TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

Institute of Telecommunications
TU Wien

Master Thesis

Sparse Superposition Codes for the Gaussian Channel

Author:

William Holt
0926965

Supervisors:

Univ.Prof. Dipl.-Ing. Dr.-Ing. Norbert Görtz
Univ.Ass. M.Sc. Osman Musa

May 2018

Declaration of Authorship

Hiermit erkläre ich, William Holt
0926965, dass die vorliegende Arbeit gemäß dem Code of Conduct – Regeln zur Sicherung guter wissenschaftlicher Praxis (in der aktuellen Fassung des jeweiligen Mitteilungsblattes der TU Wien), insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In– noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Unterschrift:

Datum:

TU Wien

Abstract

Institute of Telecommunications

Sparse Superposition Codes for the Gaussian Channel

by William Holt
0926965

Recently sparse regression codes / sparse superposition codes have been proposed and it has been shown that those codes asymptotically (that is for large block size) achieve the performance limits of information theory.

The codes are formed from Gaussian random sequences that are arranged in the columns of a large matrix, and groups of data bits are used to identify which column-vectors are chosen for superimposed transmission over a Gaussian channel. Hence the coding process can be written as a matrix-vector multiplication, with the matrix containing the Gaussian code-sequences and the data vector containing only few non-zero components (that are determined by the data bits) that identify which of the code-column vectors are used to form a codeword. Since the vector is sparse, the decoding can be understood as a sparse recovery problem that has been recently and intensively studied in the field of compressed sensing. Some of the best sparse recovery algorithms are based on approximate message passing (AMP) which is also a good candidate decoder for sparse regression codes (SPARCs).

A particularity of the SPARCs is that for good decoding by AMP not only values of constant amplitude are used to identify which of the column vectors are chosen but rather a sophisticated power allocation scheme that determines the scaling factors is required, and only then theoretical limits can be achieved.

The first goal of the thesis is to understand the construction, encoding, power allocation, and decoding of SPARCs for the unconstrained Gaussian channel from the recent literature. In a second step, the codes shall be implemented (mostly in Matlab) and simulations shall be conducted to verify performance known from the literature.

In terms of novelty, several interesting topics occur: one is to investigate SPARCs for various block sizes to understand how those codes would work in practical applications with strong delay constraints, that naturally limit the block size. Another interesting question is if it makes sense to concatenate sparse regression codes with classical binary channel codes and, if possible, to apply iterative decoding between the component codes.

Contents

Declaration of Authorship iii

Abstract v

1 Introduction and setting 1
- 1.1 Introduction . . . 1
  - 1.1.1 Notation . . . 3
- 1.2 Setting . . . 3
  - 1.2.1 Shaping Gain . . . 6
- 1.3 Related Literature . . . 7

2 Communicating over an AWGN channel 9
- 2.1 Introduction . . . 9
  - 2.1.1 Notation . . . 9
- 2.2 Transmission setting and decoding in general . . . 9
  - 2.2.1 Channel description . . . 10
  - 2.2.2 General decoding . . . 10
  - 2.2.3 Relation to compressed sensing . . . 11
- 2.3 Approximate message passing decoder . . . 11
  - 2.3.1 Test statistics . . . 12
  - 2.3.2 State evolution . . . 15
- 2.4 Power Allocation . . . 17
  - 2.4.1 Constant power allocation . . . 18
  - 2.4.2 Exponentially decaying power allocation . . . 18
  - 2.4.3 Algorithmic power allocation . . . 19
  - 2.4.4 Choice of the power allocation rate R_{PA} . . . 23
- 2.5 Complexity . . . 26
  - 2.5.1 Online state evolution estimation . . . 26
  - 2.5.2 Hadamard coding matrix . . . 28
- 2.6 Estimating error rates . . . 29

3 Numerical evaluation 31
- 3.1 Introduction . . . 31
  - 3.1.1 The bit error probability and E_b/N_o . . . 31
- 3.2 Number of AMP iterations . . . 33
- 3.3 Error exponent . . . 35

3.4 Choosing L and B 37
3.5 Dependency of code parameters on the optimal R_{PA} 40
3.6 Bit error rate over SNR 41
3.7 Conclusion 43

Bibliography **45**

Chapter 1

Introduction and setting

1.1 Introduction

Since Shannon's ground breaking work on communication and information theory in [1], the quest to efficiently encode and decode data over linear Gaussian channels has occupied the attention of the researchers in the past. One major result of Shannon's work was the analytic description of the capacity of the additive white Gaussian noise channel. Given the transmit bandwidth B, the mean maximum signal power P and assuming the noise power spectral density to be equal to $N_0/2$, the signal-to-noise ratio (SNR) is defined as $SNR = P/(BN_0)$. The channel capacity C (here in bits per seconds), is given by:

$$\mathcal{C}_{[\text{b/s}]} = B \ \log_2(1 + SNR) \ . \tag{1.1}$$

This result gives an upper bound on the achievable communication rate, for which a vanishing error probability is possible. In other words, it is possible with proper encoding, to make the error probability arbitrarily small, as long as the information is sent at a rate below the capacity threshold. On the other hand exceeding the rate over the capacity limit makes reliable communication impossible. Note that $\mathcal{C}$ is an upper bound on the communication rate in relation to B and the SNR, while it does not restrict the error probability. Further $\mathcal{C}$ provides an attainable goal but does not deliver a solution of how to achieve such performance with a specific scheme or, equally important, in an efficient manner. Therefore $\mathcal{C}$ can be used as a benchmark for a given coding and transmission scheme. In [1] it was shown that a set of $\propto 2^{M\mathcal{C}}$ randomly generated and independent Gaussian codewords can reach the capacity limit with vanishing error probability, when the block length $M \to \infty$. These codewords are randomly generated and known to the receiver. With such an approach the computational complexity as well as the storage complexity grow exponentially with block length M. Since such a coding scheme would not be feasible to implement, the challenge of how to reach this limit efficiently is still an open question. The aspiration is to find a low complexity coding scheme that introduces small delay and enables reliable transmission of information close to the Shannon capacity (see (1.1)). One state-of-the-art coding approach for the linear Gaussian channel is so-called *coded modulation*, where coding of the binary data and modulating on a specific constellation are logically separate tasks. Figure 1.1 shows a block diagram of the coded modulation scheme. This approach preforms well in practical

settings and is established in many technical standards. However with a fixed constellation scheme it is not possible to reach the capacity limit over a broad SNR regime.

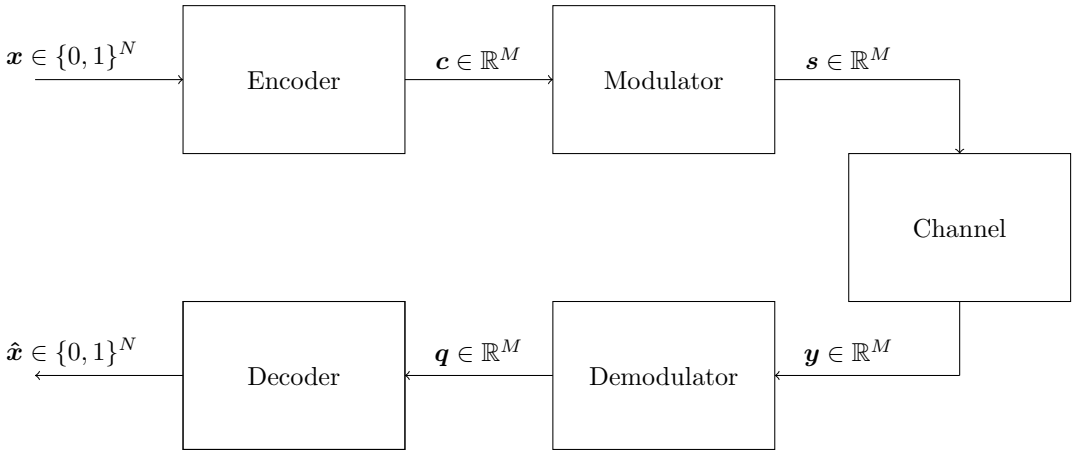


FIGURE 1.1: Illustrates the logical separation of coding and modulation in the coded modulation approach. The binary code could be for example a LDPC or Turbo code and the modulation format 4 or 16 QAM.

Figure 1.2 shows the maximal achievable rates under which reliable communication is still possible for different modulation schemes. It is apparent, that by using a fixed modulation format, for example 16 QAM, it is possible to reach the Shannon capacity only in a limited SNR region. The graphs of the modulation schemes with m possible points in the constellation diagram converge towards $\log_2(m)$ [bits/channel use], since, as a maximum number, $\log_2(m)$ number of information bits are mapped to one modulation symbol. To circumvent this restriction, modern technical standards use pairs of different channel code rates and constellation sizes, which together are called modes. For example a rate 8/9 code and 64 ASK modulation, the modes are chosen according to the underlying SNR. The goal is to choose a mode, with which reliable communication close to $\mathcal{C}$ is possible. From Figure 1.2 we can see that the 64 QAM mode covers the region of possible information rate of all lower modes. This does not mean that lower modes are unnecessary, which is explained by the following example. Assume we want to transmit information with a rate of 0.5. By applying a BPSK modulation scheme we can use a code rate of $1/2$ to get the desired communication rate. By using a 16 QAM modulation format the code rate has to be $1/8$ to reach the same communication rate. With the higher modulation format more code bits are added for the same number of information bits. Sending more code bits equals an increase in block length and therefore an increase in delay and necessary computations.

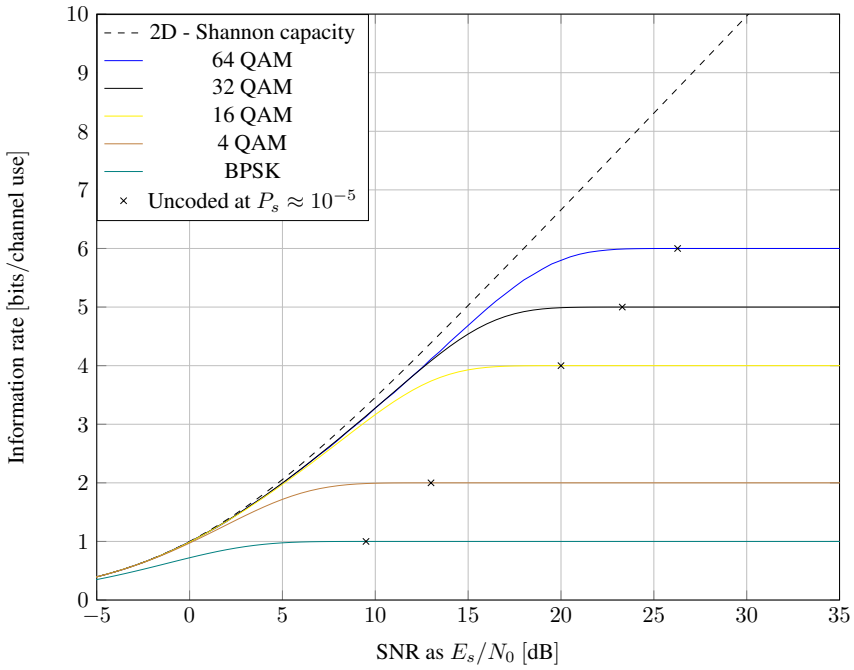


FIGURE 1.2: Capacity bounds over the SNR of different modulation formats. It is assumed that the input symbols are uniformly distributed and sent over an AWGN channel. The dashed line represents the two-dimensional unconstrained Shannon capacity, $\mathcal{C} = \log_2(1 + SNR)$. The capacity curves for the different modulation schemes are calculated with the *Coded Modulation Library* (CML), available at http://www.iterativesolutions.com/Matlab.htm.

In this thesis we take a detailed look at a code that takes a different route than coded modulation. *Sparse superposition codes* or *sparse regression codes* (SPARCs) make use of a continuous modulation alphabet and resemble Shannon's idea of a Gaussian codebook, but with reasonable complexity demands.

- In Section 1.2 the encoding procedure for SPARCs is explained and after presenting the basic principles, insight into the so-called *shaping gain* is given.
- Related literature and their core achievements are mentioned in Section 1.3.

1.1.1 Notation

Capital and bold letters $\boldsymbol{X}$ denote matrices, bold symbols $\boldsymbol{x}$ vectors and letters x scalars. $\mathbb{E}$ is the expectation, the Gaussian distribution with mean μ and variance σ^2 is denoted by $\mathcal{N}(\mu, \sigma^2)$. The l_2 norm of a vector is written as $\|\boldsymbol{x}\|$. The transpose of matrix $\boldsymbol{X}$ is denoted by $\boldsymbol{X}^\mathsf{T}$. Further for describing the i^{th} column of a matrix $\boldsymbol{X}$ a Matlab style notation is used $\boldsymbol{X}_{:,i}$. A specific element of a vector is denoted as $[\boldsymbol{x}]_i$. Additional notation is explained in text and at the beginning of the following chapters.

1.2 Setting

For the code construction assume an input message vector $\boldsymbol{x}$ of length N, which is partitioned

into L sections. With a slight abuse of notation we will denote the l^{th} section $\boldsymbol{x}_l = \{x_i \in l\}$, where $i \in sec(l)$ denotes the indices included in the l^{th} section. Therefore the input message vector can be written as $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_L]^\intercal$ with each section having B number of elements. The length of the input vector $\boldsymbol{x}$ is $N = BL$. Each of the sections $\boldsymbol{x}_l$, $l \in \{1 \dots L\}$ fulfills the constraint of having exactly one non-zero element. An example of a possible input sequence for $L = 3$ and $B = 3$ could be $\boldsymbol{x} = [(001), (100), (010)]^\intercal$. The sections can be seen as input symbols and the position of the non-zero element defines the symbol to transmit. Therefore the section size B is equal to the cardinality of the alphabet. One can also think of *one-hot-encoding* each section. The non-zero values of the input vector are positive and are related to the input power to the channel. The non-zero values for each section can be chosen differently, which will be explained in more detail later on. Next assume a so-called *coding matrix* $\boldsymbol{A} \in \mathbb{R}^{M \times N}$, then the codeword $\boldsymbol{c} \in \mathbb{R}^M$ is generated by matrix multiplication $\boldsymbol{Ax}$. At first we assume that the entries of $\boldsymbol{A}$ are i.i.d $\sim \mathcal{N}(0, \sigma^2_{\boldsymbol{A}})$. In general the encoding relation $\boldsymbol{Ax}$ corresponds to a superposition of the columns $\boldsymbol{A}_{:,i}$. In this specific setting only the columns $\boldsymbol{A}_{:,i}$ are weighted and added up that line-up with the non-zero elements of $\boldsymbol{x}$. The encoding procedure is shown in Figure 1.3. One point worth mentioning is that although the transformation $\boldsymbol{Ax}$ is linear, the code itself is not. The codewords are not closed under linear combination, since adding two codewords will not necessarily result in another codeword. Therefore SPARCs are non-linear codes.

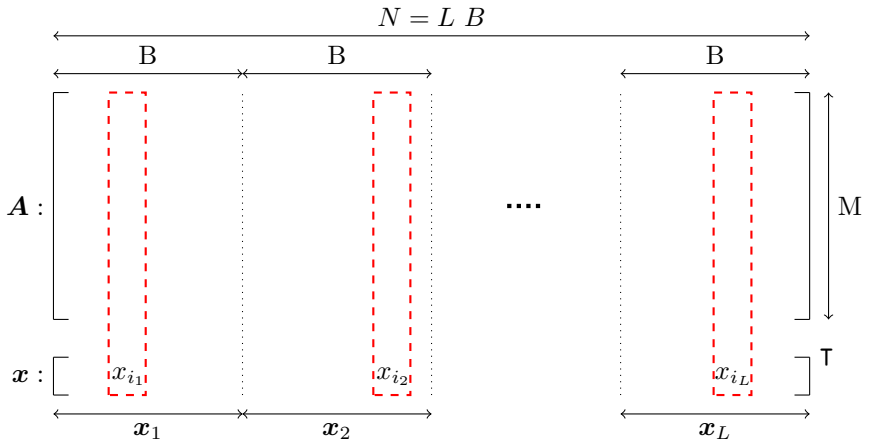


FIGURE 1.3: Illustrates the encoding procedure of SPARCs. The non-zero elements of each section are denoted by x_{i_j} for $j \in \{1, \dots, L\}$ and $i_j \in \{1, \dots, LB\}$. Which columns of the code matrix $\boldsymbol{A}$ are added up, is determined by the position of the non-zero coefficients in each section of $\boldsymbol{x}$. Only a single column of each section of the coding matrix $\boldsymbol{A}$ is part of the summation, that results in the codeword $\boldsymbol{c} = \boldsymbol{Ax}$.

When considering a coding scheme for transmission of information over a communication channel, two of the relevant parameters are the communication rate R, which defines the information sent per channel use, and the aforementioned block length M, that describes the number of channel uses. For the rate following holds: Since each section corresponds to $\log_2(B)$ bits and the input vector $\boldsymbol{x}$ consists of L sections, the number of input bits is

$K = L\ \log_2(B)$. Rewriting the rate $R = K/M$ as:

$$R = \frac{L\ \log_2(B)}{M} \quad \text{or} \quad M = \frac{L\ \log_2(B)}{R}\ . \tag{1.2}$$

According to (1.2) the code is specified by (R, B, M). First we will investigate how these quantities R, B, M are related, in an asymptotic limit.
There are different possibilities to choose L, B and still satisfy both (1.2) and produce an integer M. One extreme case is setting $L = 1$ and therefor $B = 2^{RM}$, which recovers the Shannon-style random codebook [1]. This lets the coding matrix $\boldsymbol{A}$ grow exponentially with the block length M making such choice impractical. Another option is to define $B = L^a$ with some constant a. This has the advantage that the number of elements in the coding matrix $\boldsymbol{A}$ grow polynomial with the block length M. This can be explained in the following fashion.

$$M = L\ \log_2(B)/R = aL\ \log_2(L)/R\,. \tag{1.3}$$

L grows asymptotically with M according to $M/\log_2(M)$. This can be shown by dividing L by $M/\log_2(M)$ and taking the limes $L \to \infty$. First we divide L and reformulate the fraction.

$$\frac{L}{M/\log_2(M)} = \frac{L\log_2(M)}{M}\ .$$

Inserting (1.3)

$$\frac{L\log_2(aL\ \log_2(L)/R)}{aL\ \log_2(L)/R} = \frac{R}{a}\,\frac{\log_2[aL\log_2(L)] - \log_2(R)}{\log_2(L)} = \frac{R}{a}\left\{\frac{\log_2(a)}{\log_2(L)} + 1 + \frac{\log_2[\log_2(L)]}{\log_2(L)} - \frac{\log_2(R)}{\log_2(L)}\right\}.$$

Now taking the limes $L \to \infty$

$$\lim_{L\to\infty} \frac{R}{a}\left\{\frac{\log_2(a)}{\log_2(L)} + 1 + \frac{\log_2[\log_2(L)]}{\log_2(L)} - \frac{\log_2(R)}{\log_2(L)}\right\} = \frac{R}{a}\,.$$

Since the result, of the division and letting L go to infinity, is a constant (converges), L grows asymptotically according to $M/\log_2(M)$. As a further consequence the number of elements of the coding matrix $\boldsymbol{A}$, which are MN, grow polynomial in M. Due to $MN = MLB = ML^{a+1}$ and therefore $\boldsymbol{A}$ grows with $M^{a+2}/\log_2(M)^{a+1}$. Relating the section size B with L has the advantage of keeping the dictionary size compact and manageable, which is crucial to achieve computationally efficient coding and decoding schemes. As a counter example assume that the number of sections L is fixed, the length of the input sequence N would increase exponentially with the block length, because $N = LB$ and reformulating (1.2) $B = 2^{MR/L}$ results in $N \propto 2^M$.

