Dissertation

Message Passing For Multidimensional Inverse Problems

in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

TU Wien
Institute of Telecommunications

Stefan Birgmeier
Mat.Nr.: 00725468

2019

Advisor

Univ.Prof. Dipl.-Ing. Dr.-Ing. Norbert Görtz

Institute of Telecommunications
TU Wien
Austria

Examiners

Prof. Dr.-Ing. Robert Fischer

Institute of Communications Engineering
Universität Ulm
Germany

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Gerald Matz

Institute of Telecommunications
TU Wien
Austria

Abstract.

Numerous algorithms for the recovery of sparse, high-dimensional signals with independent, identically distributed samples have been discussed in the literature. Some of the most advanced ones are based on approximated message passing algorithms and have been shown to perform well in terms of mean-squared error. Restrictions such as the independence of signal samples as well as rigid requirements imposed upon the measurement matrix often limit the practical usability and performance of these algorithms. This thesis attempts to lift some of these restrictions without constraining the solutions to specific problems. The resulting algorithms maintain or surpass the performance of Bayesian approximate message passing under adverse conditions, while retaining reasonable computational complexity for practical applications.

Declaration.

I hereby declare that, except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification at this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the references and acknowledgments.

Stefan Birgmeier
Wien, September 3, 2019

Contents

Notation

- x is a scalar.
- x is a random variable.
- $\boldsymbol{x}$ is a vector.
- $\mathbf{x}$ is a random vector.
- $\boldsymbol{A}$ is a matrix.
- $\mathcal{X}$ is a set or a general operator, as evident from the context.
- $\mathbf{0}_N$ is the all-zero column vector $(0, 0, \ldots, 0)^T$ of dimension N.
- $\mathbf{1}_N$ is the all-one column vector $(1, 1, \ldots, 1)^T$ of dimension N.
- $f_{\mathbf{x}}(\boldsymbol{x})$ is the probability density function of the random vector $\mathbf{x}$ using the variable $\boldsymbol{x}$.
- $\delta(\boldsymbol{x})$ is the Dirac delta function which is zero everywhere except at $\boldsymbol{x} = \mathbf{0}$ and integrates to one.
- $\mathrm{diag}(\boldsymbol{A})$ is the diagonal of $\boldsymbol{A}$, a vector.
- $\mathrm{Diag}(\boldsymbol{x})$ is a diagonal matrix and $\mathrm{diag}(\mathrm{Diag}(\boldsymbol{x})) = \boldsymbol{x}$.
- $\nabla_{\boldsymbol{x}}$ is the nabla-operator, which computes the gradient with respect to $\boldsymbol{x}$ of the expression to the right-hand side.
- $\nabla \boldsymbol{F}(\boldsymbol{\mu})$ is the Jacobian of $\boldsymbol{F}(\boldsymbol{\mu})$ with respect to $\boldsymbol{\mu}$, where $\boldsymbol{F}(\boldsymbol{\mu})$ is a function with vector-valued argument and result.
- $\mathcal{N}_{\mathbf{x}}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the probability density function of the multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. If the vector $\boldsymbol{x}$ is omitted, it is assumed to be identical to the calligraphic version of the random vector $\mathbf{x}$. If $\boldsymbol{\Sigma}$ is written as lowercase $\boldsymbol{\sigma}^2$, the latter is a vector and $\boldsymbol{\Sigma}$ is a diagonal matrix with $\mathrm{diag}(\boldsymbol{\Sigma}) = \boldsymbol{\sigma}^2$.
- $\mathrm{GF}(q)$ with $q = p^m$ and p prime, $m \in \mathcal{N}^+$, $m > 0$ is the Galois field (finite field) with q elements.

Chapter 1

Introduction

The concept of compressed sensing (CS) was first proposed by David L. Donoho in 2006 [22]. The goal was to minimize effort spent on data acquisition and compression: often, a high-dimensional representation of a complicated signal is acquired only to apply a lossy compression scheme to obtain a low-dimensional representation (examples are audio, image and video compression). If it was possible to obtain a low-dimensional representation right away, a lot of expensive signal processing could be eliminated. Also in 2006, Candes, Romberg and Tao presented results regarding the reconstruction of signals from an incomplete set of Fourier transform coefficients [17]. They motivated their research with the problem of signal reconstruction using data from magnetic resonance imaging (MRI) devices.

Representation of an arbitrary signal of limited bandwidth requires sampling at the Nyquist rate in order to maintain a perfect representation of the signal. If only a small, "sub-Nyquist", number of measurements is known, additional information about the signal is required for reconstruction. A popular model is that of a "sparse" signal, i.e. the assumption that it is zero most of the time. Even assuming linear measurements, the problem of signal recovery is hard. By applying certain relaxations it can be formulated as a convex optimization problem, however. Since dimensions of such problems are usually too large to employ classical convex solvers, a plethora of algorithms optimized for the case of linear measurements of sparse signals were developed. Among these are matching pursuit [44, 47], basis pursuit [19] as well as the "LASSO" (least absolute shrinkage and selection operator) [58]. Subsequently, algorithms making more explicit use of the signal's sparsity, such as iterative thresholding [12, 21] and approximate message passing [23] appeared.

Compressive sensing recovery algorithms are used for a range of applications such as single-pixel camera systems [27, 35] and medical imaging settings [17]. With the arrival of Bayesian approximate message passing (BAMP) it became possible to incorporate even more detailed information about the signal by making explicit use of its prior. Bayesian approximate message passing extended the scope of compressive sensing towards the recovery of discrete-valued signals such as the detection of sparse regression codes [52] and other general inverse problems.

1.1 Structure and Novel Contributions

This thesis presents new methods that overcome some of the restrictions existing algorithms are inhibited by. It is composed of four parts:

- In Chapter 2, the derivation of new reconstruction algorithms for compressed sensing problems under various conditions is discussed. First, Bayesian approximate message passing is extended towards multi-dimensional priors. Subsequently, novel methods are presented that allow BAMP to be applied to problems involving non-zero mean measurement matrices or variable measurement noise. A "state evolution" formalism accurately predicting the performance of each algorithm is provided.

- Chapter 3 discusses the construction of multidimensional "denoisers" and associated functions that are required for the operation of BAMP. Explicit expressions of these functions are provided for several priors. Special attention is paid to numerical stability.

- A practical application is reviewed in Chapter 4, namely the reconstruction of compressed LDPC code words. The chapter presents a structured approach for incorporation of discrete priors with BAMP. Simplifications allowed for by special signal structures such as those of binary block codes are discussed.

- Finally, numerical simulations are presented in Chapter 5. Comparisons with existing algorithms are provided. State evolution is employed both for validation of the numerical results as well as for highlighting improved behavior of the new methods. We show that for certain signals with high-dimensional prior, the proposed algorithms perform close to theoretical limits.

The thesis concludes with a short review of its development and discussion of open research topics.

Chapter 2

Compressed Sensing: Reconstruction Using Multivariate Priors

In this chapter, several algorithms to recover an unknown vector $\boldsymbol{x}$ from linear measurements are presented.

2.1 System Model

For a discrete signal $\boldsymbol{x} \in \mathbb{R}^N$, N samples are required for representation in general. Let $\boldsymbol{y} = \mathcal{T}(\boldsymbol{x})$ be a compressed version of $\boldsymbol{x}$, with $\boldsymbol{y} \in \mathbb{R}^L$, $L < N$ and $\mathcal{T}$ a general compression operator. If (almost) lossless reconstruction of $\boldsymbol{x}$ from $\boldsymbol{y}$ is required, certain additional information must be known about the signal $\boldsymbol{x}$. A popular property is "sparsity": for a signal $\boldsymbol{x}$ there exists a sparse representation if there is a bijective transformation $\mathcal{S} : \mathbb{R}^N \to \mathbb{R}^N$ such that $\boldsymbol{\chi} = \mathcal{S}(\boldsymbol{x})$ is sparse "in the traditional sense", i.e. the majority of $\boldsymbol{\chi}$'s coefficients are (almost) zero.
In compressive sensing, the compression operator $\mathcal{T}$ is usually assumed to be linear, which is reflected by recovery algorithms such as iterative thresholding [12, 21], matching pursuit [44, 47] and approximate message passing [23]. The general acquisition equation $\boldsymbol{y} = \mathcal{T}(\boldsymbol{x})$ of a linear CS system can thus be written as

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

Here, the matrix $\boldsymbol{A}$ is the *measurement matrix* (or *sensing matrix*). Often, it is assumed to have zero-mean, independent and identically distributed (i.i.d.) entries drawn from a Gaussian or Rademacher distribution. Some algorithms furthermore require its columns to be normalized. Almost always it is assumed to be a dense matrix. Since the measurements $\boldsymbol{y} \in \mathbb{R}^L$ and the signal $\boldsymbol{x} \in \mathbb{R}^N$, with $L < N$, the matrix $\boldsymbol{A} \in \mathbb{R}^{L \times N}$ is a wide matrix. The vector $\boldsymbol{w}$ is additive measurement noise. A system's subsampling ratio ρ is defined as $\rho = {}^{L}/_{N}$.

