "Noise analysis for dimension expanding mappings in source-channel coding," in *2006 IEEE Workshop on Signal Processing Advances in Wireless Communications*, July 2006, pp. 1–5.

[11] R. G. Gallager, "Energy limited channels: Coding, multiaccess and spread spectrum," in *Proceedings of the Conference on Information Science and Systems*, March 1988, p. 372.

[12] M. Gastpar, "To code or not to code," Ph.D. dissertation, Ecole Polytechnique Fédérale de Lausanne, 2002.

[13] M. Gastpar and B. Rimoldi, "Source-channel communication with feedback," in *2003 IEEE Information Theory Workshop*, 2003, pp. 279–282.

[14] M. Gastpar, B. Rimoldi, and M. Vetterli, "To code, or not to code: lossy source-channel communication revisited." *IEEE Transactions on Information Theory*, vol. 49, no. 5, pp. 1147–1158, 2003.

[15] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Boston: Kluwer Academic Publishers, 1992.

[16] T. Goblick, Jr., "Theoretical limitations on the transmission of data from analog sources," *IEEE Transactions on Information Theory*, vol. 11, no. 4, pp. 558–567, Oct 1965.

[17] F. Hekland, P. Floor, and T. Ramstad, "Shannon-kotel'nikov mappings in joint source-channel coding," *IEEE Transactions on Communications*, vol. 57, no. 1, pp. 94–105, January 2009.

[18] M. Horstein, "Sequential transmission using noiseless feedback," *IEEE Transactions on Information Theory*, vol. 9, no. 3, pp. 136–143, July 1963.

[19] A. Ingber, I. Leibowitz, R. Zamir, and M. Feder, "Distortion lower bounds for finite dimensional joint source-channel coding," in *2008 IEEE International Symposium on Information Theory*, Toronto, Canada, 2008, pp. 1183–1187.

[20] M. Kleiner and B. Rimoldi, "Asymptotically optimal joint source-channel coding with minimal delay," in *2009 IEEE Globecom Communication Theory Symposium*, Honolulu, HI, December 2009.

[21] ——, "On fidelity per unit cost," in *2009 IEEE International Symposium on Information Theory*, Seoul, South Korea, July 2009, pp. 1639–1643.

[22] ——, "A tight bound on the performance of a minimal-delay joint source-channel coding scheme," in *2010 IEEE International Symposium on Information Theory*, Austin, TX, June 2010.

[23] V. A. Kotel'nikov, *The Theory of Optimum Noise Immunity.* New York: Dover Publications, 1960.

[24] D. McRae, "Performance evaluation of a new modulation technique," *IEEE Transactions on Communication Technology*, vol. 19, no. 4, pp. 431–445, August 1971.

[25] U. Mittal and N. Phamdo, "Hybrid digital-analog (HDA) joint source-channel codes for broadcasting and robust communications," *IEEE Transactions on Information Theory*, vol. 48, no. 5, pp. 1082–1102, May 2002.

[26] T. A. Ramstad, "Shannon mappings for robust communication," *Telektronikk*, vol. 98, no. 1, pp. 114–128, 2002.

[27] F. M. Reza, *An Introduction to Information Theory.* Dover Publications, 1994.

[28] Z. Reznic, M. Feder, and R. Zamir, "Distortion bounds for broadcasting with bandwidth expansion," *IEEE Transactions on Information Theory*, vol. 52, no. 8, pp. 3778–3788, Aug. 2006.

[29] D. J. Sakrison, *Notes on Analog Communication*, G. L. Turin, Ed. Van Nostrand Reinhold, 1970.

[30] J. Schalkwijk and T. Kailath, "A coding scheme for additive noise channels with feedback–i: No bandwidth constraint," *IEEE Transactions on Information Theory*, vol. 12, no. 2, pp. 172–182, Apr 1966.

[31] L. Scharf, *Statistical Signal Processing.* Addison-Wesley, 1990.

[32] S. Shamai (Shitz), S. Verdú, and R. Zamir, "Systematic lossy source/channel coding," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 564–579, Mar 1998.

[33] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, July and October 1948.

[34] ——, "Communication in the presence of noise," *Proceedings of the I.R.E.*, vol. 37, pp. 10–21, January 1949.

[35] ——, "Coding theorems for a discrete source with a fidelity criterion," in *IRE Convention Record*, vol. 7, no. 4, 1959.