Another important parameter for SPARCs is the *power allocation* (PA), which defines the values of the power coefficients P_l with $l \in \{1, \ldots, L\}$. The power coefficients are related to the value of the non-zero coefficient of the l^{th} section $\boldsymbol{x}_l$ according to $x_{i_l} = \sqrt{MP_l}$, with i_l denoting the index of the non-zero coefficient. The overall power P of the codeword $\boldsymbol{c}$, that

is transmitted over the channel, fulfills the constraint $P = \sum_{l=1}^{L} P_l$ on average. First we will assume that each entry of the coding matrix a_{ij} is an i.i.d zero-mean normal distributed with variance $\sigma_{\boldsymbol{A}}^2$. The value of $\sigma_{\boldsymbol{A}}^2$ is set to $1/M$ so that the mean codeword power $\mathbb{E}\left[\|\boldsymbol{A}\boldsymbol{x}\|^2/M\right]$ is P, when averaged over all 2^K possible codewords.
The value for $\sigma_{\boldsymbol{A}}^2$ is derived as followed. Since each codeword element $[\boldsymbol{A}\boldsymbol{x}]_i$ is the superposition of certain matrix elements a_{ij} the elements of the codeword vector

$$[\boldsymbol{A}\boldsymbol{x}]_k = \sum\nolimits_{j=1}^{LB} [\boldsymbol{x}]_j \mathsf{a}_{kj} = \sum\nolimits_{l=1}^{L} \sqrt{MP_l}\mathsf{a}_{ki_l}, \quad k \in \{1 \dots M\} ,$$

are random variables $\sim \mathcal{N}(0, \sigma_{\boldsymbol{A}}^2 \sum_{l=1}^{L} \sqrt{MP_l})$.
Therefore the signal energy $\|\boldsymbol{A}\boldsymbol{x}\|^2 = \sum_{i=1}^{M} [\boldsymbol{A}\boldsymbol{x}]_i^2$ is a chi-squared distributed random variable with mean $M^2 P \sigma_{\boldsymbol{A}}^2$. By setting $\sigma_{\boldsymbol{A}}^2 = 1/M$ the mean power $\mathbb{E}\{\|\boldsymbol{A}\boldsymbol{x}\|^2/M\} = P$. This results in the signal to noise ratio $SNR = P/\sigma_W^2$ where σ_W^2 is the variance of the Gaussian noise. Both the set of power allocation coefficients $\{P_l\}$ and the coding matrix $\boldsymbol{A}$ are known to the receiver before the communication starts. Two examples for PA schemes are, 1) a flat power allocation with $P_l = P/L$ so every section has the same value assigned, 2) an exponentially decaying scheme. For the exponential scheme assume a fixed constant $\kappa > 0$ and the values for the power coefficients are calculated according to $P_l \propto 2^{-\kappa l/L}$, each for $l \in \{1, \dots, L\}$. The choice of the power allocation plays a crucial role for theoretical limits as well as for simulated error rates. To decrease computational complexity and memory demands, the Gaussian coding matrix $\boldsymbol{A}$ can be replaced with a Hadamard matrix, as will be discussed in later chapters.

1.2.1 Shaping Gain

To explain and illustrate an interesting phenomenon, we take a more detailed look at a capacity curve of 64 QAM in comparison to the Shannon capacity limit over the AWGN channel. Taking a look at the linear region of the modulation capacity plot in Figure 1.4, where the two plots are nearly parallel, one can see a constant offset between the curves. This offset is called *shaping gain* and it is defined as the difference in SNR to reach the same mutual information. The reason for the loss is the usage of equiprobable modulation signals, in other words using an uniform distribution over the signal set.

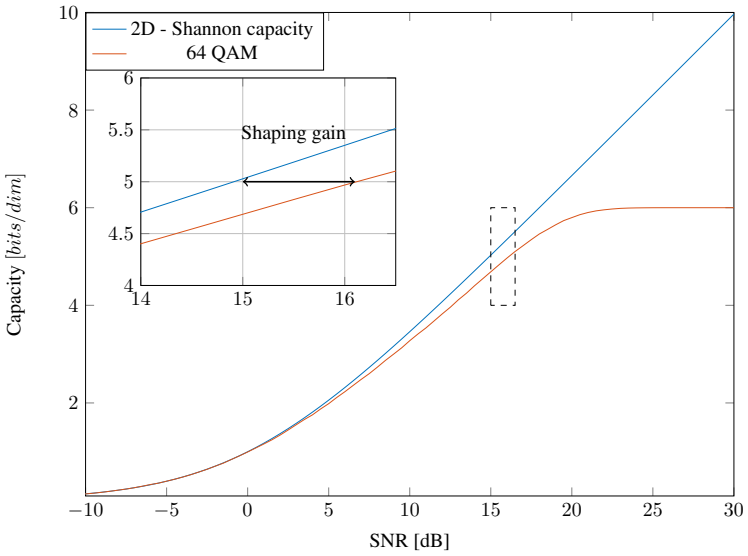


FIGURE 1.4: Illustrates the logical separation of coding and modulation in the coded modulation approach. The binary code could be for example an LDPC or Turbo code and the modulation format 4 or 16 QAM.

To overcome this loss, so called constellation *shaping techniques* are required. These shaping techniques assure a Gaussian similar distribution over the set of possible modulation signals. Several of these techniques are briefly explained in [2]. As one can imagine SPARCs do not encounter this problem, since the value of each codeword coefficient $[\boldsymbol{Ax}]_i$ will be Gaussian distributed, due to the construction scheme itself. So to say, shaping is already built into the design of SPARCs.

1.3 Related Literature

Barron and Joseph in [3] introduce sparse superposition codes for the additive white Gaussian noise channel and the authors show that these codes can come near the Shannon capacity with an iterative decoding scheme and proper scaling of the input coefficients, with an error probability decreasing exponentially with the length of the input sequence. Further in [4] they propose a scheme to combine an outer Reed-Solomon (RS) Code with an inner SPARC and provided an analytical description of the error exponent of the block error probability. According to [4] a composite code of RS and SPARC using a Least Squares Decoder achieves a block error probability that decreases exponentially with an exponent that is proportional to $M/\log(M)^2$ at a rate gap from capacity, that decreases with $1/\log(M)$. In a companion paper Barron and Joseph present a computational feasible decoder [5], unlike the before mentioned optimal least squares decoder. It is named adaptive successive decoder. Although the theoretical results are promising, the error rates in practical simulations are not. Further empirical improvement could be achieved by Barron and Cho in [6] with the iterative soft-decision decoder. The decoding scheme, that is a center piece of this thesis is named *approximate message passing* (AMP) algorithm. Generally AMP are a class of iterative algorithms that are based on an approximation of the so called loopy belief propagation

algorithm [7], which are used in the field of compressed sensing for example. The authors of [8] derive an AMP algorithm adapted to the setting of SPARCs, which outperforms the before mentioned decoding algorithms, in the sense of error rates at reasonable block lengths, while keeping the computational complexity at a polynomial order. In the same paper the authors prove that the AMP decoding scheme is asymptotically capacity-achieving for the AWGN channel. In a follow up paper [9] they derive a bound for the error exponent for the AMP decoder, given an exponential decaying power allocation and Gaussian coding matrix. They show that the mean section error rate $\mathcal{E}_{sec} = 1/L \sum_{i=1}^{L} \mathbb{I}\{\boldsymbol{x}_i \neq \hat{\boldsymbol{x}}_i\}$ decays exponentially in $M/[\log_2(M)]^{2T}$. With T the number of iterations that are required for successful decoding. For which they also find a proportionality of $T \propto \log_2(\mathcal{C}/R)$, given large block lengths.
Condo and Gross in [10] concern themselves with complexity estimation and variable quantization for a possible hardware implementation of the AMP decoder for SPARCs. They propose a partially parallel hardware decoder, since coding and decoding in SPARCs mainly involve matrix and vector multiplications. A fully parallel hardware architecture would not be feasible at present state of art. In [11] the same authors present a VHDL implementation for an encoder and decoder architecture. The survey paper [12] provides a broad view over the topic of superposition codes. It mentions the different decoding schemes and provides empirical simulation results. Further the authors elaborate on the possibility to use SPARCs for lossy compression coding.

Chapter 2

Communicating over an AWGN channel

2.1 Introduction

After establishing the process of encoding SPARCs, we will define the channel and investigate the decoder under test in a detailed manner.

- In Section 2.2 the AWGN channel is introduced, as well as the theoretical optimal decoder for this setting. Moreover a relation to the compressed sensing regime and our decoding problem is highlighted.
- In Section 2.3 we take a detailed look at the *approximate message passing* (AMP) decoder and visualize different aspects of the algorithm.
- Section 2.4 is dedicated to different methods of scaling the non-zero coefficients of the input vector, which can have a significant influence on the decoders performance.
- Further there exist different methods to reduce the complexity of the AMP decoder, which will be the topic of Section 2.5. Exchanging the Gaussian coding matrix with an Hadamard matrix can reduce the necessary computational calculations and save memory. Also an adaptation of the AMP decoder algorithm is presented that does not require to pre-calculate certain quantities.
- Finally in Section 2.6 we will inspect two different methods for estimating error rates, when applying the AMP decoder on the AWGN channel.

2.1.1 Notation

$\lfloor \cdot \rfloor$ denotes the rounding to the next smaller integer. $\mathbf{0}$ is a vector containing all zeros.

2.2 Transmission setting and decoding in general

2.2.1 Channel description

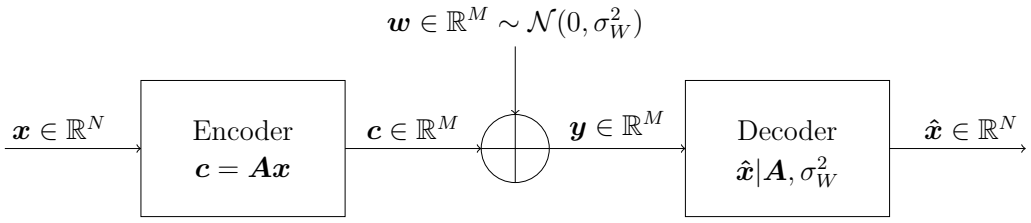


FIGURE 2.1: The input vector $\boldsymbol{x}$, fulfilling the constraint of having only one non-zero component per section, is fed into the encoder, where it is multiplied with the coding matrix. The codeword is then sent over an AWGN channel. The components of the noise vector $\boldsymbol{w}$ are i.i.d zero mean Gaussian random variables, with variance σ_W^2. The Decoder receives the codeword corrupted by noise and produces an estimate $\hat{\boldsymbol{x}}$ of the input vector, knowing the coding matrix and the noise variance.

One way of taking the physical realm between the transmitter and the receiver into account, is to apply a mathematical channel model, that is supposed to mimic the real world channel conditions. In this thesis the channel under consideration is assumed to be an ideal additive white Gaussian noise channel, which produces the output $\boldsymbol{y} \in \mathbb{R}^M$ according to

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w} \, . \tag{2.1}$$

The noise vector is denoted as $\boldsymbol{w} \in \mathbb{R}^M$ with coefficients that are i.i.d according to $[\boldsymbol{w}]_i \sim \mathcal{N}(0, \sigma_W^2)$. Additionally, the vector $\boldsymbol{w}$ is statistically independent of the transmitted codeword $\boldsymbol{c} = \boldsymbol{A}\boldsymbol{x}$. When considering the additive channel in (2.1), it is assumed that i) the attenuation between transmitter and receiver is known and accounted for ii) the input codeword $\boldsymbol{c}$ is neither filtered nor distorted by the channel and iii) delay and phase shifts are known and removed. The time domain is discrete and the ideal AWGN channel is memoryless.

The quality of the communication channel is characterized by the value of the SNR. As already mentioned in the previous Section 1.2, the mean input power of the codewords is constrained to P and the signal-to-noise ratio is therefore denoted as $SNR = P/\sigma_W^2$.

The infamous Shannon capacity [1] for the continuous valued, time discrete AWGN channel reads:

$$\mathcal{C} = \frac{1}{2}\log_2(1 + SNR) \quad \text{[bits/channel use]} \, , \tag{2.2}$$

and it provides an upper bound for the communication rate R, under which reliable communication is still possible. To summarize the importance of the capacity $\mathcal{C}$:

For $R < \mathcal{C}$ codes exist, such that the error probability can be made arbitrarily small with sufficiently large block lengths M, while for $R > \mathcal{C}$ this is impossible.

2.2.2 General decoding

We denote the set of possible input codewords as $\mathcal{X}$. When considering a Bayesian setting the mapping of the input sequence to the received senseword $\mathsf{x} \rightarrow \mathsf{y}$ can be described by the transition pdf $f_{\mathsf{y}|\mathsf{x}}(\boldsymbol{y}|\boldsymbol{x})$. The objective of the ideal decoder is to minimize the probability of a

block error $P\{\mathcal{E}\} = P\{\mathbf{x} \neq \hat{\mathbf{x}}\}$. This objective results in the so called maximum a posteriori (MAP) sequence decoder:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathcal{X}}{\arg\max}\, f_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x}|\boldsymbol{y}) \,. \tag{2.3}$$

For a uniform distribution of the possible input vectors $\boldsymbol{x} \in \mathcal{X}$ the MAP decoder is equal to the maximum likelihood decoder, which in the case of the memoryless AWGN channel reduces to finding the minimum distance between the received senseword $\boldsymbol{y}$ and all the possible codewords $\boldsymbol{Ax}$, i.e.,

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathcal{X}}{\arg\min} \|\boldsymbol{y} - \boldsymbol{Ax}\|^2 \,. \tag{2.4}$$

Since this requires an exhaustive search over all $\boldsymbol{x} \in \mathcal{X}$ it is computationally infeasible and one has to resort to other faster algorithms, which still provide a reasonable error probability.

2.2.3 Relation to compressed sensing

Given the sparsity constraint of the input sequences $\boldsymbol{x}$ and the dimensions of the coding matrix $\boldsymbol{A}$ there is an obvious relation to compressed sensing (CS). One CS problem is to recover a sparse vector $\boldsymbol{x} \in \mathbb{R}^N$ from the measurement $\boldsymbol{y} = \boldsymbol{Ax}+\boldsymbol{w}$. With $dim(\boldsymbol{A}) = n \times N$ and $n < N$ of the measurement matrix and $\boldsymbol{x}$ with a known prior distribution. $\boldsymbol{w}$ is an additive noise vector. An often used algorithm to solve such a problem is the *least absolute shrinkage and selection operator* commonly known as LASSO, which objective is

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathbb{R}^N}{\arg\min} \|\boldsymbol{y} - \boldsymbol{Ax}\|^2 \quad \text{subject to} \quad \|\boldsymbol{x}\|_1 < \tau \,, \tag{2.5}$$

for some parameter $\tau \geq 0$. The goal is to minimize the distance between $\boldsymbol{y}$ and $\boldsymbol{A}\hat{\boldsymbol{x}}$ under the constraint of recovering a sparse vector $\hat{\boldsymbol{x}}$. An efficient class of algorithms to solve the LASSO are the *approximate message passing* (AMP) algorithms, which are approximations of the loopy belief algorithm also known as sum-product or min-sum algorithm. Detailed insight on AMP can be found in [13], which is out of the scope of this thesis.

2.3 Approximate message passing decoder

The AMP for the general LASSO case should not be used in our setting, since the prior information on the structure of the input vectors $\boldsymbol{x}$ (one non-zero element per section) is not considered. A derivation for an AMP algorithm in the context of SPARCs can be found in [8]. The algorithm is iterative and generates successive estimates of the input vector $\boldsymbol{x}^t \in \mathbb{R}^N$ for $t = \{0, \ldots, t_{max} - 1\}$. The index t denotes the iteration and $\boldsymbol{x}^t$ can be interpreted as an estimate of the vector $\boldsymbol{x}$ at the iteration t. The decoder can be split into two parts: online and offline. The offline part has to be computed before running the AMP decoder. The quantities that are calculated beforehand do not change for fixed values of B, L, M. The offline computations are

$$\tau_0^2 = \sigma_W^2 + P, \quad \tau_{t+1}^2 = \sigma_W^2 + P(1 - \beta_{t+1}), \quad t \geq 0 \,, \tag{2.6}$$

with i.i.d random variables $\mathsf{U}_j^l \sim \mathcal{N}(0,1)$ for $j \in \{1,\dots,B\}, l \in \{1,\dots,L\}$ and

$$\beta_{t+1} = \sum_{l=1}^{L} \frac{P_l}{P} \mathbb{E}\left[\frac{\exp(\frac{\sqrt{MP_l}}{\tau_t}(\mathsf{U}_1^l + \frac{\sqrt{MP_l}}{\tau_t}))}{\exp(\frac{\sqrt{MP_l}}{\tau_t}(\mathsf{U}_1^l + \frac{\sqrt{MP_l}}{\tau_t})) + \sum_{j=2}^{B} \exp(\frac{\sqrt{MP_l}}{\tau_t}\mathsf{U}_j^l)} \right] . \tag{2.7}$$

The online part consists of the following computation, for $t = \{0,\dots,t_{max}-1\}$ and the initial conditions $\boldsymbol{x}^0 = \mathbf{0}$, $\boldsymbol{z}^{-1} = \mathbf{0}$, $\tau_{-1} = 0$ calculate:

$$\boldsymbol{z}^t = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^t + \frac{\boldsymbol{z}^{t-1}}{\tau_{t-1}^2}\Big(P - \frac{\|\boldsymbol{x}^t\|^2}{M}\Big) , \tag{2.8}$$

$$x_i^{t+1} = \eta_i^t(\boldsymbol{x}^t + \boldsymbol{A}^\intercal \boldsymbol{z}^t), \quad i = \{1,\dots,N\} , \tag{2.9}$$

with the component wise *tresholding* function:

$$\eta_i^t(\boldsymbol{s}) = \sqrt{MP_l} \frac{\exp(s_i \frac{\sqrt{MP_l}}{\tau_t^2})}{\sum_{j \in \boldsymbol{s}_l} \exp(s_j \frac{\sqrt{MP_l}}{\tau_t^2}))} . \tag{2.10}$$

As mentioned in Section 1.2, $\boldsymbol{s}_l$ denotes the B-dimensional section vector of $\boldsymbol{s}$. Therefore $j \in \boldsymbol{s}_l$ in (2.10) denotes the indices of the l^{th} B-dimensional section vector. Note that $\eta_i^t(\boldsymbol{s})$ only depends on the indices of the section containing the index i. The final estimate of the input vector $\hat{\boldsymbol{x}}$ results from setting the maximum entry of each section to $\sqrt{MP_l}$ and all other entries to zero. Have in mind, that the values of the non-zero entries $\sqrt{MP_l}$ are determined in advance (see Section 1.2).