Since $\boldsymbol{A}$ is a wide matrix, reliably obtaining $\boldsymbol{x}$ knowing $\boldsymbol{y}$ and $\boldsymbol{A}$ is impossible in general, even in the noiseless case. Any algorithm recovering $\boldsymbol{x}$ therefore needs to make use of additional information about $\boldsymbol{x}$. In this thesis, such knowledge is specified and employed in the form of a prior $f_{\mathsf{x}}(\boldsymbol{x})$ of x. This has several advantages: on the one hand, a prior is a more complete description of a signal than traditional models like sparsity. On the other hand, if the prior does not describe a sparse signal in a traditional sense, it is possible to find an algorithm directly operating on the non-sparse signal. It is then not necessary to find a "sparsifying" base $\boldsymbol{S}$ such that $\boldsymbol{\chi} = \boldsymbol{S}\boldsymbol{x}$ with $\boldsymbol{\chi}$ sparse. Finally, formulating recovery algorithms in terms of the prior $f_{\mathsf{x}}(\boldsymbol{x})$ makes it possible to re-use these algorithms for a variety of problems by plugging in the appropriate prior. Approximations and optimizations enabled by specially structured priors can then be applied.

2.2 Bayesian Optimal Estimation

For the moment, assume noiseless compressed sensing, i.e. $\boldsymbol{w} = \boldsymbol{0}$. Even in this case, the problem (2.1) is under-determined. A naïve approach to reconstruction of $\boldsymbol{x}$ is to compute the pseudo-inverse $\boldsymbol{A}^+ = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1}$. The reconstructed vector is

$$\hat{\boldsymbol{x}} = \boldsymbol{A}^+\boldsymbol{y}\,. \tag{2.2}$$

This solution does not take into account the information provided by the prior $f_{\mathsf{x}}(\boldsymbol{x})$. In fact, it makes implicit assumptions about the prior: it can be shown that the pseudo-inverse results in

$$\hat{\boldsymbol{x}} = \arg\min_{\hat{\boldsymbol{x}}} \|\hat{\boldsymbol{x}}\|_2 \;\text{ s.t. }\; \boldsymbol{y} = \boldsymbol{A}\hat{\boldsymbol{x}}\,, \tag{2.3}$$

i.e. the solution with the minimal l_2 norm that satisfies $\boldsymbol{y} = \boldsymbol{A}\hat{\boldsymbol{x}}$. Let the optimality criteria be the minimization of the error term

$$\varepsilon = \mathbb{E}_{\mathsf{x}}\left\{\|\mathsf{e}\|^2\right\} = \mathbb{E}_{\mathsf{x}}\left\{\|\hat{\boldsymbol{x}} - \mathsf{x}\|^2\right\}. \tag{2.4}$$

The following steps can be performed to minimize the error ε:

$$\hat{\boldsymbol{x}} = \arg\min_{\hat{\boldsymbol{x}}} \mathbb{E}_{\mathsf{x}}\left\{\|\hat{\boldsymbol{x}} - \mathsf{x}\|^2\right\} \tag{2.5}$$

$$= \arg\min_{\hat{\boldsymbol{x}}} \left(\mathbb{E}_{\mathsf{x}}\left\{\hat{\boldsymbol{x}}^T\hat{\boldsymbol{x}}\right\} - \mathbb{E}_{\mathsf{x}}\left\{\mathsf{x}^T\hat{\boldsymbol{x}}\right\} - \mathbb{E}_{\mathsf{x}}\left\{\hat{\boldsymbol{x}}^T\mathsf{x}\right\} + \mathbb{E}_{\mathsf{x}}\left\{\mathsf{x}^T\mathsf{x}\right\}\right) \tag{2.6}$$

$$\hat{\boldsymbol{x}} = \left\{\hat{\boldsymbol{x}} \;\middle|\; \nabla_{\hat{\boldsymbol{x}}}\left(\hat{\boldsymbol{x}}^T\hat{\boldsymbol{x}} - 2\hat{\boldsymbol{x}}^T\mathbb{E}_{\mathsf{x}}\{\mathsf{x}\} + \sum_i P_{\mathsf{x},i}\right) = \boldsymbol{0}_N\right\} \tag{2.7}$$

$$\boldsymbol{0}_N = 2\hat{\boldsymbol{x}} - 2\mathbb{E}_{\mathsf{x}}\{\mathsf{x}\} \tag{2.8}$$

$$\hat{\boldsymbol{x}} = \mathbb{E}_{\mathsf{x}}\{\mathsf{x}\}\,, \tag{2.9}$$

i.e. the error is minimized by the expected value of $\mathbf{x}$. Since $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, knowing $\boldsymbol{y}$ allows to further reduce the estimation error by minimizing the conditional error

$$\varepsilon' = \mathbb{E}_{\mathbf{x}|\mathbf{y}} \left\{ \|\mathbf{e}\|^2 \,|\mathbf{y} = \boldsymbol{y} \right\}. \tag{2.10}$$

Performing the same steps as above results in the optimal estimate of $\boldsymbol{x}$ (in terms of mean squared error (MSE)) as

$$\hat{\boldsymbol{x}} = \mathbb{E}_{\mathbf{x}|\mathbf{y}} \left\{ \mathbf{x} | \mathbf{y} = \boldsymbol{y} \right\}. \tag{2.11}$$

The distribution $f_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x}|\boldsymbol{y})$ is generally unknown, however the distribution of $f_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}|\boldsymbol{x})$ can be written as

$$f_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}|\boldsymbol{x}) = \delta(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}) \tag{2.12}$$

in the noiseless case. In the language of convex optimization, this conditional distribution is an equality constraint, i.e.

$$y_a - \boldsymbol{A}_a \boldsymbol{x} = 0 \quad \forall a \in \{1 \dots L\}. \tag{2.13}$$

This constraint describes L hyperplanes in N-dimensional space and thus their intersection is an at least $(N-L)$-dimensional space. Additional information provided by the prior $f_{\mathbf{x}}(\boldsymbol{x})$ is then hoped to be sufficient to reduce this space of possible solutions to a single solution. For the noiseless case, (2.11) can further be developed using (2.12) and Bayes' rule:

$$f_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x}|\boldsymbol{y}) = \frac{1}{f_{\mathbf{y}}(\boldsymbol{y})} f_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}|\boldsymbol{x}) f_{\mathbf{x}}(\boldsymbol{x}). \tag{2.14}$$

The distribution $f_{\mathbf{y}}(\boldsymbol{y})$ can be written as

$$f_{\mathbf{y}}(\boldsymbol{y}) = \int_{\mathbb{R}^N} f_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}|\boldsymbol{x}) f_{\mathbf{x}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}. \tag{2.15}$$

In the noiseless case, (2.14) and (2.15) are

$$f_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x}|\boldsymbol{y}) = \frac{1}{f_{\mathbf{y}}(\boldsymbol{y})} \delta(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}) f_{\mathbf{x}}(\boldsymbol{x}) \tag{2.16}$$

$$f_{\mathbf{y}}(\boldsymbol{y}) = \int_{\mathbb{R}^N} \delta(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}) f_{\mathbf{x}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}. \tag{2.17}$$

The estimate of $\boldsymbol{x}$ can thus be written as (cf. (2.11))

$$\hat{\boldsymbol{x}} = \frac{1}{f_{\mathbf{y}}(\boldsymbol{y})} \int_{\mathbb{R}^N} \boldsymbol{x} \delta(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}) f_{\mathbf{x}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}. \tag{2.18}$$

It turns out that the integrals required to evaluate (2.18) can be cumbersome to compute due to the Dirac term. Thus, it can be practical to "soften" the constraint. The Dirac is substituted by

$$f_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}|\boldsymbol{x}) \propto \exp\left(-\frac{\beta}{2} \sum_a (y_a - \boldsymbol{A}_a \boldsymbol{x})^2 \right). \tag{2.19}$$

For $\beta \to \infty$ (2.19) approaches (2.12).