[36] O. Shayevitz and M. Feder, "Communication with feedback via posterior matching," in *2007 IEEE International Symposium on Information Theory*, Nice, France, 2007.

[37] ——, "The posterior matching feedback scheme: Capacity achieving and error analysis," in *2008 IEEE International Symposium on Information Theory*, Toronto, Canada, 2008.

[38] M. Skoglund, N. Phamdo, and F. Alajaji, "Design and performance of vq-based hybrid digital-analog joint source-channel codes," *IEEE Transactions on Information Theory*, vol. 48, no. 3, pp. 708–720, Mar 2002.

[39] B. Stroustrup, *The C++ Programming Language, Third Edition.* Addison-Wesley, 1997.

[40] M. Taherzadeh and A. K. Khandani, "Robust joint source-channel coding for delay-limited applications," *arXiv:0805.4023v1 [cs.IT]*, 2008. [Online]. Available: http://arxiv.org/abs/0805.4023

[41] V. Vaishampayan and S. Costa, "Curves on a sphere, shift-map dynamics, and error control for continuous alphabet sources," *IEEE Transactions on Information Theory*, vol. 49, no. 7, pp. 1658–1672, July 2003.

[42] S. Verdú, "On channel capacity per unit cost." *IEEE Transactions on Information Theory*, vol. 36, no. 5, pp. 1019–1030, 1990.

[43] ——, "The exponential distribution in information theory." *Problemy Peredachi Informatsii (Problems of information transmission)*, vol. 32, no. 1, pp. 100–111, 1996.

[44] ——, "Spectral efficiency in the wideband regime," *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1319–1343, Jun 2002.

[45] N. Wernersson, T. A. Ramstad, and M. Skoglund, "Analog source-channel codes based on orthogonal polynomials," in *Asilomar Conference on Signals, Systems and Computers*, 2007.

[46] Wikipedia, "Object-oriented programming – Wikipedia, the free encyclopedia," 2010, accessed March 4, 2010. [Online]. Available: http://en.wikipedia.org/wiki/Object-oriented_programming

[47] J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering*. New York: John Wiley & Sons, 1965.

[48] J. Ziv, "The behavior of analog communication systems," *IEEE Transactions on Information Theory*, vol. 16, no. 5, pp. 587–594, Sep 1970.

[49] J. Ziv and M. Zakai, "Some lower bounds on signal parameter estimation," *IEEE Transactions on Information Theory*, vol. 15, no. 3, pp. 386–391, 1969.

[50] ——, "On functionals satisfying a data-processing theorem," *IEEE Transactions on Information Theory*, vol. 19, no. 3, pp. 275–283, May 1973.

Curriculum Vitæ

Marius Kleiner

Address
Edisonstrasse 20
8050 Zürich
Switzerland

Personal
Born 12 July 1979
in Zürich, Switzerland
Swiss Citizen

EDUCATION

2005–2010 **Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland**
PhD thesis *"Strategies for Delay-Limited Source-Channel Coding"*
(supervisor: Prof. Bixio Rimoldi)

1999–2005 **Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland**
B.Sc./M.Sc. in Communication Systems Engineering

2001–2002 **Carnegie Mellon University, Pittsburgh, PA**
Exchange Program in Electrical and Computer Engineering

1992–1999 **Realgymnasium Rämibühl, Zürich, Switzerland**
High School Degree

EMPLOYMENT HISTORY

2005–2010 **Mobile Communications Laboratory, EPFL**
Research Assistant

2004–2005 **Qualcomm, Inc., San Diego, CA**
Engineering Intern

2003 **Logitech, Inc., Fremont, CA**
Video Software Intern

PUBLICATIONS

1. M. Kleiner and B. Rimoldi, "On fidelity per unit cost," in *2009 IEEE International Symposium on Information Theory*, Seoul, South Korea, July 2009, pp. 1639–1643.

2. ——, "Asymptotically optimal joint source-channel coding with minimal delay," in *2009 IEEE Globecom Communication Theory Symposium*, Honolulu, HI, December 2009.

3. ——, "A tight bound on the performance of a minimal-delay joint source-channel coding scheme," in *2010 IEEE International Symposium on Information Theory*, Austin, TX, June 2010.

LANGUAGE SKILLS

German: native
English: fluent
French: fluent
Italian: basic