2.3.1 Test statistics

First we will take a more detailed look at (2.9). The argument of the update equation (2.9) will further be denoted as *test statistics* $\boldsymbol{s}^t = \boldsymbol{x}^t + \boldsymbol{A}^\intercal \boldsymbol{z}^t$. A remarkable property of the test statistic $\boldsymbol{s}^t$ is the following: with increasing block length $M \to \infty$, the test statistic is asymptotically distributed as $\boldsymbol{s}^t \sim \boldsymbol{x} + \bar{\tau}_t \boldsymbol{v}$. The effective noise variance $\bar{\tau}_t$ is the limit of τ_t as $M \to \infty$ and $\boldsymbol{v}$ is a i.i.d random vector with elements $\sim \mathcal{N}(0,1)$, which is independent of the input sequence $\boldsymbol{x}$. In simple words, one of the main objectives of the AMP algorithm is to force the test statistic to be distributed close to $\boldsymbol{x} + \bar{\tau}_t \boldsymbol{v}$. Under the objective to minimize the MSE:

$$\underset{\boldsymbol{x}^{t+1} \in \mathbb{R}^N}{\arg\min} \|\boldsymbol{x} - \boldsymbol{x}^{t+1}\|^2 \quad \text{given :} \quad \boldsymbol{s}^t = \boldsymbol{x} + \tau_t \boldsymbol{v} , \tag{2.11}$$

the resulting *minimum mean square error* (MMSE) estimator equals the posterior expectation of $\boldsymbol{x}$:

$$\boldsymbol{x}^{t+1} = \mathbb{E}[\boldsymbol{x}|\boldsymbol{s}^t] . \tag{2.12}$$

Under the assumption that $\boldsymbol{s}^t = \boldsymbol{x} + \tau_t \boldsymbol{v}$ we can derive the MMSE estimator in (2.12). Following the steps in [8], we will come to the conclusion that the thresholding function $\eta_i^t(\boldsymbol{s})$ (2.10) corresponds to the MMSE estimator in (2.12):

We introduce the notation $i \in sec(l)$ where i is an index of the l^{th} section and therefore $i \in \{(l-1)B+1, \ldots, lB\}$. For each element of each section, $i \in sec(l)$ and $l \in \{1, \ldots, L\}$, we calculate the posterior expectation of the i^{th} element given the observation $\boldsymbol{s}^t$. For ease of notation we will set $\boldsymbol{s}^t = \boldsymbol{s}$, while keeping in mind that the test statistics vector $\boldsymbol{s}$ changes over iterations t.

$$\begin{aligned} x_i^{t+1}(\boldsymbol{s}) &= \mathbb{E}[x_i | \boldsymbol{x} + \tau_t \boldsymbol{v} = \boldsymbol{s}] \\ &\overset{(a)}{=} \mathbb{E}[x_i | \{x_j + \tau_t v_j = s_j\}_{j \in sec(l)}] = \mathbb{E}[x_i | \boldsymbol{s}_l] \\ &\overset{(b)}{=} \sqrt{MP_l}\, P(x_i = \sqrt{MP_l} | \boldsymbol{s}_l)\ . \end{aligned} \tag{2.13}$$

Consider that (2.13) (a) follows from the fact that for the i^{th} element only other elements of the same section l are statistically dependent. This holds under the assumption that the AMP decoder indeed achieves the required decoupling of different sections. Further (2.13) (b) is the result of the expectation, since only one element is non-zero in each section. Denoting the joint pdf of a section as $f_{sec}()$ and applying Bayes' theorem, (2.13) can be written as:

$$\begin{aligned} &\sqrt{MP_l}\, P(x_i = \sqrt{MP_l} | \boldsymbol{s}_l) \\ &= \sqrt{MP_l} \frac{f_{sec|x_i}(\boldsymbol{s}_l | \sqrt{MP_l}) P(x_i = \sqrt{MP_l})}{\sum_{k \in sec(l)} f_{sec|x_k}(\boldsymbol{s}_l | \sqrt{MP_l}) P(x_k = \sqrt{MP_l})}\ . \end{aligned} \tag{2.14}$$

Since $\boldsymbol{x}$ and $\boldsymbol{v}$ are statistically independent and $\boldsymbol{v}$ has i.i.d $\mathcal{N}(0,1)$ entries, the joint pdf of $\boldsymbol{s}_l$ conditioned on x_k, $k \in sec(l)$ being the non-zero component is formulated as:

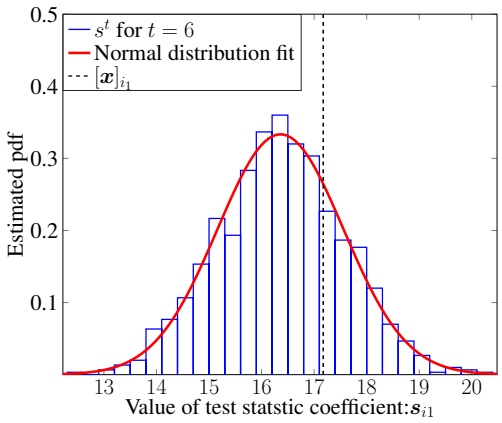
$$\begin{aligned} &f_{sec|x_k}(\boldsymbol{s}_l | \sqrt{MP_l}) \\ &= \left(\frac{1}{2\pi\tau_t^2}\right)^B \exp\left(\frac{-(s_k - \sqrt{MP_l})^2}{2\tau_t^2}\right) \prod_{j \in sec(l), j \neq k} \exp\left(\frac{-s_j^2}{2\tau_t^2}\right) \\ &= \left(\frac{1}{2\pi\tau_t^2}\right)^B \exp\left(\frac{s_k\sqrt{MP_l}}{\tau_t^2}\right) \exp\left(\frac{MP_l}{2\tau_t^2}\right) \prod_{j \in sec(l)} \exp\left(\frac{-s_j^2}{2\tau_t^2}\right)\ . \end{aligned} \tag{2.15}$$

The position of the non-zero element in one section is uniformly distributed over the whole section, therefor inserting $P(x_k = \sqrt{MP_l}) = 1/B$, $k \in sec(l)$ and (2.15) in (2.14) results in the equation (2.10) for the thresholding function $\eta_i^t(\boldsymbol{s})$. To summarize

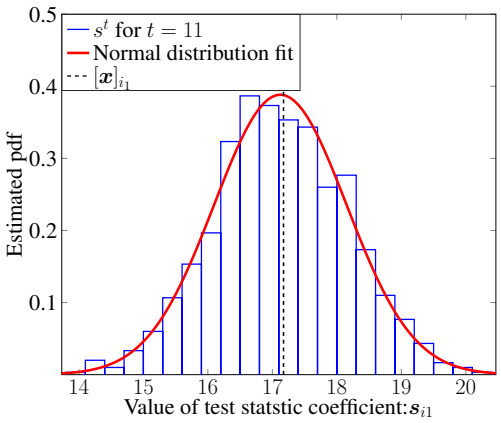
$$\boldsymbol{x}_i^{t+1} = \mathbb{E}[x_i | \boldsymbol{s}^t] = \eta_i^t(\boldsymbol{s})\ . \tag{2.16}$$

Figure 2.2 shows an example of the distribution of the test statistics vector $\boldsymbol{s}^t$ by calculating the histogram over 1000 runs. The histograms of two coefficients of $\boldsymbol{s}^t$ are displayed, where the first one is at the position of the non-zero coefficient in the first section of the input sequence $\boldsymbol{x}$. The second one is at the position of the non-zero coefficient in the last section. The non-zero coefficient of a section l is denoted as i_l with $l \in \{1, \ldots, L\}$. In the Subfigure 2.2a we can see that the test statistic coefficient in the first section $[\boldsymbol{s}^t]_{i_1}$ for $t = 6$ iterations is already closely distributed according to $\boldsymbol{s}^t \sim \boldsymbol{x} + \bar{\tau}_t \boldsymbol{v}$. Some further improvement is shown

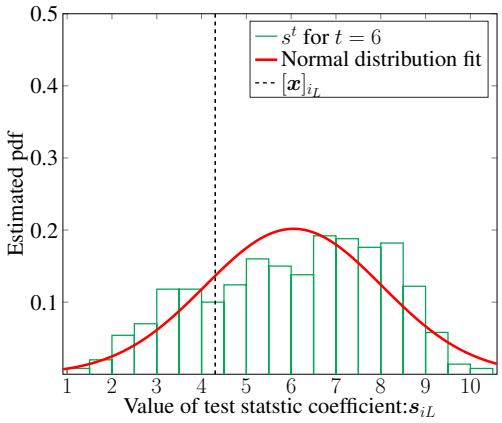
in Subfigure 2.2b, where for $t = 11$ the mean value is near the actual value of the input coefficient $[\boldsymbol{x}]_{i_1}$ and the variance of the fitted normal distribution is close to τ_{11}^2. In case of the last section at iteration $t = 6$ the histogram does not resemble a normal distribution and it takes more steps t until the wanted distribution is reached, as can be seen in the Subfigure 2.2c and 2.2d.



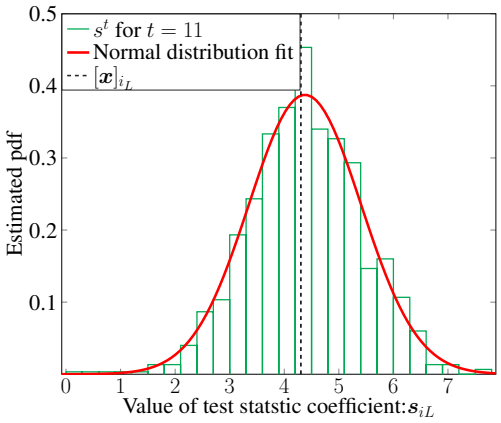
(A) At iteration $t = 6$, the test statistic coefficient at the first non-zero coefficient index. The variance of the fitted normal distribution is $\sigma^2 = 1.43$



(B) At iteration $t = 11$, the test statistic coefficient at the first non-zero coefficient index. The variance of the fitted normal distribution is $\sigma^2 = 1.05$



(C) At iteration $t = 6$, the test statistic coefficient at the last non-zero coefficient index. $\sigma^2 = 3.91$



(D) At iteration $t = 11$, the test statistic coefficient at the last non-zero coefficient index. $\sigma^2 = 1.05$

FIGURE 2.2: Simulation results evaluating the distribution of two different coefficients of the test statistic vector $\boldsymbol{s}^t$ at different iterations for 1000 runs with block length $M = 3413$, section length $B = 256$, number of sections $L = 512$, rate $R = 1.2$, $SNR = 15$ and $\sigma_W^2 = 1$. The histograms of the 1000 runs, are fitted with a normal distribution. At this specific example the value of the corresponding input coefficient of the first section is $[\boldsymbol{x}]_{i_1} = 17.173$ and of the last section $[\boldsymbol{x}]_{i_L} = 4.3049$.

Figure 2.2 indicates, that later sections, which are allocated less power than the first sections, need more iterations until the test statistic elements are actually distributed according $\boldsymbol{s}^t \sim \boldsymbol{x} + \bar{\tau}_t \boldsymbol{v}$ and the expectation operator outputs a reliable result. To further illustrate the decoding procedure, Figure 2.3 shows a different viewpoint of the AMP decoding scheme. Especially the fact, that later sections, with less power allocated, need more iterations to be decoded correctly.

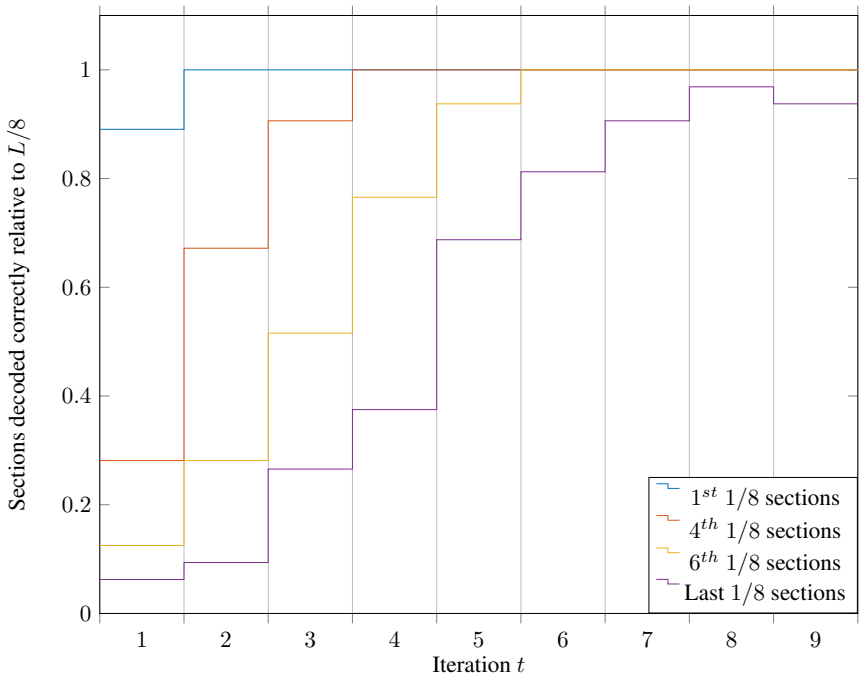


FIGURE 2.3: The simulation run is conducted with number of sections $L = 256$, section length $B = 512$, rate $R = 0.6C$, $SNR = 15$, $\sigma_W^2 = 1$ and exponential PA. The sections are grouped into 8 parts, beginning at the first sections. So the first $256/8$ sections are part of the first group. At each iteration it is evaluated, how many sections of each group could be decoded correctly, if the algorithm would be stopped. In the Figure the time evolution of the correctly decoded sections relative to the total sections of each group is displayed. In this simulation run, not all sections are decoded correctly. As can be seen, the errors occur at the last section group, where the least power is allocated to. Unfortunately at iteration $t = 9$ a section is wrongly decoded, that would have been correctly decoded at $t = 8$.

The last additive term on the RHS of (2.8) is called the *Onsager term* :

$$\frac{\boldsymbol{z}^{t-1}}{\tau_{t-1}^2}\left(P - \frac{\|\boldsymbol{x}^t\|^2}{M}\right). \tag{2.17}$$

and plays a crucial role in enforcing the test statistic to be distributed in the large system limit according to $\boldsymbol{s}^t \sim \boldsymbol{x} + \bar{\tau}_t \boldsymbol{v}$. Eliminating the Onsager term results in the basis for the *iterative soft-thresholding* (IST) algorithm, which is introduced for compressed sensing in [14]. A more detailed comparison of IST and AMP is given in [7], which can be used to gather intuition about the Onsager term. Further examples of the role of the Onsager term can be found in [13].

2.3.2 State evolution

Equations (2.6) and (2.7) are commonly know as *state evolution* (SE) in the CS regime. For large dimensions $N, M \to \infty$ the SE predicts several statistical properties of the AMP algorithm over the number of iterations t, as shown in [7]. To gather insight into the AMP algorithm and parameters of SE, we will take a look at a proposition stated in [8].

Proposition (1) *Under the assumption that* $\boldsymbol{s}^t = \boldsymbol{x} + \tau_t \boldsymbol{v}$*, where* $\boldsymbol{v}$ *is i.i.d.* $\sim \mathcal{N}(0,1)$ *and independent of* $\boldsymbol{x}$*, the quantity* β^{t+1} *defined in (2.7)*

$$\beta_t = \frac{1}{MP}\mathbb{E}[\boldsymbol{x}^\intercal \boldsymbol{x}^t], \quad 1-\beta_t = \frac{1}{MP}\mathbb{E}[\|\boldsymbol{x}-\boldsymbol{x}^t\|^2]\,. \tag{2.18}$$

and consequently $\tau_{t+1}^2 = \sigma_W^2 + 1/M\ \mathbb{E}[\|\boldsymbol{x}-\boldsymbol{x}^t\|^2]$.

Therefore the iteration variable β_t can be interpreted as the expected power-weighted fraction of correctly decoded sections at the iteration t. To put this in other words, β_t can be seen as the fraction of power that is correctly decoded at a given iteration t (in the large system limit). For illustrative purposes we can assign each input coefficient the same value, which means that the power over all sections is equally distributed. Applying the constant PA, we can use β_t to track the actual fraction of correctly decoded symbols at each iteration (instead of the power-weighted fraction). Because for a constant PA the fraction of the correctly decoded power, then resembles the fraction of correct sections, which can be seen in Figure 2.4.