2.3 Noisy Case

At this point, the noise can be factored in. For now, it is assumed that the noise components w_a are independent and that they have a finite mean and variance. Typically, the noise is modeled as Gaussian with

$$f_{\mathbf{w}}(\boldsymbol{w}) \propto \exp\left(-\frac{1}{2}\sum_a \frac{(w_a - \mu_{\mathsf{w}_a})^2}{\sigma^2_{\mathsf{w}_a}}\right). \tag{2.20}$$

With $y_a = \boldsymbol{A}_a\boldsymbol{x} + w_a$ and $\boldsymbol{y}$ known thus

$$w_a = y_a - \boldsymbol{A}_a\boldsymbol{x}\,. \tag{2.21}$$

The conditional distribution $f_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}|\boldsymbol{x})$ can then be written as

$$f_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}|\boldsymbol{x}) \propto \exp\left(-\frac{1}{2}\sum_a \frac{(y_a - \boldsymbol{A}_a\boldsymbol{x} - \mu_{\mathsf{w}_a})^2}{\sigma^2_{\mathsf{w}_a}}\right). \tag{2.22}$$

For noise variance $\sigma^2_{\mathsf{w}_a} = 0$ and mean $\mu_{\mathsf{w}_a} = 0$, the expression (2.22) reduces to (2.12). In the presence of noise or when using the "softened" constraint as in (2.19), the conditional distribution $f_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}|\boldsymbol{x})$ can be thought of as the product (versus the intersection in the hard constraint description) of L "blurry" hyperplanes. Without loss of generality it shall be assumed that $\boldsymbol{\mu}_{\mathbf{w}} = \mathbb{E}\{\mathbf{w}\} = \mathbf{0}$. If the noise is not zero-mean the problem can be transformed by setting $\boldsymbol{y}' = \boldsymbol{y} - \boldsymbol{\mu}_{\mathbf{w}}$. Plugging (2.22) into (2.11) gives

$$\hat{\boldsymbol{x}} = \frac{1}{f_{\mathbf{y}}(\boldsymbol{y})}\int_{\mathbb{R}^N} \boldsymbol{x}\exp\left(-\frac{1}{2}\sum_a \frac{(y_a - \boldsymbol{A}_a\boldsymbol{x})^2}{\sigma^2_{\mathsf{w}_a}}\right) f_{\mathbf{x}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\,. \tag{2.23}$$

For large N, (2.23) becomes computationally prohibitive. First, one needs to precompute expressions for N integrals over N dimensions (one for each entry of $\boldsymbol{x}$). For all but the simplest priors $f_{\mathbf{x}}(\boldsymbol{x})$, this results in large expressions. Secondly, each of these expressions needs to be evaluated for every instance of the problem. Another drawback of such an approach is its lack of flexibility. Small changes to the prior or the noise distribution require recomputation of all integrals.
It is thus beneficial to approximate the computation of $\hat{\boldsymbol{x}}$. To this end, the sum-product algorithm has been found to be effective. It should be noted right away that the sum-product algorithm was derived for graphs that are trees. It shall be shown that the expression (2.23) does not have a tree-like structure for general matrices $\boldsymbol{A}$. The sum-product algorithm efficiently computes the marginals of a high-dimensional probability density function (pdf). The original sum-product algorithm (when applied to a pdf that can be represented as a tree) computes all marginals in the same time it would take a naïve, variable-by-variable algorithm,

to compute two marginals [40]. Using these marginals, any moment of a single variable can easily be calculated. The pdf in (2.23) is

$$f_{\mathsf{x},\mathsf{y}}(\boldsymbol{x},\boldsymbol{y}) \propto \exp\left(-\frac{1}{2}\sum_a \frac{(y_a - \boldsymbol{A}_a\boldsymbol{x})^2}{\sigma^2_{\mathsf{w}_a}}\right) f_{\mathsf{x}}(\boldsymbol{x})\,. \tag{2.24}$$

Assuming that the prior $f_{\mathsf{x}}(\boldsymbol{x})$ can be factored, i.e.

$$f_{\mathsf{x}}(\boldsymbol{x}) = \prod_{k=1}^{K} f_{\mathsf{x}_k}(\boldsymbol{x}_k)\,, \tag{2.25}$$

then the pdf (2.24) can be written as

$$f_{\mathsf{x},\mathsf{y}}(\boldsymbol{x},\boldsymbol{y}) \propto \exp\left(-\frac{1}{2}\sum_a \frac{(y_a - \boldsymbol{A}_a\boldsymbol{x})^2}{\sigma^2_{\mathsf{w}_a}}\right) \prod_{k=1}^{K} f_{\mathsf{x}_k}(\boldsymbol{x}_k)\,. \tag{2.26}$$

Message passing provides a simplification of the problem only if the prior $f_{\mathsf{x}}(\boldsymbol{x})$ can be factored into many small factors, i.e. K not much smaller than N. The reason for this will become apparent in the next paragraphs.

2.4 Representation as Graph

In order to apply the sum-product algorithm, the graphical representation of the pdf (2.26) must be found. In the graph, random variables are represented by white circular variable nodes, known variables are represented by black circular variable nodes. Factors are depicted as black squares. If a variable appears in a factor, it is connected to the appropriate factor node by an edge. An example for such a factor graph can be seen in Fig. 2.1. In this thesis, variable nodes use the indices i, j, k while factor nodes are indexed with the letters a, b, unless specified otherwise. To simplify notation, products of terms indexed by a certain letter include all terms using the index. A product $\prod_i$ therefore usually ranges over N terms associated with variable nodes, while the product $\prod_{j\neq i}$ is composed of $N-1$ terms, omitting the j^{th} term.
Let $\mathcal{X}_k = \{\mathsf{x}_{k,1}, \mathsf{x}_{k,2}, \ldots, \mathsf{x}_{k,\kappa_k}\}$, i.e. the set of all random variables $\mathsf{x}_{k,i}$ on which the factor $f_{\mathsf{x}_k}(\boldsymbol{x}_k)$ depends, with $\kappa_k = |\mathcal{X}_k|$. The prior factors $f_{\mathsf{x}_k}(\boldsymbol{x}_k)$ are not required to depend on the same number of variables. If the condition

$$\mathcal{X}_k \cap \mathcal{X}_l = \emptyset \quad \forall k, l \in \{1, \ldots, K\}\,,\ k \neq l \tag{2.27}$$

is satisfied, then the prior factors $f_{\mathsf{x}_k}(\boldsymbol{x}_k)$ are disjoint, i.e. each variable x_i is only connected to a single prior factor (cf. Fig. 2.1). If (2.27) is not fulfilled, it is possible for any x_i to appear in multiple prior factors. This thesis focuses on disjoint prior factors. A possible approach for a particular problem featuring overlapping priors

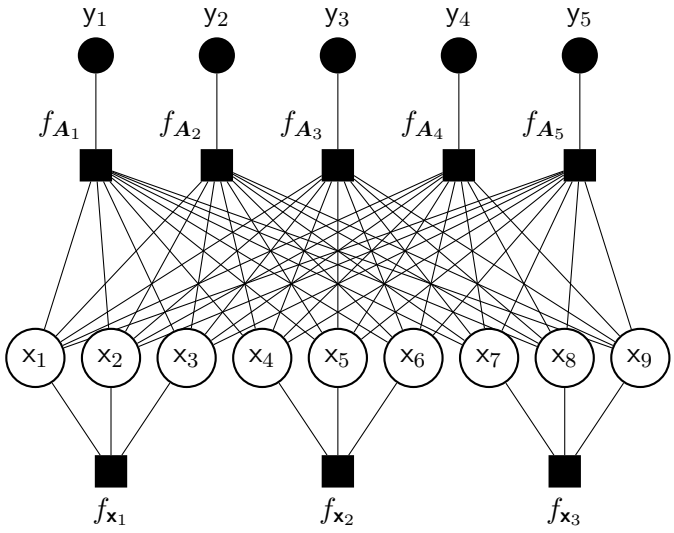


Figure 2.1: Example of a graphical model describing a pdf with $K = 3$ factors of the prior $f_{\mathsf{x}}(\boldsymbol{x})$, $N = 9$ unknown variables x_i and $L = 5$ known variables y_i.