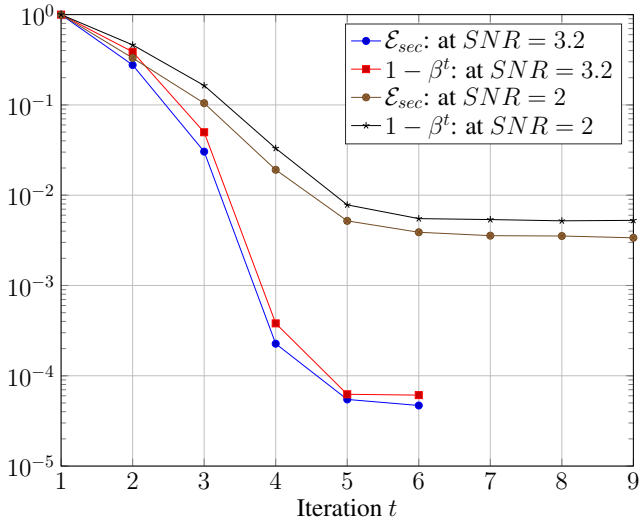


FIGURE 2.4: The simulation for $L = 256$ $B = 512$ and rate $R = 0.5$ at two different SNR values over 1000 runs. The encoder uses flat PA. Note that the number of necessary iterations decreases with the SNR.

The SE can also be used to predict the power weighted MSE of the AMP. For the large system limit the relations in (2.18) are exact, but not for finite sized codewords. Nevertheless simulation results show that the SE of $1-\beta_t$ matches the empirical found weighted MSE $\frac{1}{MP}\mathbb{E}[\|\boldsymbol{x}-\boldsymbol{x}^t\|^2]$ for reasonable block lengths. Also the SE can be a useful estimation for the expected power weighted fraction of correctly decoded sections at a given iteration t. This is shown in Figure 2.5.

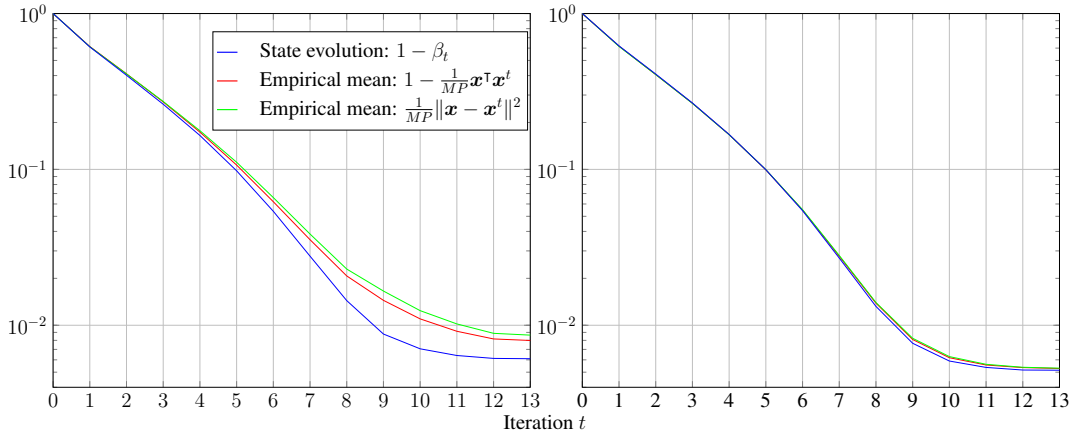


FIGURE 2.5: Simulation results compared to the analytical state evolution for two cases: 1) left Figure with block length $M = 1000$, section length $B = 256$ rate $R = 0.7C$ and 2) right figure with block length $M = 6582$, section length $B = 512$, number of sections $L = 1024$, rate $R = 0.7C$. Both at $SNR = 15$. The empirical mean is evaluated for 200 runs.

In Figure 2.5 it can be seen that the state evolution can be used to track the MSE, but only for increasing block lengths the state evolution gets indistinguishable from the empirical MSE. Since we assume that $\boldsymbol{s}^t = \boldsymbol{x} + \tau_t \boldsymbol{v}$, the empirical variance of the test statistics vector $var(\boldsymbol{s}^t - \boldsymbol{x})$ can be compared to the pre-calculated τ_t^2. To gather further intuition about the SE parameters this comparison is depicted in Figure 2.6.

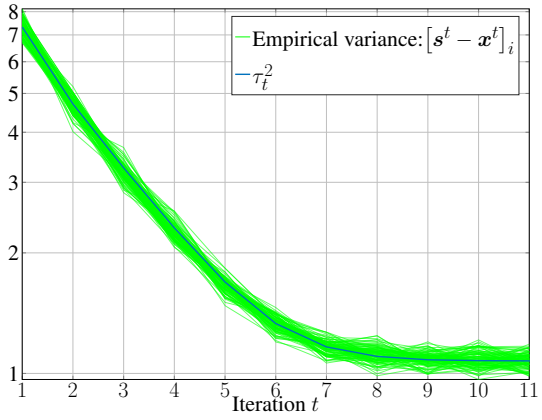


FIGURE 2.6: The empirical variance for 100 different coefficients of the vector $\boldsymbol{s}^t - \boldsymbol{x}^t$, where the choice of coefficients i is equally spread over the $B \cdot L$ possible coefficients. One can see that the variances of $[\boldsymbol{s}^t - \boldsymbol{x}^t]_i$ track τ_t^2 very closely. For this example $L = 256$, $B = 512$, $R = 1$ $SNR = 6.3$. 200 simulation runs are conducted over which the empirical variance is calculated.

2.4 Power Allocation

In this Section we will take a look at some different possibilities to scale the non-zero coefficients of the input vector $\boldsymbol{x}$. As already indicated in Section 1.2, the allocation of power to each section of the input word $\boldsymbol{x}$ plays an important factor in the performance of SPARCs for finite block lengths.

2.4.1 Constant power allocation

The most simple scaling is the constant power allocation, where each non-zero coefficient has the same value assigned.

$$P_l = P\,\frac{1}{L}, \quad l \in \{1, \dots, L\}\,. \tag{2.19}$$

In later Sections we will conduct empirical simulations to compare the error performances of different PA schemes. For most of the simulated scenarios, the SNR is set such that $R/\mathcal{C} > 0.5$. Therefore the upcoming remarks are valid for a SNR region close to the Shannon Limit. The simulations show that the AMP decoder with a constant value across each section results in a good error performance at rates $R < 1$. At higher rates error performance of the constant PA decreases drastically. It seems that at approximately $R > 1$ more power is needed at the initial sections to kick start the AMP algorithm, by decreasing the effective AMP noise τ_t for the decoding of later sections. Among others this constant power allocation is researched in [4], with an adapative successive decoding scheme.

2.4.2 Exponentially decaying power allocation

Next we consider an exponentially decaying power allocation for which the section power $P_l \propto 2^{-\kappa l/L}$. For evaluating the performance of the AMP decoder we will choose the following PA in specific:

$$P_l = P\,\frac{2^{2C/L} - 1}{1 - 2^{-2C}}\,2^{-2Cl/L}, \quad l \in \{1, \dots, L\}\,, \tag{2.20}$$

where $\kappa = 2C$ with the Shannon capacity according to (2.2) and a normalization factor, such that $P = \sum_{l=1}^{L} P_l$. In [8] it is shown, that with this PA and AMP decoding, the SPARCs asymptotically reach the AWGN capacity with growing block length M. Figure 2.7 shows that for the exponentially decaying PA section errors occur most likely at sections that are allocated the least power to. At this depicted setting, no decoding errors happen at lower sections, indicating that more power than necessary could be allocated to the first coefficients of $\boldsymbol{x}$.

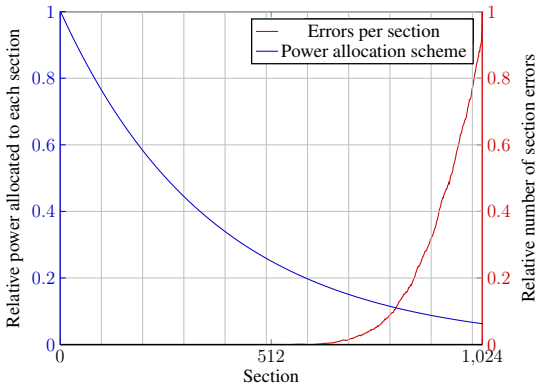


FIGURE 2.7: The number of incorrect decodings with the power allocated to each section. The values associated with both axis are relative to the maximum of the found data. The numerical evaluation is conducted for 1000 runs, with the number of sections $L = 1024$, section size $B = 512$, at rate $R = 1.5$ and $SNR = 15$.

Although the theoretical findings are promising, at practical block lengths the exponential PA is outperformed by other PA schemes.

2.4.3 Algorithmic power allocation

Next we will introduce another PA scheme, which is based on a Lemma derived in [8], with lim denoting the large system limit $N, M, L \to \infty$:

Lemma 1. *For any power allocation* $\{P_l\}, l \in \{1, \dots, L\}$ *that is non-increasing with l, we have*

$$\bar{\beta}(\tau) := lim\ \beta(\tau) = lim \sum_{l=1}^{\lfloor \eta^*(\tau) L \rfloor} \frac{P_l}{P}\,, \tag{2.21}$$

where $\eta^*(\tau)$ *is the supremum of all* $\eta \in (0, 1]$ *that satisfy*

$$lim\ LP_{\lfloor \eta L \rfloor} > 2ln(2)R\tau^2\,. \tag{2.22}$$

If $lim\ LP_{\lfloor \eta L \rfloor} < 2ln(2)R\tau^2$ *for all* $\eta > 0$*, then* $\bar{\beta}(\tau) = 0$.

As already stated β_{t+1} represents the expected power-weighted fraction of correctly decoded sections at the iteration $t + 1$. Therefore, Lemma 1 states that in the large system limit sections l up to $\lfloor \eta^*(\bar{\tau}_t) L \rfloor$ can be decoded correctly at a given iteration $t + 1$ and AMP noise $\bar{\tau}_t$. All sections that do not satisfy the condition 2.22, meaning $l > \lfloor \eta^*(\bar{\tau}_t) L \rfloor$ are not correctly decoded at step $t + 1$. We can use Lemma 1 to derive a PA scheme, which was first introduced in [15] and denoted as *algorithmic power allocation*. First we fix the number of maximal AMP iterations to T^* and we plan to decode L/T^* sections at each iteration. So for the first fraction of sections we determine asymptotically the value of $\{P_l\}$ for $l \in \{1, \dots, L/T^*\}$ that is necessary to be able to decode the L/T^* fractions and iteratively continue to the next part of the sections $\{P_l\}$ for $l \in \{L/T^* + 1, \dots, 2\ L/T^*\}$. This procedure can be described as follows, $t = \{0, \dots, T^* - 1\}$, tolerance value $\delta > 0$ and with:

$$\tau_0^2 = \sigma_W^2 + P, \quad \beta_0 = 0, \quad \tilde{L}_t = \frac{L}{T^*} t + 1\,. \tag{2.23}$$

At each step t calculate:

$$P_{th} = \frac{2ln(2)R\tau_t^2}{L} + \delta \ , \tag{2.24}$$

$$P_{rem} = \frac{P - \sum_{l=1}^{\tilde{L}_t} P_l}{L - \tilde{L}_t} \ , \tag{2.25}$$

$$P_l = max(P_{th}, P_{rem}), \quad for\ \tilde{L}_t < l \leq \tilde{L}_{t+1} \ , \tag{2.26}$$

$$\beta_{t+1} = \sum_{l=1}^{\tilde{L}_{t+1}} \frac{P_l}{P}, \quad \tau_{t+1}^2 = \sigma_W^2 + P(1 - \beta_{t+1}) \ . \tag{2.27}$$

Especially at the later sections/iterations the value of P_{th} (2.24) can be lower than if the remaining power would be distributed equally over the remaining sections in (2.25). Therefore at each iteration the value of P_{th} is compared to P_{rem}. The greater quantity is chosen as power value for the sections $\tilde{L}_t < l \leq \tilde{L}_{t+1}$ at iteration t, in (2.26). The tolerance value δ in (2.24) guarantees that the inequality (2.22) in Lemma 1 is fulfilled. Figure 2.8 gives a graphical explanation of the algorithmic PA.

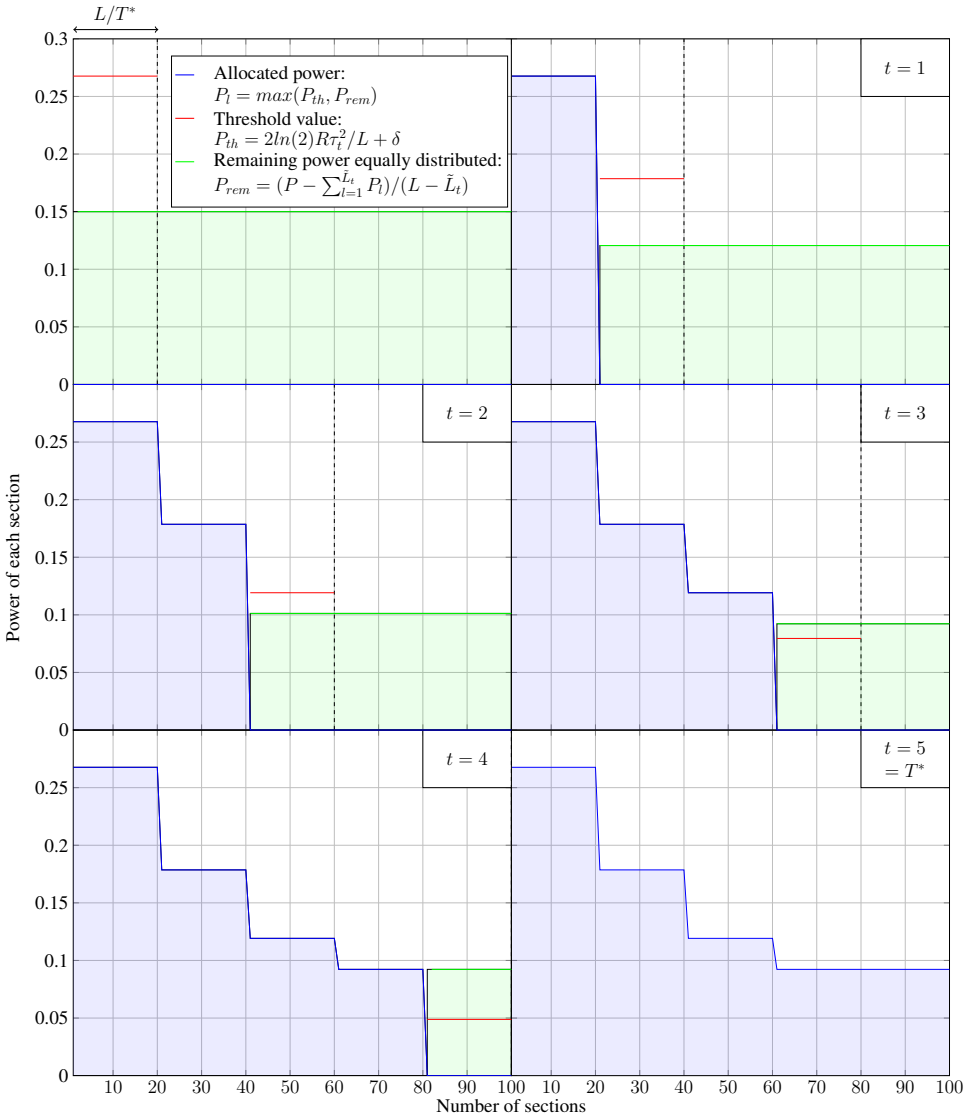


FIGURE 2.8: An example of the algorithmic PA, for the number of sections $L = 100, T^* = 5, P = 15, \sigma_W^2 = 1$. Note that at iteration $t = 3$ the green horizontal line P_{rem} is greater than the red threshold value P_{th} and therefore the sections under consideration get appointed the value of $P_l = P_{rem},\ for\ \tilde{L}_3 < l \leq \tilde{L}_4$. Once $P_{th} < P_{rem}$ the remaining sections can be set to P_{rem} since P_{th} is monotonically decreasing with t.

Figure 2.8 is an explanatory example, where as Figure 2.9 compares the two beforehand mentioned PA schemes under a more practical setting.

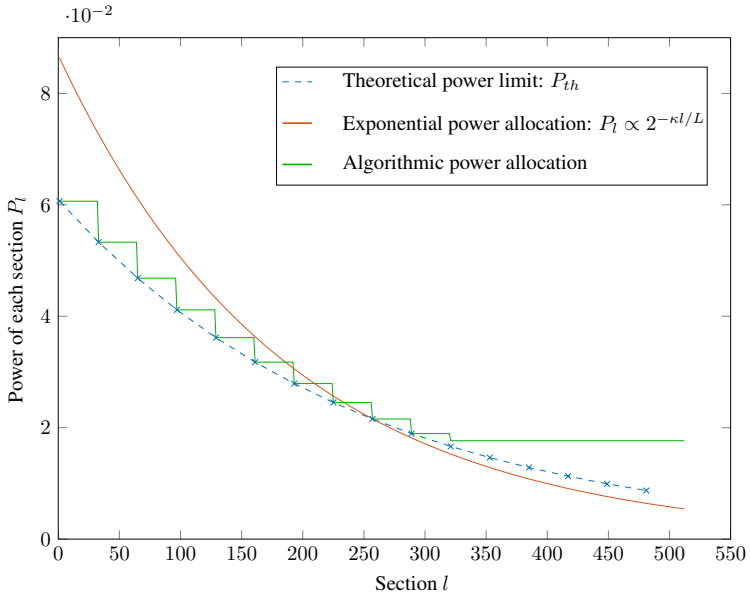


FIGURE 2.9: Comparison between the exponential PA $P_l \propto 2^{-\kappa l/L}$ and the iterative algorithmic PA with the number of sections $L = 512$, rate $R = 1.4$, $SNR = 15$ and number of steps $T^* = 16$. The dashed line shows the threshold value P_{th} calculated as in (2.24).

Lemma 1 is stated for the large system limit $M, L, B \to \infty$ but can be used as a guideline for finite block lengths. To be able to modify the set of PA values P_l, we set $\delta = 0$ and introduce an additional parameter $R_{PA} \geq 0$. The parameter R_{PA} is used to compute a PA rate R', which is only used for calculating the PA values and deviates from the actual communication rate. This is done in the following fashion: i) select some R_{PA}, for example 1.1, and multiply it with the rate R to calculate $R' = R_{PA} \cdot R$. ii) compute the section power P_l, $l \in \{1, \ldots, L\}$ according to (2.24) - (2.27), while exchanging R with the modified rate R'.

Note that the actual communication is carried on at rate R. Figure 2.10 illustrates the effect of choosing a different rate for the power allocation $R' \neq R$. As one can see, choosing $R_{PA} > 1$ results in an increase of power at the initial sections and a reduction of power at the later sections, in comparison to $R = R'$. For $R_{PA} < 1$ the power at the lower sections is decreased and the values at the higher sections are increased.

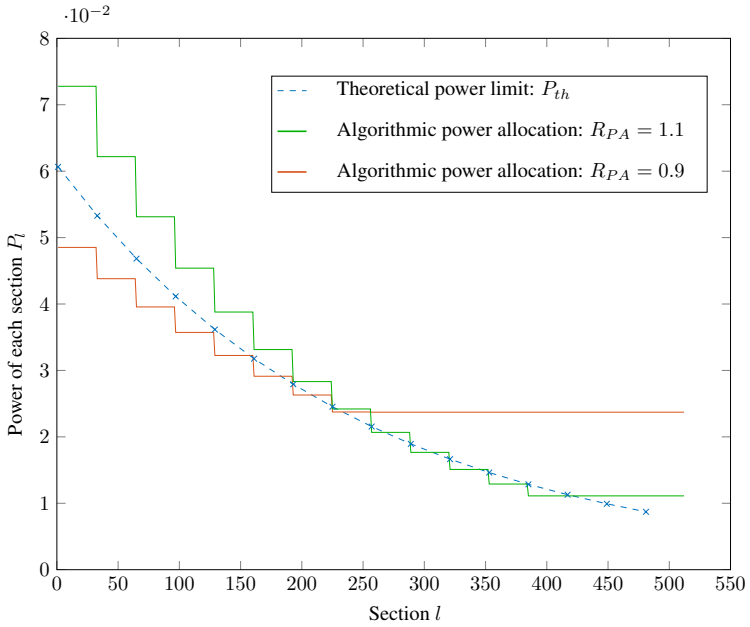


FIGURE 2.10: Different PA at $R_{PA} = 1.1$ and $R_{PA} = 0.9$ and the theoretical power threshold P_{th} for parameters: number of sections $L = 512$, rate $R = 1.4$, $SNR = 15$ and number of steps $T^* = 16$. For $R_{PA} = 1.1$ more power is allocated to the initial sections at the cost of having less power for the later sections and especially the flat sections at the end. For $R_{PA} = 0.9$ the reverse effect occurs.

An additional benefit of the algorithmic PA is, that the β_t and τ_t computed as in (2.27) can be used in the AMP algorithm in (2.8) and (2.9), without the Monte-Carlo pre-computations in (2.6) and (2.7).

The number of iterations T^* for the PA, on the one hand defines the granularity of how many sections get assigned the same value of power and on the other hand, it sets how often the "online" part of the AMP ((2.8) and (2.9)) iterates. Ideally one wants to choose the number of steps in the PA equal to the number of sections $T^* = L$, which would cause the AMP run time to prolong. To overcome this, [16] introduced an estimation of the "offline" AMP values, that can be computed during run-time of the "online" decoding. This has the advantage that one can choose T^* of the PA independently of the actual AMP iterations.

2.4.4 Choice of the power allocation rate R_{PA}

A theoretical analysis of the optimal R_{PA} for finite length codes and minimal error performance has not been undertaken at this point. The value of R_{PA} has a drastic influence on the section error rate for a given code setting, which can be seen in Figure (2.11), where the simulated mean section error rate varies up to a decade with different values for R_{PA}.

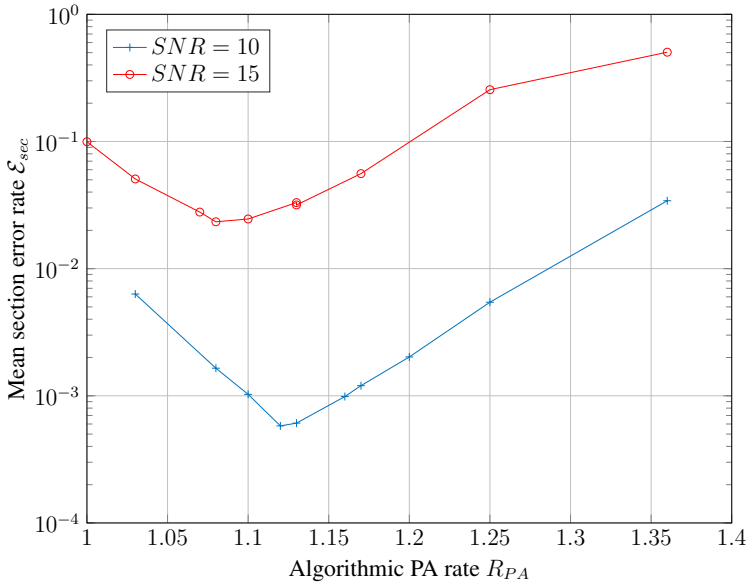


FIGURE 2.11: Two examples for the dependence of the section error rate on the value of R_{PA}. The number of sections L is equal to the section length B, i.e. $B = L = 256$. The communication rate is is fixed to $R = 1.5$.

To find a rough optimum for R_{PA} and a specific setting $\{B, L, R, SNR\}$, we apply a so called bisection algorithm [17] to the region of the possible optimum R_{PA}. In Figure 2.11 we see, that the section error rate has a steep descent when approaching the region of the minimal R_{PA} value. When increasing R_{PA} further past the minimum value, the mean section error rate shows a positive gradient. This can be explained by observing the distribution of the number of section errors. As an illustrative example we take a look at three distributions at different R_{PA} values.

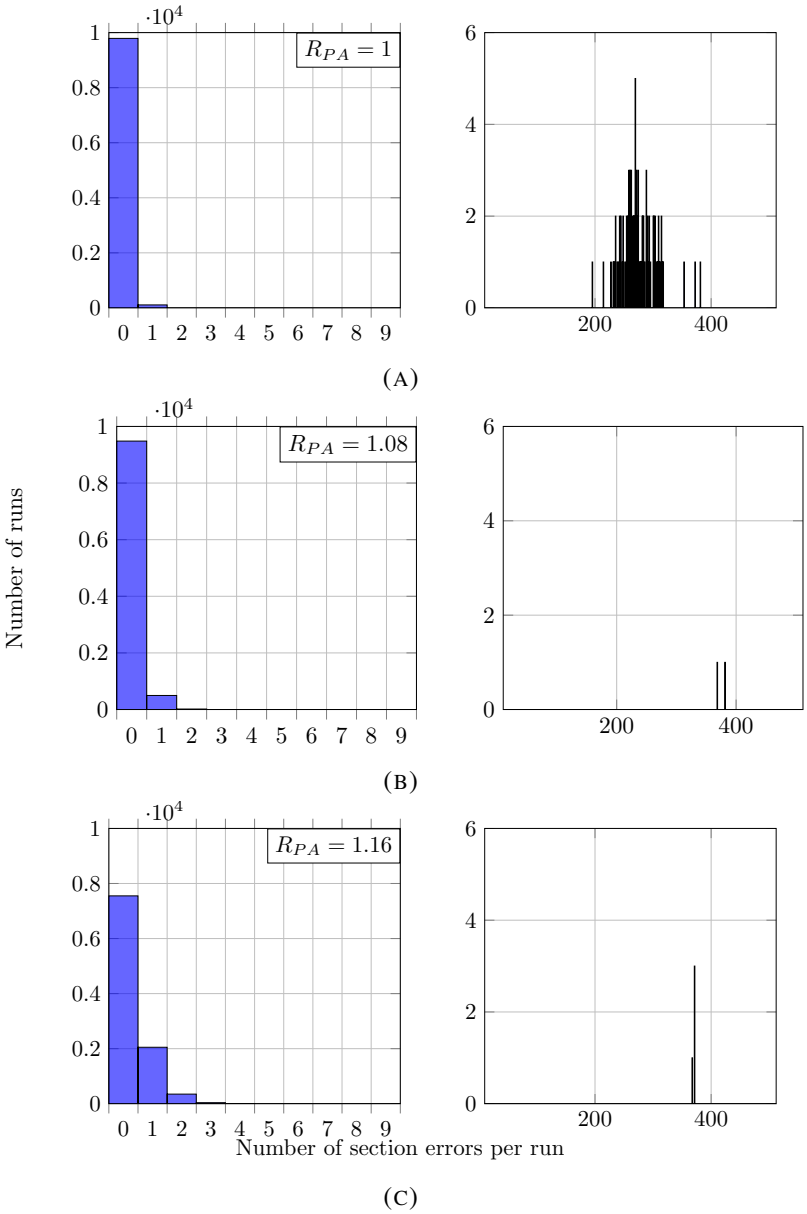


FIGURE 2.12: Histogram for the number of section errors with $L, B = 512$, $SNR = 15$, $R = 1.5 = 0.7\mathcal{C}$ and 10^4 simulation runs. The histogram is split into two regions, one with the number of runs that resulted in a low number of section errors (≤ 10) (left) and the second one with more than ten section errors (right). For the lowest R_{PA} value of the three scenarios, see Figure 2.12a, the error distribution is very different to the distributions of the higher R_{PA} values.

First we compare $R_{PA} = 1$, in Figure 2.12a, with the scenario of an higher $R_{PA} = 1.08$ value, which found to show the lowest section error rate (Figure 2.12b). In Figure 2.12a it is more likely for a simulation run to have an high error event. In this setting an high error event is denoted as a simulation run with more than 10 section errors. It is noteworthy that the section error rate for $R_{PA} = 1$ is the highest of the three compared scenarios, but the codeword error rate is the lowest, since it shows the greatest number of error-free trials. In general $R_{PA} > 1$ allocates more power to the initial sections compared to the theoretical optimal value and decreases the power at later sections. Increasing R_{PA} has the following effect on the distribution of the number of section errors over simulation runs: The number

of runs with a low section error count increases, while the number of runs with an high count of section errors decreases (see Figure 2.12b). At the R_{PA} value with the lowest observed section error rate, we see a very low chance of an high error event. By further increasing R_{PA} the number of high error events hardly change, but since a lower amount of power is allocated to the last sections, the low error count increases (see Figure 2.12c). This explains the positive gradient of the mean section error rate after passing the region of the optimal R_{PA} value.

2.5 Complexity

2.5.1 Online state evolution estimation

The so-called state evolution parameters τ_t and β_t, which are described in the Sections 2.3 and 2.3.2, are part of the AMP decoding scheme and can be used to estimate the (power-weighted) MSE of the decoding process. For a short review we take a look at the first mentioned approach, which computes the SE parameters according to (2.6) and (2.7). This calculation has to be performed before the communication starts and requires lengthy Monte-Carlo simulation, or other methods, to numerically evaluate the expectation. This can be avoided, when the algorithmic PA scheme is used and τ_t, β_t are calculated as in (2.27). Another possibility, that was introduced in [16], is to estimate the SE parameters while running the online part of the AMP decoder. At each iteration of the AMP compute the following equation, with z^t as in (2.8).

$$\hat{\tau_t}^2 = \frac{\|z^t\|^2}{M} \,. \tag{2.28}$$

This estimation is based on a part of Lemma 5(e) in [8], which concludes that (2.28) holds true with high probability under the large system limit. Including (2.28) in the online computation only adds low computational complexity of order $\mathcal{O}(M)$ but has several advantages. When using the algorithmic PA the number of steps T^* is independent of the online AMP iterations and can therefore be chosen to be equal to the number of sections L. The change of the estimates $\hat{\tau_t}^2$ can be used as a stopping criterion. If the difference of $\hat{\tau_t}^2$ between iterations is smaller than a certain threshold ϵ, $|\hat{\tau_t}^2 - \hat{\tau}_{t-1}^2| < \epsilon$, the AMP decoder can be stopped. The simulated section error rates using the estimated $\hat{\tau_t}^2$ according to (2.28) are close and even lower at specific rates R than when using the SE parameters calculated by other methods. This can be seen in Figure 2.13, which shows the mean section error rate over the rate R, a plot reproduced from literature [16], but it includes additional information of the PA parameters, R_{PA}, that are used.

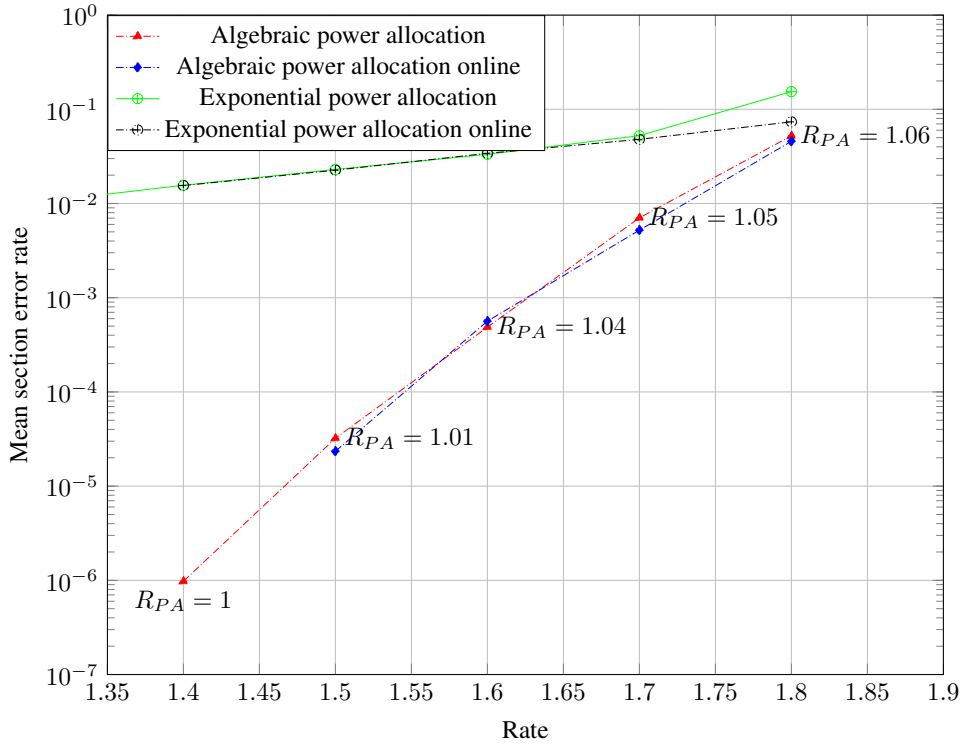


FIGURE 2.13: Mean section error rate plotted against the rate R, for $L = 1024$, $B = 512$ at $SNR = 15$ averaged over 1000 runs. The block length decreases linearly with the rate $1/R$ and is in between $[6582, 5120]$. In the above Figure two power schemes are compared, the exponentially decaying and the algorithmic PA. As one can see, the algorithmic PA scheme outperforms the former. At each simulated point the rate for the algorithmic PA R_{PA} is given, see Section 2.4.3.

Further we can compare the estimated SE parameter $\hat{\tau}_t^2$ with the pre-calculated values of τ_t^2 according to (2.6) and see in Figure 2.14, that $\hat{\tau}_t^2$ is close to τ_t^2.

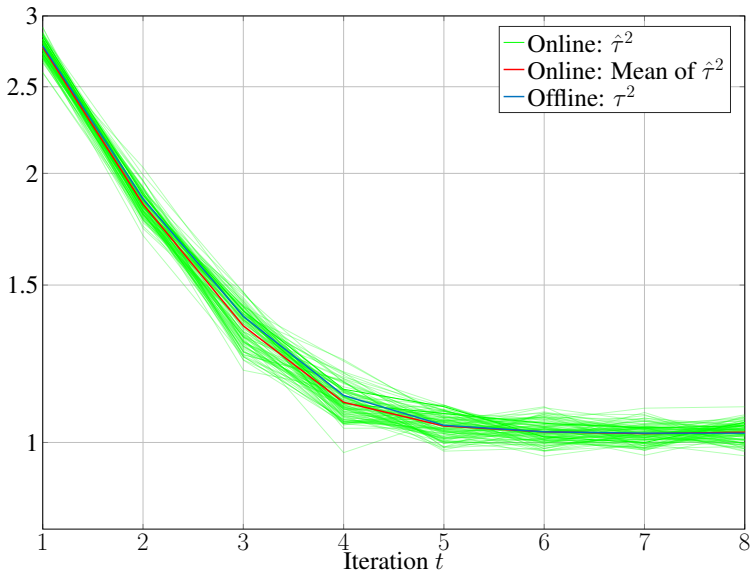


FIGURE 2.14: 100 runs for $\hat{\tau}_t^2 = \|z^t\|^2/M$ plotted against the iterations. Further displayed are the mean of $\hat{\tau}_t^2$ for 100 runs and the pre-calculated τ_t^2. For this plot the number of sections $L = 256$, the section length $B = 512$, rate $R = 0.5$, $SNR = 1.77$ and algorithmic PA.

As described in Section 2.3.2 the SE parameters can be used to track the progress of the AMP decoding algorithm. In specific we can track the (power-weighted) MSE of the estimated $\boldsymbol{x}^t$ compared to the actual input sequence $\boldsymbol{x}$, as well as the (power-weighted) fraction of sections that are decoded correctly at a given iteration t. In Figure 2.15 the before mentioned quantities for the AMP algorithm, implemented with online SE estimation, are plotted. At the initial iterations, we can see that the MSE and the fraction of correctly decoded symbols differ from the corresponding SE parameters, while at later steps the different plots converge.

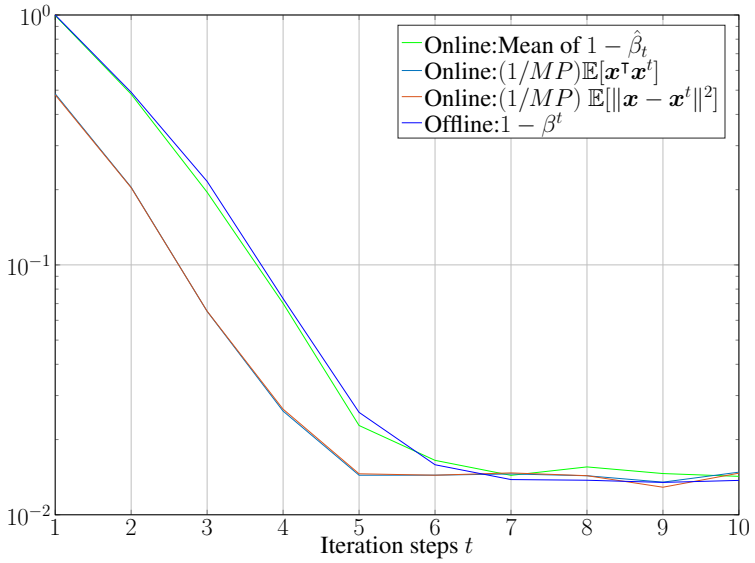


FIGURE 2.15: The mean values are evaluated over 200 runs. The simulation parameters are set to $L = 256$, $B = 512$, $R = 0.5$ $SNR = 1.77$.