is shown in Chapter 4. Requiring the sets $\mathcal{X}_k$ to be disjoint limits the applicability of compressed sensing recovery algorithms, however.
The sum-product algorithm defines "messages" from variable node to factor node and vice versa. These messages are functions of a single random variable, which is involved in the edge connecting the respective nodes. Note that due to the definition of the graph structure, edges always connect nodes of different type. The graph is thus "bipartite" as no direct connections between two variable nodes or two factor nodes can exist. The message passing rules are [40]

$$m_{a\to i}(x_i) = \int_{j\neq i} f_a(\boldsymbol{x}_a) \prod_{j\in\partial a\setminus i} m_{j\to a}(x_j)\mathrm{d}x_j \tag{2.28}$$

$$m_{i\to a}(x_i) = \prod_{b\neq a} m_{b\to i}(x_i)\,, \tag{2.29}$$

where in (2.28), the integral computes the marginal with respect to (w.r.t.) the random variable x_i. The functions $m_{a\to i}(x_i)$ and $m_{i\to a}(x_a)$ are the messages from factor node to variable node and vice versa respectively. The function $f_a(\boldsymbol{x}_a)$ is the factor corresponding to the factor node with index a. The vector $\boldsymbol{x}_a$ contains all variables adjacent to the factor. The expression $j \in \partial a \setminus i$ indicates that the product includes all messages from variable nodes adjacent to the factor node a *except* the message from the variable node i (which is the target of the message $m_{a\to i}(x_i)$). The marginal $m_{\mathsf{x},i}(x_i)$ w.r.t. a variable x_i is obtained as

$$m_{\mathsf{x},i}(x_i) = \prod_{a} m_{a\to i}(x_i)\,. \tag{2.30}$$

When the pdf can be represented by a tree, message passing rules (2.28) and (2.29) do not incur circular dependencies. Thus it is sufficient to calculate the messages for all edges in both directions to obtain marginals of every random variable x_i.
In the compressed sensing case, the pdf is given in (2.26). For general matrices $\boldsymbol{A}$, this pdf cannot be represented by a tree; it contains cycles. Thus the message passing rules will contain circular dependencies, giving rise to an iterative algorithm. The factors corresponding to factor nodes are

$$f_a(\boldsymbol{x}) = \exp\left(-\frac{(y_a - \boldsymbol{A}_a\boldsymbol{x})^2}{2\sigma^2_{\mathsf{w}_a}}\right) \tag{2.31}$$

$$f_k(\boldsymbol{x}_k) = f_{\mathbf{x}_k}(\boldsymbol{x}_k)\,. \tag{2.32}$$

The functions $f_a(\boldsymbol{x})$ represent the linear relationship between the measurements $\boldsymbol{y}$, the unknown vector $\boldsymbol{x}$ and the sensing matrix $\boldsymbol{A}$. They are therefore referred to as the "matrix factors", with one factor for each row of the matrix $\boldsymbol{A}$. The functions $f_k(\boldsymbol{x}_k)$ are factors of the prior $f_{\mathbf{x}}(\boldsymbol{x})$. The variables y_a are known and marked as such. The random variables x_i are depicted as unknown variable nodes. If condition (2.27) is true, inserting into the message passing rules (2.28) and (2.29) results in

$$\begin{aligned} m_{a\to i}(x_i) &= \int_{j\neq i} f_a(\boldsymbol{x}) \prod_{j\in\partial a\setminus i} m_{j\to a}(x_j)\mathrm{d}x_j \\ &= \int_{j\neq i} \exp\left(-\frac{(y_a - \boldsymbol{A}_a\boldsymbol{x})^2}{2\sigma^2_{\mathsf{w}_a}}\right) \prod_{j\in\partial a\setminus i} m_{j\to a}(x_j)\mathrm{d}x_j \end{aligned} \tag{2.33}$$

$$m_{i\to a}(x_i) = m_{k\to i}(x_i) \prod_{b\neq a} m_{b\to i}(x_i) \tag{2.34}$$

$$m_{k\to i}(x_i) = \int_{j\neq i} f_{\mathbf{x}_k}(\boldsymbol{x}_k) \prod_{j\in\mathcal{X}_k\setminus i} m_{j\to k}(x_j)\mathrm{d}x_j \tag{2.35}$$

$$m_{i\to k}(x_i) = \prod_a m_{a\to i}(x_i)\,. \tag{2.36}$$

In (2.33), the index a has been dropped from the factor node's function argument due to the fact that it depends on all variables in the vector $\boldsymbol{x}$. For special matrices $\boldsymbol{A}$ this might be different. Examples are sparse matrices and blockwise-sparse matrices. In this thesis, the matrix $\boldsymbol{A}$ is assumed to be dense, however. Due to the requirement that the sets $\mathcal{X}_k$ be disjoint, every variable node x_i is only connected to a single prior $f_k(\boldsymbol{x}_k)$. If condition (2.27) is not true, the message passing rules (2.34) and (2.36) become

$$m_{i\to a}(x_i) = \prod_{k\in\partial i} m_{k\to i}(x_i) \prod_{b\neq a} m_{b\to i}(x_i) \tag{2.37}$$

$$m_{i\to k}(x_i) = \prod_{l\in\partial i\setminus k} m_{l\to i}(x_i) \prod_a m_{a\to i}(x_i)\,. \tag{2.38}$$

The left-hand products in (2.37) and (2.38) are over messages from the prior factors connected to the variable with index i. Note that the messages defined in (2.33) to (2.38) contain products of functions which are defined on $\mathbb{R}$. This is computationally intractable for all but few functions. Approximations are required to implement the message passing algorithm. In the following sections, some approximations that are possible under certain conditions are discussed. These assume disjoint prior factors.

While the initialization and termination of the sum-product algorithm is well defined on graphs without cycles, these steps are complicated in presence of cyclic dependencies. Initialization of the sum-product algorithm is done at the leaf nodes. A cyclic graph might not contain any leaf nodes, however. Another problem that only arises in cyclic graphs is message scheduling. Unknown messages are typically replaced by the "least-informative" one, i.e. $m(x) = 1$. In cyclic graphs, the order of message computation needs to be chosen since it is not predefined. Different choices yield different algorithmic behavior. One possible scheduling algorithm is to use "parallel" ("flooding") updates, where all messages from factor nodes to variable nodes (and vice versa) are computed simultaneously. Another scheduling procedure is that of sequential updates, where messages are passed between a (set of) variable and factor node(s) until they converge, after which the next (set of) variable and factor node(s) is updated, taking into account previously converged messages. The sum-product algorithm does not naturally terminate on cyclic graphs. Typically, a maximum number of iterations limits the execution time. Another stopping criterion is normalized change of unknown variable's marginals or their moments.

2.5 Approximating Message Passing

As noted in Section 2.4, applying the message passing rules as they are given in (2.33) to (2.36) *verbatim* is computationally intractable. It is desirable to find realizable approximations to the iterative message passing algorithm. In a first step, some messages are replaced by normal distributions. Normal distributions are described by just two parameters, namely mean μ and variance σ^2, which reduces computational complexity compared to general function-valued messages. If the graph corresponding to a pdf is a tree, the messages are scaled pdfs. The scaling factor is well defined in the sense that the marginal w.r.t. a particular random variable x_i obtained by (2.30) is a valid pdf. On graphs with cycles, a naïve iterative message passing algorithm might produce arbitrarily scaled messages (growing, shrinking and oscillating with iterations). This causes numerical problems. An iterative message passing algorithm must therefore always keep message scaling under control. Passing the moments instead of function-valued message achieves this as a side effect.
Using a Gaussian approximation is well justified in the compressed sensing case. In the subsequent sections, Bayesian approximate message passing is derived for unknown vectors $\boldsymbol{x}$ with multidimensional, disjoint priors. This work has been presented in part at the 12$^{\text{th}}$ International ITG Conference on Systems, Communications and Coding (SCC 2019) [9].

2.5.1 Gaussian Approximation

To approximate the message $m_{a\to i}(x_i)$ (cf. (2.33)) with a Gaussian function, a new random variable $\mathsf{z}_{a\to i}$ shall be introduced, defined as

$$\mathsf{z}_{a\to i} = y_a - \sum_{j\neq i} A_{a,j}\mathsf{x}_{j\to a}\,, \tag{2.39}$$

with mean $\mu_{\mathsf{z},a,i}$ and variance $\sigma^2_{\mathsf{z},a,i}$

$$\mu_{\mathsf{z},a\to i} = y_a - \sum_{j\neq i} A_{a,j}\mu_{\mathsf{x},j\to a} \tag{2.40}$$

$$\sigma^2_{\mathsf{z},a\to i} = \sum_{j\neq i} A^2_{a,j}\sigma^2_{\mathsf{x},j\to a}\,. \tag{2.41}$$

It can be shown [43] that (2.33) can be approximated by

$$\hat{m}_{a\to i}(x_i) = \sqrt{\frac{A^2_{a,i}}{2\pi(\sigma^2_{\mathsf{w},a}+\sigma^2_{\mathsf{z},a\to i})}}\exp\left(-\frac{(A_{a,i}x_i-\mu_{\mathsf{z},a\to i})^2}{2(\sigma^2_{\mathsf{w},a}+\sigma^2_{\mathsf{z},a\to i})}\right) \tag{2.42}$$

$$= \mathcal{N}_{A_{a,i}\mathsf{x},i}(\mu_{\mathsf{z},a\to i}, \sigma^2_{\mathsf{w},a}+\sigma^2_{\mathsf{z},a\to i}) \tag{2.43}$$

$$= \mathcal{N}_{\mathsf{x},i}\left(\frac{\mu_{\mathsf{z},a\to i}}{A_{a,i}}, \frac{\sigma^2_{\mathsf{w},a}+\sigma^2_{\mathsf{z},a\to i}}{A^2_{a,i}}\right). \tag{2.44}$$