2.5.2 Hadamard coding matrix

One of the main disadvantages of using a Gaussian random coding matrix $\boldsymbol{A}$ is the need to store each of the MN elements. This causes a bottleneck in memory, when one wants to use larger block lengths with limited memory resources. As an intuitive example, given the section length $B = 512$, number of sections $L = 1024$ and rate $R = 1.2$ results in a block length of $M = 7680$ (see (1.2)). At these medium size block lengths one needs to store $MN \approx 4{\cdot}10^9$ elements. Further, the decoder's computational complexity is dominated by the matrix-vector multiplications $\boldsymbol{A}\boldsymbol{x}^t$, $\boldsymbol{A}^\intercal \boldsymbol{z}^t$ in (2.9) and (2.8). In general, these matrix-vector multiplications are of complexity $\mathcal{O}(MN)$, the running time of the remaining operations is $\mathcal{O}(N)$. To speed up the decoding process and relax the memory requirements, we will consider *structured random matrices*, which are also common in compressed sensing. These structured matrices are generated by a random choice of a number of parameters, which are much smaller than all the MN elements of a Gaussian random matrix. In specific we will consider *Hadamard* matrices, which can be constructed recursively, with $dim(\boldsymbol{H}_m) = 2^m \times 2^m$ and $H_0 = 1$, as:

$$\boldsymbol{H}_m = \begin{bmatrix} \boldsymbol{H}_{m-1} & \boldsymbol{H}_{m-1} \\ \boldsymbol{H}_{m-1} & -\boldsymbol{H}_{m-1} \end{bmatrix}. \tag{2.29}$$

The coding matrix then would be constructed by 1) Setting $m = log_2(N)$ resulting in an $N \times N$ matrix $\boldsymbol{H}_m$. 2) Randomly choose M rows of $\boldsymbol{H}_m$, except the first row which consist of all ones, and stack them to a coding matrix $\boldsymbol{A}$. Each row has equal probability to be chosen as a candidate for $\boldsymbol{A}$ 3) Normalize each column vector to norm 1 by multiplying each element with $1/\sqrt{M}$. The codeword then would be obtained by a matrix-vector multiplication $\boldsymbol{c} = \boldsymbol{A}\boldsymbol{x}$. This alone will not reduce the computational complexity, but the use of the *Walsh-Hadamard Transform* (WHT) will, for which there is a "fast" algorithm similar to the Cooley-Tukey algorithm used for the *Fast Fourier Transform* (FFT). Instead of saving the whole coding matrix $\boldsymbol{A}$, the M indices of the randomly chosen rows are kept and further denoted as the set $\mathcal{S}_M$. So instead of calculating the matrix-vector product $\boldsymbol{A}\boldsymbol{x}$, one would compute the length-N WHT of $\boldsymbol{x}$ and only consider the indices given by $\mathcal{S}_M$. For calculating $\boldsymbol{A}^\intercal \boldsymbol{z}^t$ with the WHT the vector $\boldsymbol{z}^t \in \mathbb{R}^M$ has to be extended to $\tilde{\boldsymbol{z}}^t \in \mathbb{R}^N$, where the M coefficients of $[\tilde{\boldsymbol{z}}^t]_i, i \in \mathcal{S}_M$ are set to the coefficients of $\boldsymbol{z}^t$ and the remaining elements to zero. Since the Hadamard matrix is symmetric:

$$WHT(\tilde{\boldsymbol{z}}^t) = \boldsymbol{H}_m \tilde{\boldsymbol{z}}^t = \boldsymbol{H}_m^\intercal \tilde{\boldsymbol{z}}^t = \sum\nolimits_{i \in \mathcal{S}_M} [\boldsymbol{H}_m]_{i:} \tilde{z}_i^t = \sum\nolimits_{i=1}^{M} [\boldsymbol{A}]_{i:} z_i^t = \boldsymbol{A}^\intercal \boldsymbol{z}^t \,. \tag{2.30}$$

For the AMP decoder it is only necessary to store the indices $\mathcal{S}_M$ making the memory requirements $\mathcal{O}(M)$. The WHT reduces the complexity of the matrix-vector multiplication to $\mathcal{O}(Nlog_2N)$ (see [18]).

2.6 Estimating error rates

In this Section we will take a look at different possibilities to estimate the section error rate $\mathcal{E}_{sec}$ for a certain choice of code and channel parameters, without conducting empirical simulations. The authors of [8] suggest that the expectation of the correctly decoded sections after the iteration $t+1$ can be calculated as

$$v_{t+1} := \sum\nolimits_{l=1}^{L} \frac{1}{L} \mathbb{E}\left[\frac{\exp(\frac{\sqrt{MP_l}}{\tau_t}(\mathsf{U}_1^l + \frac{\sqrt{MP_l}}{\tau_t}))}{\exp(\frac{\sqrt{MP_l}}{\tau_t}(\mathsf{U}_1^l + \frac{\sqrt{MP_l}}{\tau_t})) + \sum_{j=2}^{B} \exp(\frac{\sqrt{MP_l}}{\tau_t}\mathsf{U}_j^l)} \right], \tag{2.31}$$

with i.i.d random variables $\mathsf{U}_j^l \sim \mathcal{N}(0,1)$ for $j \in \{1,\dots,B\}, l \in \{1,\dots,L\}$. The derivation is similar to the one of the state evolution parameter β_t, but in contrast to β_t (see Subsection 2.3.2), which estimates the expectation of the power weighted correctly decoded section after $t+1$, v_t can be used to predict the expected section error rate at a certain iteration. With the number of iterations T^*, the expected section error rate at the end of the AMP decoding is then v_{T^*}.

$$\mathcal{E}_{sec} = v_{T^*} \,. \tag{2.32}$$

Another method to predict the section error rate and the codeword error rate is proposed in [16].

Proposition 1([16]). *Let the power allocation $\{P_l\}$ be such that the state evolution iteration using the asymptotic approximation converges to $\tau_T^2 = \sigma_N^2$. Under the assumption that $\boldsymbol{x}^T + \boldsymbol{A} * \boldsymbol{z}^T = \boldsymbol{x} + \tau_T \boldsymbol{Z}$ (where $\mathbf{Z}$ is a standard normal random vector independent of $\boldsymbol{\beta}$), the finite length section error rate and codeword error rate are given by*

$$\bar{\mathcal{E}}_{sec} = 1 - \frac{1}{L} \sum\nolimits_{l=1}^{L} \mathbb{E}\left[\Phi\left(\frac{\sqrt{MP_l}}{\sigma} + \mathsf{U} \right) \right]^{B-1}, \tag{2.33}$$

$$\bar{\mathcal{E}}_{cw} = 1 - \frac{1}{L} \prod\nolimits_{l=1}^{L} \mathbb{E}\left[\Phi\left(\frac{\sqrt{MP_l}}{\sigma} + \mathsf{U} \right) \right]^{B-1}. \tag{2.34}$$

In both expressions above, U is a standard normal random variable, and $\Phi(.)$ is the standard normal cumulative distribution function.

In Figure 2.16 numerical results for practical block lengths are compared with the before mentioned estimating methods. The predicted section error rates are very close to the empirical results. Both variants ((2.31) and (2.33)) provide similar error rate predictions. Complexity wise the prediction according to (2.33) is less demanding, since only one random variable is in use.

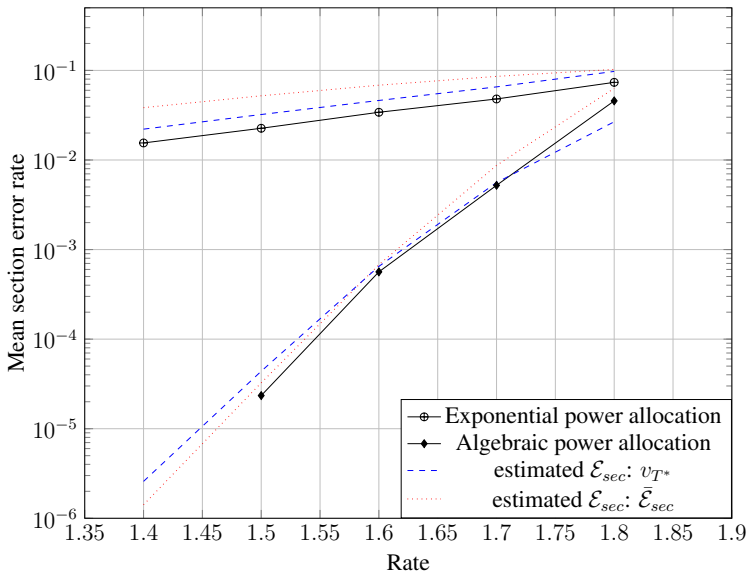


FIGURE 2.16: A Comparison of the error section prediction methods to empirical results. The mean section error rate is plotted against the rate R, for $L = 1024$, $B = 512$ at $SNR = 15$ averaged over 1000 runs. The block length is proportional to $1/R$ and is in between $[6582, 5120]$. In the above Figure two power schemes are compared, the exponential decaying PA and the algorithmic. The expectation operator of (2.31) and (2.33) is numerically evaluated via Monte-Carlo method for 1000 runs.

Chapter 3

Numerical evaluation

3.1 Introduction

Now that we have established the necessary building blocks of our AMP decoding scheme, we will numerically evaluate the reliability and further aspects like the number of iterations and the decay of the error rate with increasing block lengths.

- In Section 3.2 the number of necessary iterations for the AMP algorithm under a given setting is investigated. We introduce a theoretical limit for the necessary iterations of the AMP algorithm from recent literature and compare it with empirically found values.
- In Section 3.3 we will look into the decay of the error rate when increasing the block length M. Again theoretical and simulated results are compared.
- Since M is defined by $L,\ B$ and the rate R, the question of how to choose L and B at a fixed rate R, arises. In Section 3.4 the error performance for different L/B ratios is evaluated and an heuristic guideline is provided.
- In Section 3.5 numerous simulations are compactly presented to provide a deeper understanding of the influence of the PA's parameters on the (section) error rate.
- Next we compare the bit error rate of SPARCs with different parameters to the performance of state of the art coded modulation schemes.

For the numerical evaluation we use Hadamard coding matrices with the algorithmic PA and online SE estimation.

3.1.1 The bit error probability and E_b/N_o

When wanting to compare our SPARCs encoding and decoding scheme for different code parameters, we have to define a measure which describes how good an algorithm is, and later evaluate them for that measure. One intuitive performance measure, is the section error rate:

$$\mathcal{E}_{sec} = 1/L \sum_{i=1}^{L} \mathbb{I}\{\boldsymbol{x}_i \neq \hat{\boldsymbol{x}}_i\} \ . \tag{3.1}$$

Although the $\mathcal{E}_{sec}$ comes in handy, we want to put SPARCs in context with more conventional coding approaches. The bit error rate $\mathcal{E}_b$ is often used in coding related literature and scientific work to measure the reliability of a coding scheme. $\mathcal{E}_b$ is the ratio of incorrectly decoded information bits over the number of total information bits sent. Another common measure is the block error probability $\mathcal{E}_{block} = \mathbb{E}\{\boldsymbol{x} \neq \hat{\boldsymbol{x}}\}$. For SPARCs, Monte Carlo simulations show the relation between the bit error rate and the section error rate:

$$\mathcal{E}_b \approx \frac{1}{2}\mathcal{E}_{sec} \,. \tag{3.2}$$

The reason for this dependence is the following: For simplicity we will consider one specific section i of the input vector $\boldsymbol{x}$. The non-zero coefficient of the i^{th} section can be at B different locations. We fix the sent index to $m \in \{1, \ldots, B\}$. We assume that if a section error occurs at section i, any other section index $\hat{m} \neq m$ can be the outcome of our decoder with equal probability. An example of this circumstance is shown in Figure 3.1.

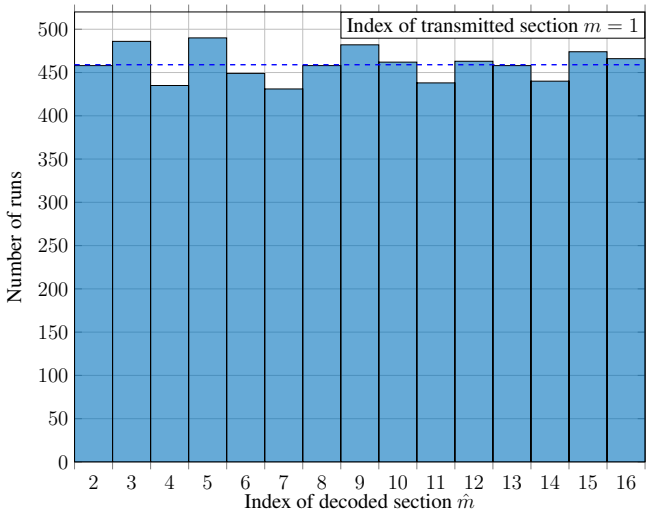


FIGURE 3.1: The distribution of decoded section indices given that a section error occurred. The sent index is fixed to $m = 1$. The code parameters are $L = 64$, $B = 16$ and $R = 1$. The dashed blue line indicates the total number of errors divided by $B - 1$.

For ease of notation we denote the total number of bits per section as $l = \log_2(B)$. We denote the number of incorrect bits given that a section error occurred as $N \in \{1, \ldots, l\}$. In general there are $\binom{l}{n}$ possible combinations of how n bits of total l bits can be in error. Each of these combinations correspond to a wrongly decoded section index $\hat{m} \neq m$, which are all equally probable with probability $1/(B-1)$. The expectation of N is

$$\mathbb{E}[N] = \mu_N = \frac{1}{B-1}\sum\nolimits_{n=1}^{l} \binom{l}{n} n \,. \tag{3.3}$$

Rewriting the sum in (3.3) and applying the binomial theorem

$$\begin{aligned}\sum_{n=1}^{l}\binom{l}{n}n &= \sum_{n=1}^{l}\frac{l!}{n!(l-n)!}n = \sum_{n=1}^{l}\frac{l!}{(n-1)!(l-n)!}\\ &= l\sum_{n=1}^{l}\frac{(l-1)!}{(n-1)!(l-n)!} = l\sum_{k=0}^{l-1}\frac{(l-1)!}{(k)![(l-1)-k]!}\\ &= l\sum_{k=0}^{l-1}\binom{l-1}{k}1^{(l-1)-k}1^k = l(1+1)^{l-1} = l2^{l-1} = l\frac{B}{2}\,.\end{aligned}$$

Therefore the expected number of erroneous bits given that a section error occurred is

$$\mu_N = \frac{l}{2}\frac{B}{B-1}\,. \tag{3.4}$$

The bit error probability $\mathcal{E}_b$ is the expected number of total erroneous bits $\mathcal{E}_{sec}\cdot L\cdot\mu_N$ divided by the number of total transmitted bits $L\cdot l$, i.e.,

$$\mathcal{E}_b = \frac{\mu_N}{l}\mathcal{E}_{sec} = \frac{1}{2}\frac{B}{B-1}\mathcal{E}_{sec}\,. \tag{3.5}$$

For large values of B the bit error rate $\mathcal{E}_b$ can be approximated in terms of the section error rate $\mathcal{E}_{sec}$ as in (3.2). So when trying to minimize the section error rate also the bit error rate is minimized. In general this is not the case for the block error probability $\mathcal{E}_{block}$.
A common method for comparing the performance of different digital modulation and coding schemes, is established by calculating and visualizing the dependence of the bit error probability $\mathcal{E}_b$ on the transmit energy per bit to noise power ratio E_b/N_o. Previously we defined the signal-to-noise ratio as $SNR = P/\sigma_W^2$. Now we want to reformulate the SNR in terms of the ratio E_b/N_o. For SPARCs the average power of a codeword is constrained to P. Therefore the average energy of a codeword equals $P\,M$. Since we transmit $L\log_2(B)$ bits, the energy per bit equals $E_b = P\,M/(L\log_2(B)) = P/R$. With the power spectral density of the noise $N_o/2 = \sigma_W^2$ (we previously denoted our noise vector as $\boldsymbol{w}$) the ratio E_b/N_o reads as

$$\frac{E_b}{N_o} = \frac{P}{\sigma_W^2\,2R} = \frac{SNR}{2R}\,. \tag{3.6}$$

3.2 Number of AMP iterations

For the AMP with an exponentially decaying PA the authors of [9] derive an upper bound for the number of AMP iterations T. The upper bound defines the number of iterations that are necessary to confine the variance τ_T^2 of the AMP noise at step T in a close interval near the variance of the channel noise σ_W^2. At this point the algorithm can be stopped. We will first review the findings of [9] and then proceed to take empirical measures to determine the number of AMP iterations under different settings for the algorithmic power allocation and "online" SE estimation (see Subsection 2.5.1). With a fixed constant $c\in(0,1/2)$ let

$$f(B) = 2\exp\left(-\frac{1}{2}\left[\ln(B)\right]^{1-2c}\right)\,,\quad \delta_B = 3[\ln(B)]^{-c}\,, \tag{3.7}$$

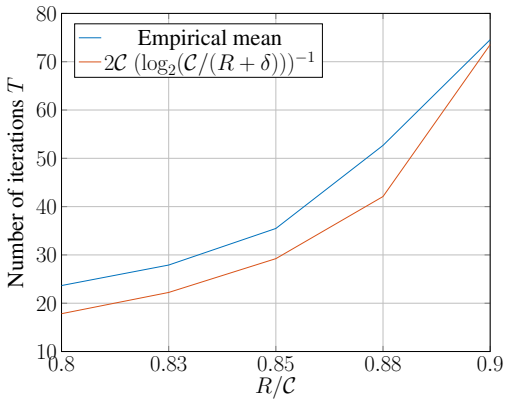
and

$$\kappa = \frac{(1+SNR)^{L+1/L}}{\mathcal{C}\ \ln\ 2}\ , \tag{3.8}$$

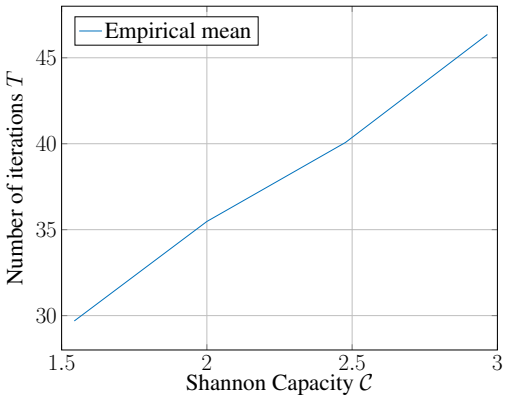
then with

$$\begin{aligned} \lceil \Delta_\xi^{-1} \rceil &\leq \lceil \left(\frac{1}{2\mathcal{C}} \log_2 \left(\frac{\mathcal{C}}{R(1+\delta_B/2)} \right) - \frac{1}{L} - \kappa f(B) \right)^{-1} \rceil \\ &\overset{(a)}{\approx} 2\mathcal{C} \left[\log_2 \left(\frac{\mathcal{C}}{R(1+\delta_B/2)} \right) \right]^{-1}\ , \end{aligned} \tag{3.9}$$

run the AMP decoder for T iterations, where $T := 1 + \lceil \Delta_\xi^{-1} \rceil$.
This guarantees that the variance τ_T^2 of the test statistic at iteration T is in the interval $[\sigma_W^2, \sigma_W^2 + \tau_0^2 f(B)]$. This is derived for the exponentially decaying PA and under the condition that the gap to the capacity is $\mathcal{C} - R \geq \delta_B$. Note that the approximation (a) of (3.9) gains accuracy as B and L grow larger. For the complete derivation see [9].
To gain insight into the number of necessary AMP iterations and the dependence of T on the values of $\mathcal{C}$ and $R/\mathcal{C}$ empirical simulations are visualized in Figure 3.2. These simulations are conducted for the algorithmic PA with "online" SE estimation. We can see that, as for the exponentially decaying PA, the number of iterations T shows an approximately linear increase with $\mathcal{C}$, when keeping the ratio $R/\mathcal{C}$ constant. Keeping the SNR and therefore the $\mathcal{C}$ constant and increasing R towards the capacity $\mathcal{C}$, the number of iterations T is proportional to $(\log_2(\mathcal{C}/R))^{-1}$.