Following the message passing rule (2.33) and using the fact that messages from matrix factor node to variable node are approximated by a Gaussian, the message from variable node to matrix factor node is

$$m_{i\to a}(x_i) = m_{k\to i}(x_i) \prod_{b\neq a} \mathcal{N}_{\mathsf{x},i}(\mu_{\mathsf{x},b\to i}, \sigma^2_{\mathsf{x},b\to i}) \,, \tag{2.45}$$

where

$$\mu_{\mathsf{x},b\to i} = \frac{\mu_{\mathsf{z},b\to i}}{A_{b,i}} \tag{2.46}$$

$$\sigma^2_{\mathsf{x},b\to i} = \frac{\sigma^2_{\mathsf{w},b} + \sigma^2_{\mathsf{z},b\to i}}{A^2_{b,i}} \,. \tag{2.47}$$

The form in (2.45) contains the product of L normal distributions in x_i. This product can be written as a single normal distribution in x_i using the relations

$$\frac{1}{\sigma^{2(l)}_{\mathsf{x},i\to a}} = \sum_{b\neq a} \frac{1}{\sigma^2_{\mathsf{x},b\to i}} \tag{2.48}$$

$$\mu^{(l)}_{\mathsf{x},i\to a} = \sigma^{2(l)}_{\mathsf{x},i\to a} \sum_{b\neq a} \frac{\mu_{\mathsf{x},b\to i}}{\sigma^2_{\mathsf{x},b\to i}} \,. \tag{2.49}$$

The notation $\sigma^{2(l)}_{\mathsf{x},i\to a}$ and $\mu^{(l)}_{\mathsf{x},i\to a}$ indicates that these are the mean and variance of the Gaussian function resulting from the product of normal distributions in (2.45), intended for computation of the message from the variable node with index i to the matrix factor node with index a (the "local" means and variances, not taking into account messages of prior factor nodes). The expressions (2.49), (2.48) are inconvenient due to the large number of divisions required. Furthermore, they become numerically unstable for small values of $\sigma^2_{\mathsf{x},a\to i}$. Note that for $\sigma^2_{\mathsf{x},a\to i} = \sigma^2_{\mathsf{x},*\to i}$ (i.e. equal variances) the equations can be reformulated as

$$\frac{1}{\sigma^{2(l)}_{\mathsf{x},i\to a}} = \frac{L-1}{\sigma^2_{\mathsf{x},*\to i}} \tag{2.50}$$

$$\mu^{(l)}_{\mathsf{x},i\to a} = \frac{1}{L-1} \sum_{b\neq a} \mu_{\mathsf{x},b\to i} \,, \tag{2.51}$$

i.e. the mean becomes the arithmetic mean and the variance is scaled by the factor $(L-1)^{-1}$. If the requirement is introduced that the columns of the matrix $\boldsymbol{A}$ are normalized ($\|\boldsymbol{A}_{:,i}\|_2 = 1$) and if it can be assumed that $A^2_{a,i} \approx L^{-1}$ then (2.47) can be written as

$$\sigma^2_{\mathsf{x},a\to i} \approx L \left(\sigma^2_{\mathsf{w}_a} + \sigma^2_{\mathsf{z},a\to i}\right) . \tag{2.52}$$

Using this approximation and plugging (2.52) into (2.50) results in

$$\sigma^{2(l)}_{\mathsf{x},i} = \sigma^{2(l)}_{\mathsf{x},i\to *} \approx \frac{L}{L-1} \left(\sigma^2_{\mathsf{w}_*} + \sigma^2_{\mathsf{z},*\to i}\right) \approx \left(\sigma^2_{\mathsf{w}_*} + \sigma^2_{\mathsf{z},*\to i}\right) . \tag{2.53}$$

For large N and L, the variance of the product of normal distributions is negligibly larger than the variance in the matrix factor node message. For the next step, set (cf. (2.46))

$$\mu_{\mathsf{x},a\to i} = \frac{\mu_{\mathsf{z},a\to i}}{A_{a,i}} = \frac{A_{a,i}}{A_{a,i}}\frac{\mu_{\mathsf{z},a\to i}}{A_{a,i}} \approx L\left(A_{a,i}\mu_{\mathsf{z},a\to i}\right) . \tag{2.54}$$

Inserting (2.54) together with (2.52) into (2.49) results in

$$\mu^{(l)}_{\mathsf{x},i\to a} \approx \frac{L}{L-1}\left(\sigma^2_{\mathsf{w}_*} + \sigma^2_{\mathsf{z},*\to i}\right)\sum_{b\neq a}\frac{L\left(A_{a,i}\mu_{\mathsf{z},a\to i}\right)}{L\left(\sigma^2_{\mathsf{w}_*} + \sigma^2_{\mathsf{z},*\to i}\right)} \tag{2.55}$$

$$= \frac{L}{L-1}\sum_{b\neq a} A_{a,i}\mu_{\mathsf{z},a\to i} \tag{2.56}$$

$$\approx \sum_{b\neq a} A_{b,i}\mu_{\mathsf{z},b\to i} . \tag{2.57}$$

To perform the simplifications in the steps above, a number of assumptions has been made. These are

1. $L \gg 1$
2. $N \gg 1$
3. $\|\boldsymbol{A}_{a,:}\| = 1$
4. $\mathbb{E}\left\{A^2_{a,i}\right\} \approx \frac{1}{L}$
5. $\sigma^2_{\mathsf{x},a\to i} = \sigma^2_{\mathsf{x},*\to i}$.

Conditions 1, 2, 3 and 4 impose tight restrictions on the sensing matrix. These are an indication why BAMP is not applicable "out of the box" to many compressed sensing problems, specifically those where the sensing matrix is predetermined by the problem and cannot be designed. Condition 5 requires that

$$\sigma^2_{\mathsf{w}_b} + \sigma^2_{\mathsf{z},b\to i} \approx \sigma^2_{\mathsf{w}_*} + \sigma^2_{\mathsf{z},*\to i} , \tag{2.58}$$

which amounts to requiring the noise variances $\sigma^2_{\mathsf{w},a}$ and also $\sigma^2_{\mathsf{z},a\to i}$ to be at least approximately independent of a, which is assumed due to the form of (2.41) [23]. An extension towards variable noise variance $\sigma^2_{\mathsf{w},a}$ is presented in Section 2.6.
For now, only expressions of the "local" means and variances have been derived. Using these, (2.45) can be written as

$$m_{i\to a}(x_i) = m_{k\to i}(x_i)\mathcal{N}_{\mathsf{x},i}(\mu^{(l)}_{\mathsf{x},i\to a}, \sigma^{2(l)}_{\mathsf{x},i\to a}) \tag{2.59}$$

$$\approx m_{k\to i}(x_i)\mathcal{N}_{\mathsf{x},i}(\mu^{(l)}_{\mathsf{x},i\to a}, \sigma^{2(l)}_{\mathsf{x},i}) . \tag{2.60}$$

The message $m_{i\to a}(x_i)$ cannot be represented by a normal distribution in many cases. Whether this is possible depends on the form of $m_{k\to i}(x_i)$ (cf. (2.45)) and thus on the prior factor $f_{\mathbf{x}_k}(\boldsymbol{x}_k)$. However, since computation of $m_{a\to i}(x_i)$ merely requires the mean and variance of $m_{i\to a}(x_i)$, it is sufficient to pass these parameters instead of the function-valued message. They are obtained by the functions $F_i(\dots)$ and $G_i(\dots)$, which depend on $\mu^{(l)}_{\mathsf{x},i\to a}$, $\sigma^{2(l)}_{\mathsf{x},i}$ and $m_{k\to i}(x_i)$:

$$\mu_{\mathsf{x},i\to a} = F_i(\dots) = \frac{1}{Z}\int x_i m_{k\to i}(x_i)\mathcal{N}_{\mathsf{x}}(\mu^{(l)}_{\mathsf{x},i\to a}, \sigma^{2(l)}_{\mathsf{x},i\to a})\mathrm{d}x_i \tag{2.61}$$

$$\sigma^2_{\mathsf{x},i\to a} = G_i(\dots) = \frac{1}{Z}\int x_i^2 m_{k\to i}(x_i)\mathcal{N}_{\mathsf{x},i}(\mu^{(l)}_{\mathsf{x},i\to a}, \sigma^{2(l)}_{\mathsf{x},i\to a})\mathrm{d}x_i - \mu^2_{\mathsf{x},i\to a}\,, \tag{2.62}$$

where Z is a normalization term. Note that the dependency on the target factor node a is introduced by the functions' arguments. For $K = N$ the message $m_{k\to i}(x_i)$ is the prior $f_{\mathsf{x},i}(x_i)$, which is one-dimensional in that case.
There are two more messages which have so far not been discussed, namely the message from variable node to prior factor (2.35) and from prior factor to variable node (2.35). If $K = N$, the message from variable node to prior factor is not needed and the message from prior factor to variable node is the prior factor itself (this can be verified using the message passing rules (2.28), (2.29)). For multi-dimensional priors they need to be taken into account, however.
The message from variable node with index i to prior factor node with index k, $m_{i\to k}(x_i)$ is similar to the product-term in (2.34). It is a product of normal distributions and as such again Gaussian. Mean and variance are given by (cf. (2.48), (2.49))

$$\frac{1}{\sigma^2_{\mathsf{x},i\to k}} = \sum_a \frac{1}{\sigma^2_{\mathsf{x},a\to i}} \tag{2.63}$$

$$\mu_{\mathsf{x},i\to k} = \sigma^2_{\mathsf{x},i\to k}\sum_a \frac{\mu_{\mathsf{x},a\to i}}{\sigma^2_{\mathsf{x},a\to i}}\,. \tag{2.64}$$

There are two differences between (2.63), (2.64) and (2.48), (2.49): firstly the message's target, which is the prior factor node with index k instead of the matrix factor with index a and secondly the fact that the sum ranges over all messages $m_{a\to i}$. These messages can be computed in a simplified way if the assumptions 1 to 5 on p.13 are justified. Repeating the process outlined in (2.50) to (2.57) results in

$$\sigma^2_{\mathsf{x},i\to k} = \frac{\sigma^2_{\mathsf{x},*\to i}}{L} \tag{2.65}$$

$$\approx \sigma^2_{\mathsf{w}_*} + \sigma^2_{\mathsf{z},*\to i} \tag{2.66}$$

$$\mu_{\mathsf{x},i\to k} = \sigma^2_{\mathsf{x},i\to k}\sum_a \frac{\mu_{\mathsf{x},a\to i}}{\sigma^2_{\mathsf{x},*\to i}} \tag{2.67}$$

$$\approx \sigma^2_{\mathsf{x},i\to k}\sum_a \frac{L(A_{a,i}\mu_{\mathsf{z},a\to i})}{L\sigma^2_{\mathsf{x},i\to k}} \tag{2.68}$$

$$= \sum_a A_{a,i}\mu_{\mathsf{z},a\to i} \,. \tag{2.69}$$

In fact, for the approximated values it is obvious that

$$\sigma^{2(l)}_{\mathsf{x},i\to a} = \sigma^2_{\mathsf{x},i\to k} \tag{2.70}$$

$$\mu^{(l)}_{\mathsf{x},i\to a} = \mu_{\mathsf{x},i\to k} - A_{a,i}\mu_{\mathsf{z},a\to i} \,. \tag{2.71}$$

The message from prior factor node to variable node $m_{k\to i}(x_i)$ is the marginal w.r.t. the target variable. It is computed as

$$m_{k\to i}(x_i) = \int \cdots \int_{j\neq i} f_{\mathbf{x}_k}(\boldsymbol{x}_k) \prod_{j\in\partial k\setminus i} m_{j\to k}(x_j)\mathrm{d}x_j \,. \tag{2.72}$$

The messages $m_{i\to k}$ are normal distributions with parameters $\mu_{\mathsf{x},i\to k}$, $\sigma^2_{\mathsf{x},i\to k}$. Inserting (2.72) into (2.61), (2.62) results in

$$\mu_{\mathsf{x},i\to a} = F_i(\dots) = \frac{1}{Z}\int \cdots \int x_i f_{\mathbf{x}_k}(\boldsymbol{x}_k)\mathcal{N}_{\mathbf{x}_k}(\boldsymbol{x}_k;\boldsymbol{\mu}^{(l)},\boldsymbol{\sigma}^{2(l)})\mathrm{d}\boldsymbol{x}_k \tag{2.73}$$

$$\sigma^2_{\mathsf{x},i\to a} = G_i(\dots) = \frac{1}{Z}\int \cdots \int x_i^2 f_{\mathbf{x}_k}(\boldsymbol{x}_k)\mathcal{N}_{\mathbf{x}_k}(\boldsymbol{x}_k;\boldsymbol{\mu}^{(l)},\boldsymbol{\sigma}^{2(l)})\mathrm{d}\boldsymbol{x}_k - \mu^2_{\mathsf{x},i\to a} \,, \tag{2.74}$$

$$\text{where } Z = \int \cdots \int f_{\mathbf{x}_k}(\boldsymbol{x}_k)\mathcal{N}_{\mathbf{x}_k}(\boldsymbol{x}_k;\boldsymbol{\mu}^{(l)},\boldsymbol{\sigma}^{2(l)})\mathrm{d}\boldsymbol{x}_k \,. \tag{2.75}$$

The functions F, G and Z depend on the arguments $(\boldsymbol{\mu}^{(l)},\boldsymbol{\sigma}^{2(l)})$. The dependency on the message's target is introduced by these arguments. Let the κ_k-dimensional prior factor node $f_{\mathsf{x}_k}(\boldsymbol{x}_k)$ depend on the random variables

$$\mathbf{x}_k = (\mathsf{x}_j, \mathsf{x}_{j+1}, \dots, \mathsf{x}_i, \dots, \mathsf{x}_{j+\kappa_k-1})^T. \tag{2.76}$$

Then, strictly following the message passing rules, the normal distribution $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\sigma}^2)$ in (2.73)-(2.75) has parameters

$$\boldsymbol{\mu}^{(l)} = (\mu_{\mathsf{x},j\to k}, \mu_{\mathsf{x},j+1\to k}, \dots, \mu^{(l)}_{\mathsf{x},i\to a}, \dots, \mu_{\mathsf{x},j+\kappa_k-1\to k})^T \tag{2.77}$$

$$\boldsymbol{\sigma}^{2(l)} = (\sigma^2_{\mathsf{x},j\to k}, \sigma^2_{\mathsf{x},j+1\to k}, \dots, \sigma^{2(l)}_{\mathsf{x},i\to a}, \dots, \sigma^2_{\mathsf{x},j+\kappa_k-1\to k})^T. \tag{2.78}$$

Note that the values $\mu_{\mathsf{x},i\to k}$, $\sigma^2_{\mathsf{x},i\to k}$ depend on the incoming message from the target of (2.73), (2.74) (which is the matrix factor node with index a). It is desirable to remove such feedback. The parameters $\boldsymbol{\mu}^{(l)}$, $\boldsymbol{\sigma}^{2(l)}$ should thus be entirely composed of $\mu^{(l)}_{\mathsf{x},i\to a}$, $\sigma^{2(l)}_{\mathsf{x},i\to a}$, which take the target (i.e. the matrix factor node with index a) into account by excluding its message, avoiding feedback (cf. (2.71)).
In conclusion, the functions $F_i(\dots)$ and $G_i(\dots)$ for multi-dimensional priors are defined by (2.73), (2.74). The computation of the message $m_{i\to k}$ can be omitted by reusing the values $\mu^{(l)}_{\mathsf{x},i\to a}$, $\sigma^{2(l)}_{\mathsf{x},i\to a}$. This has the side effect of improving the resulting algorithm. Due to the multi-dimensional prior, the functions $F_i(\dots)$ and $G_i(\dots)$ are considerably more complicated than in the case of i.i.d. variables x_i.

2.5.2 Vectorization

The message passing approximation outlined in Section 2.5.1 can be implemented more easily compared to a function-valued message passing algorithm. For some problems, it is useful to further simplify the algorithm. Due to the graph's structure, a number of $L \times N$ messages need to be computed in each iteration. Since $L = \rho N$, this scales with $O(N^2)$. In this section, it shall be assumed that the approximations 1 to 5 (p.13) hold and notation is simplified accordingly.
The means of messages between variable node and matrix factor node are given by (2.40) and (2.73), repeated here for convenience:

$$\mu_{\mathsf{z},a\to i} = y_a - \sum_{j\neq i} A_{a,j}\mu_{\mathsf{x},j\to a} \tag{2.79}$$

$$\mu_{\mathsf{x},i\to a} = F_i(\dots) = \frac{1}{Z}\int\cdots\int x_i f_{\mathbf{x}_k}(\boldsymbol{x}_k)\mathcal{N}_{\mathbf{x}_k}(\boldsymbol{\mu}^{(l)},\boldsymbol{\sigma}^{2(l)})\mathrm{d}\boldsymbol{x}_k\,. \tag{2.80}$$