(A) The empirical mean of the necessary AMP iterations for fixed SNR compared to the theoretical approximation.



(B) The empirical mean of the necessary AMP iterations for a constant ratio of $R/\mathcal{C}$ and increasing SNR. One can observe a linear increase in AMP iterations.

FIGURE 3.2: The AMP iteration's dependence on parameters $\mathcal{C}$ and R, where the mean is calculated over 200 runs and is in accordance with theoretical findings. For these simulations the "online" AMP algorithm is used, with the stopping criterion as $|\hat{\tau}_T^2 - \hat{\tau}_{T-1}^2| < (P_L \cdot 10^{-3})$

To put all this into perspective and deliver an intuition about the range of the iterations needed, we further display the mean number of steps resulting from empirical simulations for different rates and SNR values.

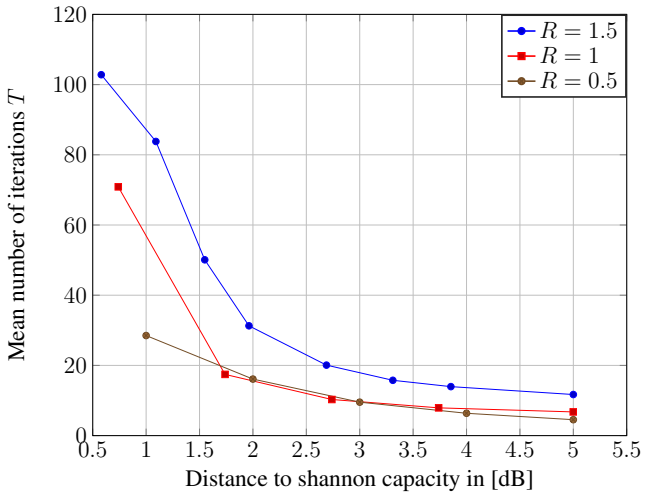


FIGURE 3.3: Mean number of iterations for 1000 runs evaluated for $L = 256$, $B = 512$ and over increasing SNR values. Here the SNR is given as the gap to capacity in dB. As stated by theory, the number of iterations depend strongly on $\mathcal{C}$ and in turn on the SNR.

As can be seen in Figure 3.3 the AMP iterations for our simulation settings are in the region of 10^1 to 10^2. As the gap to capacity increases, the plots for the different rate settings converge to a value in the range of 10. In the next section a large deviation bound is presented, for which the influence of R and $\mathcal{C}$ is represented only by the number of iterations T. Therefore a deepened understanding of T for the AMP configuration under test, can help to explain different error distributions at different SNR and R values.

3.3 Error exponent

In this Section we will look at how fast the decay of the section error rate takes place, when increasing the block length M, while keeping the SNR and the rate R constant. At first some theoretical findings and probability bounds from [9] are given and then a numerical evaluation is presented. The authors of [9] derived a large deviations bound for the section error rate of the AMP decoder given an exponentially decaying PA. The theorem gives an upper bound for the probability of deviation of the section error rate $\mathcal{E}_{sec}$ from an arbitrary value ϵ. With a fixed constant $c \in (0, 1/2)$ let

Theorem 1. *([9]) Fix any rate $R < \mathcal{C}$. Consider a rate R SPARC $\mathcal{S}_M$ with block length M, design matrix parameters L, B determined according to (1.2) and an exponentially decaying power allocation given by (2.20). Let $\epsilon > \frac{4(1+SNR)}{\ln(1+SNR)} f(B)$, where f(B) is defined as (3.7). Then the section error rate of the AMP decoder satisfies:*

$$P(\mathcal{E}_{sec}(\mathcal{S}_M) > \epsilon) \leq K_T \exp\left\{ \frac{-k_T L}{(\log B)^{2T-1}} \left(\frac{\epsilon \sigma^2 \ln(1+SNR)}{4} - \tau_0^2 f(B) \right)^2 \right\}, \quad (3.10)$$

where T as defined in Section 3.2. (The constant c required to define T and f(B) can be arbitrarily chosen in the interval (0, $1/2$), but must be the same for both quantities.)

T is the number of iterations, that are necessary for the AMP algorithm to reach its convergence in the sense, that the AMP noise τ_T^2 is close to the channel noise σ_w^2. For the exponentially decaying PA $T \propto \log(\frac{\mathcal{C}}{R})^{-1}$. Therefore T increases as R increases towards $\mathcal{C}$. κ_T and K_T are constants that solely depend on T, but are not explicitly specified. Theorem 1 can be used to state a bound for the decay of the section error rate with increasing block length M. We set ϵ and the rate $R < \mathcal{C}$ constant and let L, B and M grow large. The value of $f(B)$ (3.7) goes to zero as B increases, and the term $\epsilon\sigma^2 \ln(1 + SNR)$ stays constant. Therefore the bound in (3.10) decays with growing $L,\ B$ if $k_T L > (\log B)^{2T-1}$. As stated in Section 3.2, for large L and B the number of necessary AMP iterations T only depends on the ratio $\mathcal{C}/R$ (3.9). Hence T can be treated as a constant in the scenario of evaluating the error exponent for large $B,\ L$ and M. By choosing $B = L^a$ both L and B grow with $M/\log(M)$ and as a consequence an upper bound for the section error rate decay is an exponential function in $\frac{M}{\log(M)^{2T}}$. This can be used to describe the further asymptotic results for SPARCs with AMP decoding. For this we will revise the Borel-Cantelli Lemma, which states:

Lemma 2. *(Borel-Cantelli Lemma) Suppose that* $\{A_m\ :\ m \geq 1\}$ *is a sequence of events in a probability space. If:*

$$\sum_{m=1}^{\infty} P(A_m) \leq \infty\ , \tag{3.11}$$

then $\lim_{m\to\infty} P(A_m) = 0$

To apply the Borel-Cantelli Lemma to SPARCs, we set A_m equal to $\mathcal{S}_M$, with block length M, under the conditions stated above. From Theorem 1 and the considerations concerning the decay of the error exponent, it is clear that $\sum_{M=1}^{\infty} P(\mathcal{E}_{sec}(\mathcal{S}_M) > \epsilon) \leq \infty$. By using the Borel-Cantelli Lemma, therefore

$$\lim_{M\to\infty} P(\mathcal{E}_{sec}(\mathcal{S}_M) \geq \epsilon) = 0\ . \tag{3.12}$$

For any $\epsilon > 0$, $R < \mathcal{C}$ and $L = B^a$ as before. This states that SPARCs for the given setting are actually capacity achieving in the large system limit. An equivalent formulation can be found in [8]. Now that we have revised promising theoretical results from literature, we conduct empirical simulations over increasing block lengths.

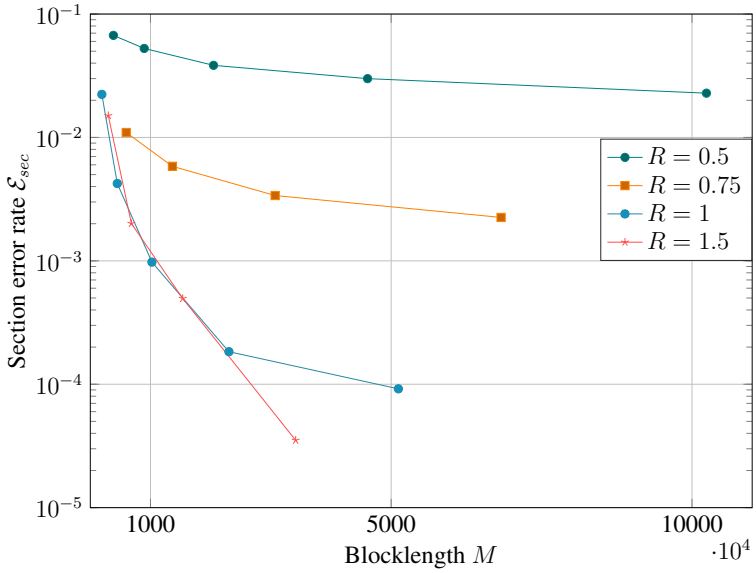


FIGURE 3.4: The mean section error rate of 1000 runs is plotted against the block length M for sets of $B = 2L$ and $L \in \{2^5, \ldots, 2^8\}$. For all four plots the SNR is set such that $R/\mathcal{C} = 0.75$. Note that the block length M depends on the rate by $\propto 1/R$. This is the reason for the longer block lengths at lower rates.

Figure 3.4 displays a dependence of the reliability of SPARCs with AMP decoding on the chosen rate R. We can see for our lower rate settings $R = 0.5$ and $R = 0.75$, the error decays only slightly with increasing M. We can conclude that lower rate settings, e.g. $R = 0.5$, need a lower $R/\mathcal{C}$ ratio than higher rate settings, e.g. $R = 1$, to exhibit a steep error rate decay. In Figure 3.9 and Figure 3.10 we can observe that the decay of error rate with increasing parameters L, B, M and fixed R is dependent on the underlying SNR and therefore dependent on the $R/\mathcal{C}$ ratio.

3.4 Choosing L and B

In practical case scenarios the maximum block length applicable is limited due to delay constraints. In this Section we will investigate the dependence of the section error rate on the choice of code parameters B and L under fixed SNR and rate R. As a reminder the length of the codeword depends on L, B and R in the following manner $M = L \log_2(B)/R$. Therefore certain values for M can be achieved by different settings of L, B and R, for example both codes $(1) : S_{512} = \{L = 128, B = 16, R = 1\}$ and $(2) : S_{512} = \{L = 64, B = 256, R = 1\}$ result in the same block length M. Now the question arises of how to choose between different possibilities. The most intuitive would be to decide for a code setting, that experiences the lowest experimental $\mathcal{E}_{sec}$. In Figure 3.5 we can observe, that for a fixed number of sections L and varying section length B the section error rate $\mathcal{E}_{sec}$ significantly decreases up to a point where $B = L$ and then approximately stays the same. Therefore the block length increases with $\log_2(B)$ and the complexity with $\mathcal{O}(B)$, but the section error rate $\mathcal{E}_{sec}$ hardly decreases.

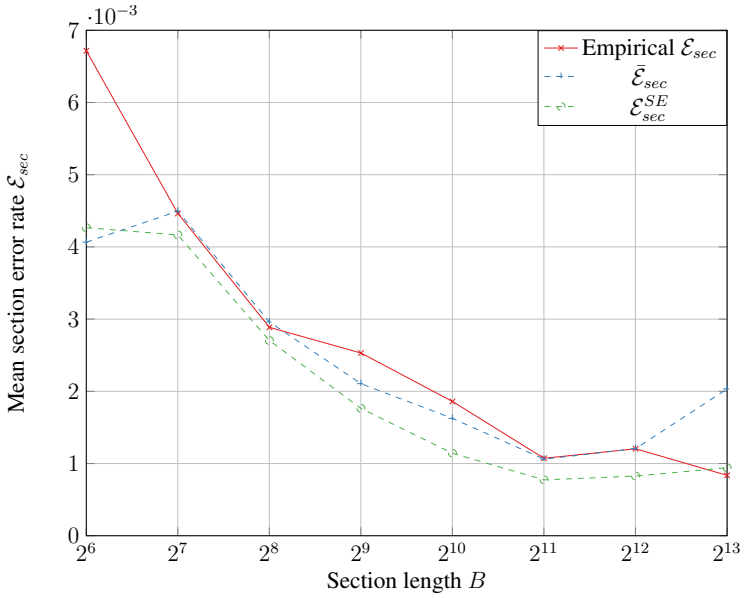


FIGURE 3.5: The number of sections $L = 1024$ is fixed and the section length B is increased to evaluate the effect on the section error rate $\mathcal{E}_{sec}$. For the point $B = 2^{13}$, 1000 runs are conducted. For all other B values more runs are simulated, such that the same amount of bits are transmitted for any choice of parameters. The dashed plots represent the estimated error rates, see Section 2.6. The block length M ranges $[4096, 8875]$. The algorithmic PA is applied and R_{PA} is $[1.01, 1.06]$ monotonically increasing with B. At $SNR = 11.146$

Note that Figure 3.5 is for one fixed SNR value and the threshold, at which the section error rate $\mathcal{E}_{sec}$ does not decrease with a further increase in section length B, depends on the SNR. This can be seen in Figure 3.10, where at lower SNR values the plots for different L/B ratios converge. With increasing SNR, code settings with a greater B value decay faster. As a guideline for empirical block lengths, it is advisable to choose B in an interval of $[L, 4L]$. By fixing L, R, SNR and R_{PA} and increasing the section length B one can observe a change in the section error distribution. This is visualized in Figure 3.6. The greater the section length B gets, the more the distribution changes and the number of section errors per run is less concentrated about the mean value of $\mathcal{E}_{sec}$.

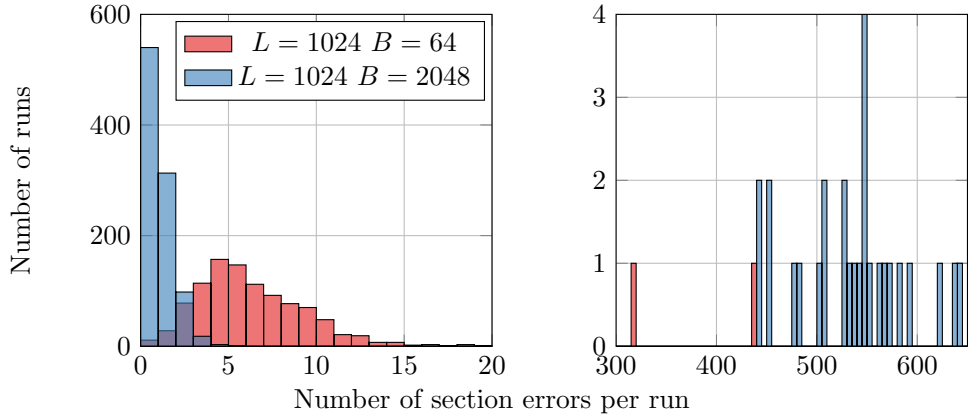


FIGURE 3.6: The error distribution of two different code settings, but $SNR = 11.14, R = 1.5$ and $R_{PA} = 1.01$ are the same for both scenarios, over 10^3 runs. The histogram is divided into a low and a high section error count region. The x-axis is truncated, where outside of the region no error count event occurred.

We see that higher section errors get more likely, when increasing the section length B. This can be explained with the large deviation bound of (3.10), where the exponent of the bound is $\propto 1/\log(B)^{2T-1}$ and therefore a growing B increases the probability of an error event, that deviates from ϵ. Note that the number of error free trials increases for greater B values but so do the high error events. The simulation in Figure 3.6 is done for the same R_{PA} value for both settings. The effect of the increased number of high error events (with greater values for B) can be counteracted by an increase of the R_{PA} value. This can be explained by assuming that the high number error events are caused by section errors at the initial sections, which then in turn propagate the error through AMP iterations to the following sections. Therefore with a greater value of R_{PA} and more power allocated to the initial sections, the overall error performance does not degrade, as we see in Figure 3.5.
By fixing the section length B as well as L, R, SNR, R_{PA} and increasing the number of sections L, we see that the error distribution is more concentrated around the empirical mean and high number error events get less likely. This is visualized in Figure 3.7. This discovery is in accordance with the deviation bound of (3.10), where the negative exponent is $\propto L$.

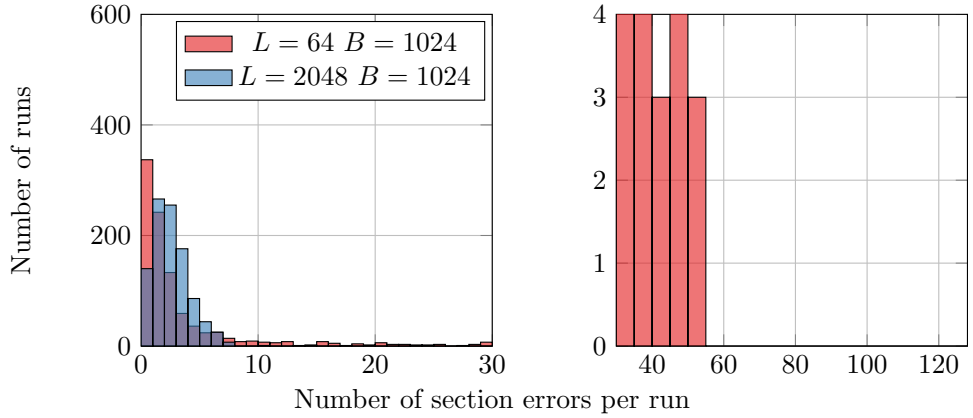


FIGURE 3.7: The error distribution of two different code settings, with $SNR = 11.14, R = 1.5$ and $R_{PA} = 1.14$ are the same for both scenarios, over 1000 runs. The histogram is divided into a low and a high section error count region. The x-axis is truncated, where outside of the region no error count event occurred.