The vector $\boldsymbol{\mu}^{(l)}$, which is the mean of the normal distribution $\mathcal{N}_{\mathbf{x}_k}(\boldsymbol{\mu}^{(l)},\boldsymbol{\sigma}^{2(l)})$ consists of entries (cf. (2.57))

$$\mu^{(l)}_{\mathsf{x},i\to a} = \sum_{b\neq a} A_{b,i}\mu_{\mathsf{z},b\to i} \tag{2.81}$$

at the i^{th} position, with the index a given by the target matrix factor index in (2.80). The goal is to achieve independence of the message's target. Messages are generally dependent on their target due to the fact that they exclude the message received from the target. This again is due to the message passing rules of the sum-product algorithm and can be thought of as feedback avoidance. It is not possible to work with non-targeted messages however. It is easy to show that this results in every node receiving and sending the same messages. Intuitively, as long as each node knows the last message they sent to a particular target, they can correct a non-targeted message using knowledge of their own message in the previous iteration as well as the remote node's way of operation. Such an approach is described in the following paragraphs.
Messages can be decomposed into several parts. One of these parts is independent of the target. The remaining terms consist of target-dependent terms, only one of which is deemed significant, i.e.

$$\mu_{\mathsf{z},a\to i} = \mu_{\mathsf{z},a} + \delta^{\mu}_{\mathsf{z},a\to i} + O(1/N) \tag{2.82}$$

$$\mu_{\mathsf{x},i\to a} = \mu_{\mathsf{x},i} + \delta^{\mu}_{\mathsf{x},i\to a} + O(1/N) \tag{2.83}$$

$$\mu^{(l)}_{\mathsf{x},i\to a} = \mu^{(l)}_{\mathsf{x},i} + \delta^{\mu(l)}_{\mathsf{x},i\to a} + O(1/N)\,. \tag{2.84}$$

In this section, the big-O-notation is used in the sense that

$$f(N) = O(g(N)) \;:\; \lim_{N\to\infty} \mathbb{E}\left\{|f(N)|\right\} \leq \varepsilon g(N), \quad \varepsilon \in \mathbb{R}^+\,, \tag{2.85}$$

where $g(N) = 1/N$ usually. Thus, terms that grow slower than ε/N (with $\varepsilon \approx 1$) as N becomes large are denoted $O(1/N)$. The decomposition is performed by inserting (2.82) to (2.84) into (2.79) to (2.81) and identifying terms based on their magnitude and dependence on the message's target:

$$\mu_{\mathsf{z},a\to i} = y_a - \sum_{j\neq i} A_{a,j}\mu_{\mathsf{x},j\to a} \tag{2.86}$$

$$= y_a - \sum_{j} A_{a,j}(\mu_{\mathsf{x},j} + \delta^{\mu}_{\mathsf{x},j\to a} + O(1/N)) + A_{a,i}(\mu_{\mathsf{x},i} + \delta^{\mu}_{\mathsf{x},i\to a} + O(1/N)) \tag{2.87}$$

$$= \underbrace{y_a - \sum_{j} A_{a,j}(\mu_{\mathsf{x},j} + \delta^{\mu}_{\mathsf{x},j\to a})}_{\mu_{\mathsf{z},a}} + \underbrace{A_{a,i}\mu_{\mathsf{x},i}}_{\delta^{\mu}_{\mathsf{z},a\to i}} \tag{2.88}$$

$$\underbrace{- \sum_{j} A_{a,j}O(1/N) + A_{a,i}(\delta^{\mu}_{\mathsf{x},i\to a} + O(1/N))}_{O(1/N)} \tag{2.89}$$

$$\mu^{(l)}_{\mathsf{x},i\to a} = \sum_{b\neq a} A_{b,i}\mu_{\mathsf{z},b\to i} \tag{2.90}$$

$$= \sum_{b} A_{b,i}(\mu_{\mathsf{z},b} + \delta^{\mu}_{\mathsf{z},b\to i} + O(1/N)) - A_{a,i}(\mu_{\mathsf{z},a} + \delta^{\mu}_{\mathsf{z},a\to i} + O(1/N)) \tag{2.91}$$

$$= \underbrace{\sum_{b} A_{b,i}(\mu_{\mathsf{z},b} + \delta^{\mu}_{\mathsf{z},b\to i})}_{\mu^{(l)}_{\mathsf{x},i}} \underbrace{- A_{a,i}\mu_{\mathsf{z},a}}_{\delta^{\mu(l)}_{\mathsf{x},i\to a}} \tag{2.92}$$

$$+ \underbrace{\sum_{b} A_{b,i}O(1/N) - A_{a,i}\delta^{\mu}_{\mathsf{z},a\to i} - A_{a,i}O(1/N)}_{O(1/N)} . \tag{2.93}$$

Note that (2.91) is the decomposition of the arguments $\mu^{(l)}_{\mathsf{x},i\to a}$ of $F(\dots)$. This gives rise to a Taylor expansion of $F(\dots)$:

$$\mu_{\mathsf{x},i\to a} \approx F(\mu^{(l)}_{\mathsf{x},i}) + \delta^{\mu(l)}_{\mathsf{x},i\to a}\frac{\partial F(\mu^{(l)}_{\mathsf{x},i})}{\partial \mu^{(l)}_{\mathsf{x},i}} + O(1/N) \tag{2.94}$$

$$= \underbrace{F(\mu^{(l)}_{\mathsf{x},i})}_{\mu_{\mathsf{x},i}} \underbrace{- A_{a,i}\mu_{\mathsf{z},a}\frac{\partial F(\mu^{(l)}_{\mathsf{x},i})}{\partial \mu^{(l)}_{\mathsf{x},i}}}_{\delta^{\mu}_{\mathsf{x},i\to a}} + O(1/N) . \tag{2.95}$$

The Taylor expansion is accurate for many functions $F(\dots)$ since $\delta^{\mu(l)}_{\mathsf{x},i\to a}$ is of $O(L^{-1/2})$ and thus much smaller than the argument $\mu^{(l)}_{\mathsf{x},i}$. The expression (2.95)

is the decomposition for one-dimensional prior factors, i.e. $K = N$. For multi-dimensional priors, the function $F(\dots)$ depends on multiple local means $\mu^{(l)}_{\mathsf{x},i\to a}$, each with its decomposition according to (2.91):

$$\mu_{\mathsf{x},i\to a} \approx F_i(\boldsymbol{\mu}^{(l)}_{\mathbf{x}_k}) + \sum_{j\in\mathcal{K}} \delta^{\mu(l)}_{\mathsf{x},j\to a} \frac{\partial F_i(\boldsymbol{\mu}^{(l)}_{\mathbf{x}_k})}{\partial \mu^{(l)}_{\mathsf{x}_j}} + O(1/N) \tag{2.96}$$

$$= \underbrace{F_i(\boldsymbol{\mu}^{(l)}_{\mathbf{x}_k})}_{\mu_{\mathsf{x},i}} \underbrace{- \sum_{j\in\mathcal{K}} A_{a,j}\mu_{\mathsf{z},a} \frac{\partial F_i(\boldsymbol{\mu}^{(l)}_{\mathbf{x}_k})}{\partial \mu^{(l)}_{\mathsf{x}_j}}}_{\delta^{\mu}_{\mathsf{x},i\to a}} + O(1/N)\,. \tag{2.97}$$

In (2.97), the set $\mathcal{K}$ contains all indices j of variables x_j that appear in the vector $\boldsymbol{x}_k$ (i.e. variables that depend on the same prior factor). It remains to insert the various terms at the appropriate places. The goal is to have expressions that only depend on $\mu_{\mathsf{z},a}$ and $\mu_{\mathsf{x},i}$. Starting with (2.91), substitution yields

$$\mu^{(l)}_{\mathsf{x},i} = \sum_b A_{b,i}(\mu_{\mathsf{z},b} + \delta^{\mu}_{\mathsf{z},b\to i}) \tag{2.98}$$

$$= \sum_b A_{b,i}\mu_{\mathsf{z},b} + \sum_b A_{b,i}A_{b,i}\mu_{\mathsf{x},i} \tag{2.99}$$

$$= \sum_b A_{b,i}\mu_{\mathsf{z},b} + \mu_{\mathsf{x},i} \underbrace{\sum_b A^2_{b,i}}_{=1}\,. \tag{2.100}$$

In (2.100), the second sum evaluates to one due to normalized columns of $\boldsymbol{A}$. The expression can be written as

$$\boldsymbol{\mu}^{(l)}_{\mathbf{x}} = \boldsymbol{A}^T\boldsymbol{\mu}_{\boldsymbol{z}} + \boldsymbol{\mu}_{\mathbf{x}}\,. \tag{2.101}$$