Note that the simulations depicted in Figures 3.6 and 3.7 are conducted for the rate $R = 1.5$ and for lower rates, e.g. $R = 0.5$, these before mentioned phenomenons of changing error distributions do not occur. So when fixing the rate to $R = 0.5$ and the number of sections L, an increase in the section length B does not decrease the error concentration of trials around the mean number of section errors. The optimal L/B ratio not only depends on the SNR but also on the rate R. This circumstance is made clear when comparing Figure 3.9 with Figure 3.10. For the rate $R = 1.5$ SPARCs with the parameters $L = 512$ and $B = 1024$ exhibit a lower $\mathcal{E}_b$ over different $SNRs$ than the setting with $L = 256$ and $B = 2048$. For the rate $R = 0.5$ the opposite is true, see Figure 3.10.

3.5 Dependency of code parameters on the optimal R_{PA}

In Section 2.4 different PA schemes are introduced including the algorithmic PA. The question of how to choose the parameter of the algorithmic PA is still unanswered. To gather intuition about the influence of different code parameters on the optimal R_{PA}, simulations are conducted with the approach of linearly sweeping the possible values of R_{PA} with a constant step width of 0.05 for different code settings. The results of this evaluation are depicted in Figure 3.8. Note that $R_{PA} = 0$ is equal to the constant power allocation. We can see that the optimal value for R_{PA} is strongly dependent on the communication rate R. For lower rates, in our case $R = 0.5$ and $R = 1$, the algorithmic PA does not deliver any improvement over choosing a constant PA scheme. While at higher rates, e.g. $R = 1.5$, the $\mathcal{E}_b$ to R_{PA} plot resembles a convex function with a minimum close to $R_{PA} = 1.1$.

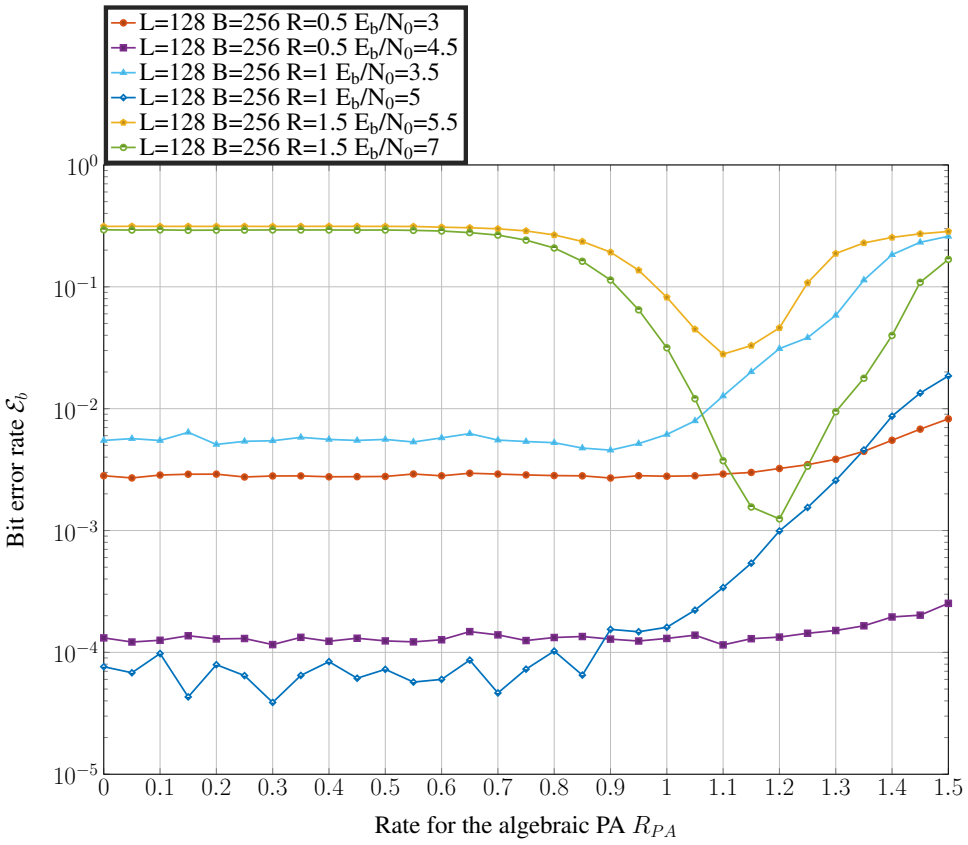


FIGURE 3.8: Bit error rate plotted over the PA-parameter R_{PA} for different code settings.

Therefore one has to be cautious when employing the algorithmic PA scheme, since a wrongly chosen R_{PA} value has a drastic influence on the error performance of SPARCs.

3.6 Bit error rate over SNR

Finally we want to compare the performance of SPARCs, decoded with the AMP algorithm, to conventional coded modulation schemes. First of we choose a communication setting at $R = 1.5$ and compare the performance to LDPC codes of the WiMax standard IEEE 802.16e. For the LDPC codes we have one plot for QAM modulation with the number of symbols $M_a = 16$ and coding rate $R_c = 0.75$, which equals to a communication rate of $\log 2(M_a) \cdot R_c/2$ per dimension and per channel use. The second setting for LDPC codes is $M_a = 64$ and $R_c = 0.5$, which results in the same communication rate. The results are shown in Figure 3.9.

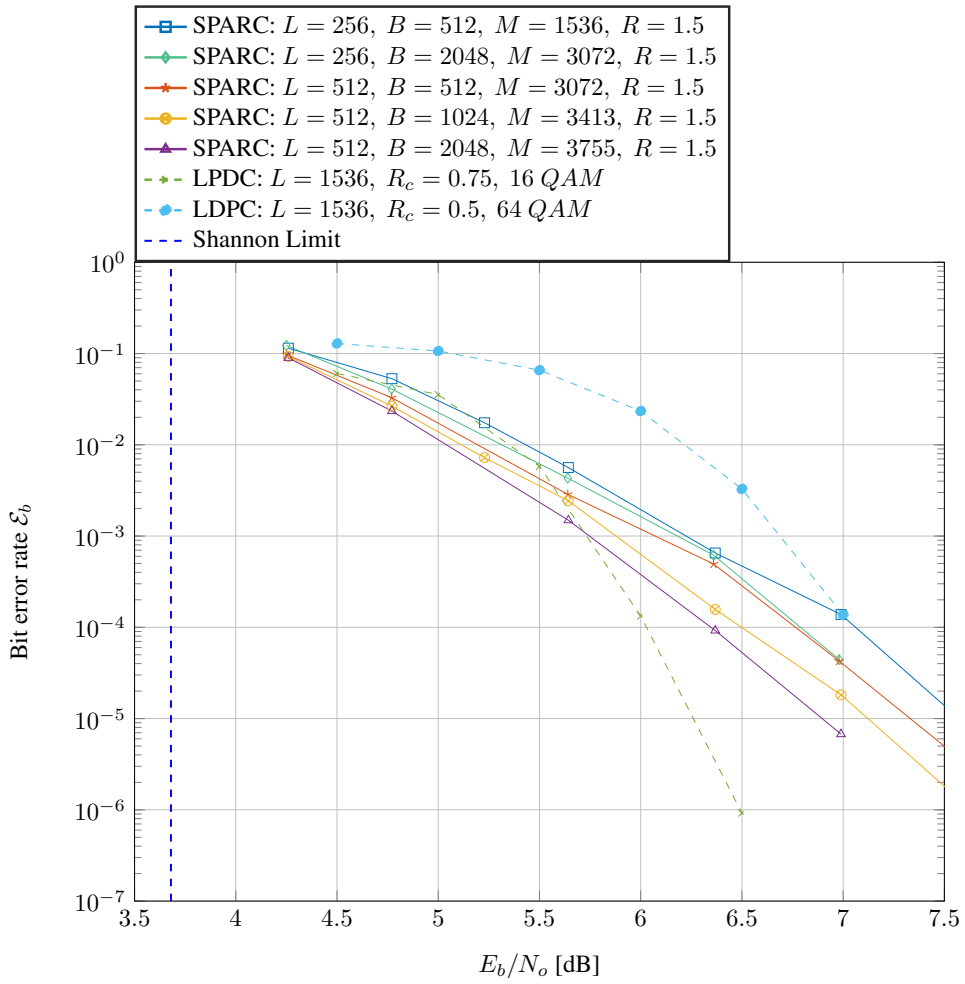


FIGURE 3.9: Bit error rate evaluated over the ratio E_b/N_o for the fixed rate $R = 1.5$, varying block lengths M over 1000 runs and algorithmic PA. Using SPARCs results in a lower $\mathcal{E}_b$ than the 64 QAM modulated LDPC code at SNR values close to the Shannon Limit. The 16 QAM modulated LDPC code outperforms SPARCs for $R = 1.5$.

Another state-of-the-art coded modulation scheme are so called turbo codes, which operate at empirical codeword lengths close to the Shannon capacity. For the rate $R = 0.5$, we compare the turbo code implementation recommended by the CCSDS (Consultative Committee for Space Data Systems) [19] to AMP decoded SPARCs of different block lengths. Note that the coding rate for the Turbo codes is $R_c = 0.5$ and K is the number of information bits. The simulation of the coded modulation schemes are conducted with the use of the coded modulation library found at http://www.iterativesolutions.com/Matlab.htm.

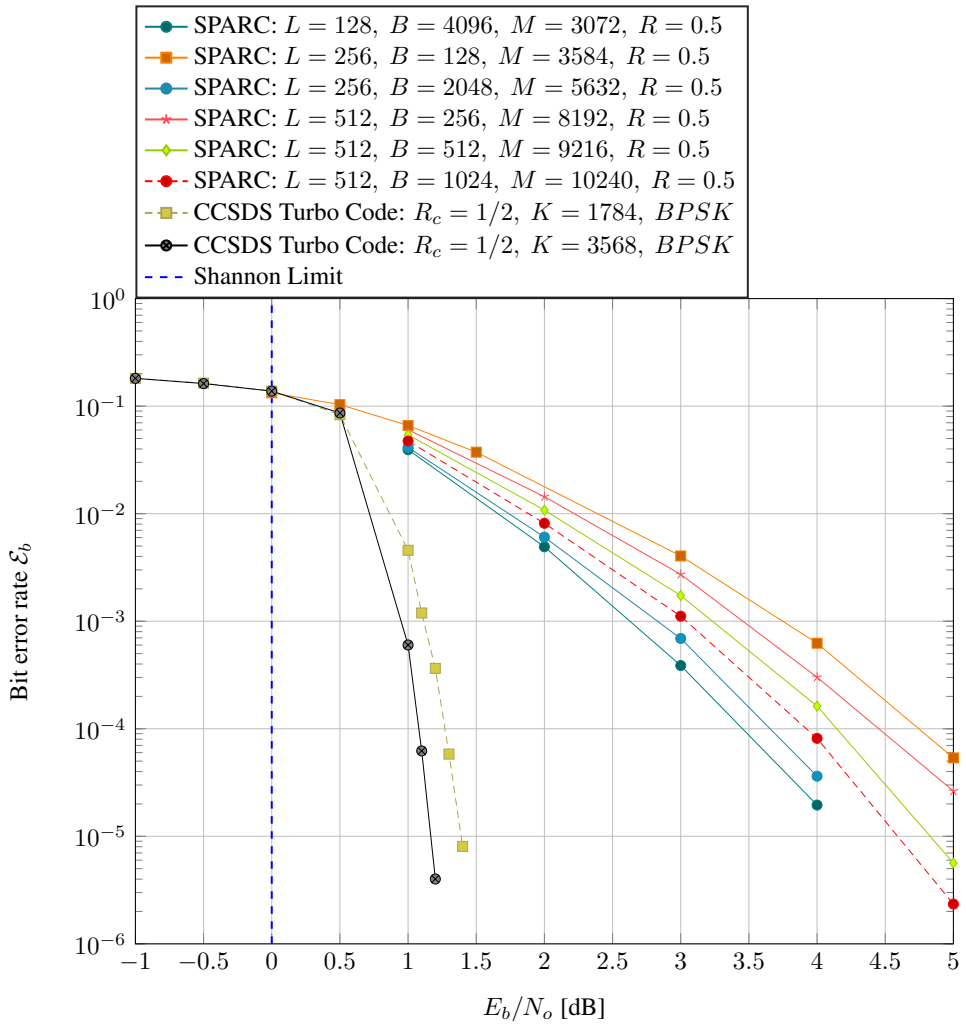


FIGURE 3.10: Bit error rate evaluated over the ratio E_b/N_o for the fixed rate $R = 0.5$, varying block lengths M over 1000 runs and algorithmic PA. For $R = 0.5$ Turbo Codes result in an overall lower $\mathcal{E}_b$ and exhibit a steeper descent in $\mathcal{E}_b$ with increasing SNR than SPARCs.

Note that in Figure 3.10 SPARCs with $L = 128$ and $B = 4092$ actually outperform other settings with a larger block length. This highlights the dependence of the optimal L/B ratio on the rate R. Therefore one has to be cautious of the different dependencies, e.g. L/B on R, L/B on SNR and the PA on R, when applying SPARCs.

3.7 Conclusion

In general SPARCs offer flexibility to dynamically adjust the communication rate R, which well established coding methods lack. The encoding procedure of SPARCs resembles Shannon's ideal of a Gaussian-distributed codebook. With an AMP decoder, specialized to the structure of SPARCs, a numerically feasible approach of estimating the sent information is given. By exchanging the Gaussian with a Hadamard coding matrix the complexity is significantly reduced. Further estimating the state evolution parameters in an on-line manner, diminishes the need of pre-calculation. We observe the SNR dependence of the number of

AMP iterations and can conclude that the number of iterations stay in a manageable region. We provide a heuristic approach of choosing a "section length to number of sections" ratio in a SNR region not too far from the channel capacity. Numerical simulation show that the SPARCs performance with AMP decoding depends strongly on the chosen communication rate. Further, the need of scaling the coefficients of the input vector $\boldsymbol{x}$ arises, when the communication rate is increased. In comparison to more well-known coded modulation schemes, SPARCs are competitive at higher rates, here $R = 1.5$ and at SNR values close to the Shannon Limit. At increased SNR values the bit error rate does not decrease as quickly as for LDPC codes. For lower communication rates classic approaches, e.g. Turbo codes, are at a great advantage. Overall the need of the power allocation and the worse performance at low communication rates are disadvantages of SPARCs with AMP decoding. SPARCs show promising results at higher rates and offer the possibility to easily adjust the rate R.

Bibliography

[1] C. E. Shannon, "A mathematical theory of communication", *The Bell System Technical Journal*, vol. 27, 379–423, 623–656, Jul. 1948.

[2] G. Böcherer, F. Steiner, and P. Schulte, "Bandwidth efficient and rate-matched low-density parity-check coded modulation", *IEEE Transactions on Communications*, vol. 63, no. 12, pp. 4651 –4665, Jul. 2015. [Online]. Available: https://arxiv.org/abs/1502.02733.

[3] A. R. Barron and A. Joseph, "Toward fast reliable communication at rates near capacity with gaussian noise", *IEEE International Symposium on Information Theory*, pp. 315–319, Jun. 2010. [Online]. Available: https://arxiv.org/abs/1006.3870.

[4] A. R. Barron and A. Joseph, "Least squares superposition codes of moderate dictionary size are reliable at rates up to capacity", *IEEE Transactions on Information Theory*, pp. 2541 –2557, May 2012. [Online]. Available: https://arxiv.org/abs/1006.3780.

[5] A. R. Barron and A. Joseph, "Sparse superposition codes are fast and reliable at rates approaching capacity with gaussian noise", 2011. [Online]. Available: https://arxiv.org/pdf/1006.3780.

[6] A. R. Barron and S. Cho, "High-rate sparse superposition codes with iteratively optimal estimates", *IEEE International Symposium on Information Theory Proceedings (ISIT)*, 2012.

[7] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing", *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 18 914–18 919, Sep. 2009. [Online]. Available: http://www.pnas.org/content/106/45/18914.full.

[8] C. Rush, A. Greig, and R. Venkataramanan, "Capacity-achieving sparse superposition codes via approximate message passing decoding", *IEEE Transactions on Information Theory*, vol. 63, no. 3, pp. 1476–1500, Mar. 2017. [Online]. Available: https://arxiv.org/abs/1501.05892.

[9] C. Rush and R. Venkataramanan, "The error exponent of sparse regression codes with amp decoding", *IEEE International Symposium on Information Theory Proceedings (ISIT)*, Feb. 2017.

[10] C. Condo and W. J. Gross, "Sparse superposition codes: A practical approach", *IEEE Workshop on Signal Processing Systems (SiPS)*, Oct. 2015.

[11] C. Condo and W. J. Gross, "Implementation of sparse superposition codes", *IEEE Transactions on Signal Processing*, vol. 65, no. 9, pp. 2421 –2427, May 2017.

[12] R. Venkataramanan, S. Tatikonda, and A. Barron, "Sparse regression codes", *IEEE Information Theory Society Newsletter*, Dec. 2016.

[13] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing", *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 764–785, Jan. 2011. [Online]. Available: https://arxiv.org/abs/1001.3448v4.

[14] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint", *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004, ISSN: 1097-0312. DOI: 10.1002/cpa.20042. [Online]. Available: https://arxiv.org/abs/math/0307152.

[15] R. Venkataramanan and A. Barron, "Isit 2016 tutorial: Sparse regression codes", *International Symposium on Information Theory*, Jul. 2016. [Online]. Available: https://goo.gl/8H8wrk.

[16] A. Greig and R. Venkataramanan, "Techniques for improving the finite length performance of sparse superposition codes", May 2017. [Online]. Available: https://arxiv.org/abs/1705.02091.

[17] W. H. Press, B. P. Flanner, S. A. Teukolsky, and W. T. Vetterling, "Numerical recipes: The art of scientific computing (3rd ed.)", in. Cambridge University Press, 2007, ch. 10.2 Golden Section Search in One Dimension, pp. 492–496.

[18] J. Shanks, "Computation of the fast walsh-fourier transform", *IEEE Transactions on Computers*, vol. C-18, no. 5, pp. 457 –459, May 1969.

[19] "Tm synchronization and channel coding", *Recommendation for Space Data System Standards, CCSDS 131.0-B-2. Blue Book*, vol. 1, Aug. 2011.