Continuing with (2.87), substitution yields (for $K = N$)

$$\mu_{\mathsf{z},a} = y_a - \sum_j A_{a,j}(\mu_{\mathsf{x},j} + \delta^{\mu}_{\mathsf{x},j\to a}) \tag{2.102}$$

$$= y_a - \sum_j A_{a,j}\mu_{\mathsf{x},j} + \sum_j A_{a,j}A_{a,j}\mu_{\mathsf{z},a} \frac{\partial F(\mu^{(l)}_{\mathsf{x},j})}{\partial \mu^{(l)}_{\mathsf{x}_j}} \tag{2.103}$$

$$= y_a - \sum_j A_{a,j}\mu_{\mathsf{x},j} + \mu_{\mathsf{z},a} \sum_j \underbrace{A^2_{a,j}}_{\approx 1/L} \frac{\partial F(\mu^{(l)}_{\mathsf{x},j})}{\partial \mu^{(l)}_{\mathsf{x}_j}} \tag{2.104}$$

$$\approx y_a - \sum_j A_{a,j}\mu_{\mathsf{x},j} + \frac{\mu_{\mathsf{z},a}}{L} \sum_j \frac{\partial F(\mu^{(l)}_{\mathsf{x},j})}{\partial \mu^{(l)}_{\mathsf{x}_j}}\,. \tag{2.105}$$

The approximation $A^2_{a,j} \approx 1/L$ can be omitted at the cost of an additional matrix-vector multiplication per iteration. Note that $\mu_{\mathsf{z},a}$ which appears in (2.105) is the

previous iteration's value. This is indicated using the iteration index $[t]$ in the vector-valued expression

$$\boldsymbol{\mu}_{\mathsf{z}}^{[t]} = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{\mu}_{\mathsf{x}} + \boldsymbol{\mu}_{\mathsf{z}}^{[t-1]} \otimes \left(\boldsymbol{B}\,\mathrm{diag}(\nabla \boldsymbol{F}(\boldsymbol{\mu}_{\mathsf{x}}^{(l)}))\right). \tag{2.106}$$

In (2.106), the operator $\otimes$ denotes component-wise multiplication, i.e. in $\boldsymbol{c} = \boldsymbol{a} \otimes \boldsymbol{b}$, the vector $\boldsymbol{c}$ is composed of entries $c_i = a_i b_i$. The matrix $\boldsymbol{B}$ has entries $B_{a,i} = A_{a,i}^2$. The expression $\nabla \boldsymbol{F}(\boldsymbol{\mu}_{\mathsf{x}}^{(l)})$ evaluates to the Jacobian of $\boldsymbol{F}(\dots)$, whose diagonal has entries

$$\mathrm{diag}(\nabla \boldsymbol{F}(\boldsymbol{\mu}_{\mathsf{x}}^{(l)}))_i = \frac{\partial F(\mu_{\mathsf{x},i}^{(l)})}{\partial \mu_{\mathsf{x},i}^{(l)}}. \tag{2.107}$$

If the approximation $A_{a,i}^2 \approx 1/L$ is applied, (2.106) becomes

$$\boldsymbol{\mu}_{\mathsf{z}} = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{\mu}_{\mathsf{x}} + \frac{1}{L}\boldsymbol{\mu}_{\mathsf{z}} \left(\mathbf{1}_N^T\,\mathrm{diag}(\nabla \boldsymbol{F}(\boldsymbol{\mu}_{\mathsf{x}}^{(l)}))\right) \tag{2.108}$$

$$= \boldsymbol{y} - \boldsymbol{A}\boldsymbol{\mu}_{\mathsf{x}} + \frac{\boldsymbol{\mu}_{\mathsf{z}}}{L} \sum_i \frac{\partial F(\mu_{\mathsf{x},i}^{(l)})}{\partial \mu_{\mathsf{x},i}^{(l)}}. \tag{2.109}$$

The notation used in (2.106) and (2.108) uses $F(\dots)$ as a vector-valued function, i.e. its argument and results are both N-dimensional vectors. However both expressions still only apply to one-dimensional ($K = N$) prior factors. Each row of $\boldsymbol{F}(\boldsymbol{\mu}_{\mathsf{x}}^{(l)})$ is a scalar-valued function with the scalar argument $\mu_{\mathsf{x},i}^{(l)}$:

$$\boldsymbol{F}(\boldsymbol{\mu}_{\mathsf{x}}^{(l)}) = \begin{pmatrix} F_1(\mu_{\mathsf{x},1}^{(l)}) \\ F_2(\mu_{\mathsf{x},2}^{(l)}) \\ \vdots \\ F_N(\mu_{\mathsf{x},N}^{(l)}) \end{pmatrix}. \tag{2.110}$$

In case the x_i are not identically distributed (but still independent, i.e. the prior $f_{\mathbf{x}}(\boldsymbol{x})$ can be fully factored, $K = N$), then $F_i(\dots)$ are different functions. For multi-dimensional prior factors $f_{\mathbf{x}_k}(\boldsymbol{x}_k)$, the expression $\delta_{\mathsf{x},i\to a}^{\mu}$ from (2.97) must be inserted into (2.88). The derivation is conceptually similar to (2.105), however there is an additional sum over $j \in \mathcal{K}$ involved:

$$\mu_{\mathsf{z},a} = y_a - \sum_i A_{a,i}(\mu_{\mathsf{x},i} + \delta_{\mathsf{x},i\to a}^{\mu}) \tag{2.111}$$

$$= y_a - \sum_i A_{a,i}\mu_{\mathsf{x},i} + \sum_i A_{a,i} \sum_{j\in\mathcal{K}} A_{a,j}\mu_{\mathsf{z},a} \frac{\partial F_i(\boldsymbol{\mu}_{\mathbf{x}_k}^{(l)})}{\partial \mu_{\mathsf{x}_j}^{(l)}} \tag{2.112}$$

$$= y_a - \sum_i A_{a,i}\mu_{\mathsf{x},i} + \mu_{\mathsf{z},a} \sum_i A_{a,i} \sum_{j\in\mathcal{K}} A_{a,j} \frac{\partial F_i(\boldsymbol{\mu}_{\mathbf{x}_k}^{(l)})}{\partial \mu_{\mathsf{x}_j}^{(l)}}. \tag{2.113}$$

The expression (2.113) can be simplified if $\mathbb{E}\{A_{a,i}A_{a,j}\}|_{j\neq i} = 0$:

$$\mu_{\mathsf{z},a} = y_a - \sum_i A_{a,i}\mu_{\mathsf{x},i} + \mu_{\mathsf{z},a}\sum_i A_{a,i}^2 \frac{\partial F_i(\boldsymbol{\mu}_{\mathbf{x}_k}^{(l)})}{\partial \mu_{\mathsf{x},i}^{(l)}} \tag{2.114}$$

$$+ \mu_{z,a}\underbrace{\sum_i \sum_{j\in\mathcal{K}\setminus i} A_{a,i}A_{a,j}\frac{\partial F_i(\boldsymbol{\mu}_{\mathbf{x}_k}^{(l)})}{\partial \mu_{\mathsf{x}_j}^{(l)}}}_{\approx 0} . \tag{2.115}$$

Finally, using $\mathbb{E}\{A_{a,i}^2\} \approx 1/L$, it is possible to write (2.114) as

$$\mu_{\mathsf{z},a} = y_a - \sum_i A_{a,i}\mu_{\mathsf{x},i} + \frac{\mu_{\mathsf{z},a}}{L}\sum_i \frac{\partial F_i(\boldsymbol{\mu}_{\mathbf{x}_k}^{(l)})}{\partial \mu_{\mathsf{x},i}^{(l)}} \tag{2.116}$$

$$\tag{2.117}$$

or, using vector-valued notation,

$$\boldsymbol{\mu}_{\mathbf{z}}^{[t]} = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} + \frac{\boldsymbol{\mu}_{\mathbf{z}}^{[t-1]}}{L}\sum_i \frac{\partial F_i(\boldsymbol{\mu}_{\mathbf{x}_k}^{(l)})}{\partial \mu_{\mathsf{x},i}^{(l)}} . \tag{2.118}$$

2.5.3 Algorithm

Corresponding to the two main steps in the derivation of approximate message passing, namely approximation by Gaussian functions and vectorization, two reconstruction algorithms can be formulated. The Gaussian message passing algorithm for multi-dimensional priors (MD-GMP) does not include the vectorization approximation. It is listed in Appendix A, p.113 as Algorithm 6. The approximate message passing algorithm for multi-dimensional priors is listed on p.114 as Algorithm 7. It uses both Gaussian approximation and vectorization which results in a computationally more efficient algorithm.

Note that MD-GMP is a message passing algorithm, it computes targeted messages. MD-BAMP on the other hand computes local states, using these instead of targeted messages to calculate updates and compensates for the error with a first-order Taylor expansion. The complexity of MD-GMP thus scales with the number of edges in the graph, while MD-BAMP seemingly scales with the number of nodes. This is not entirely true though – the complexity of MD-BAMP is still $O(NL)$ due to the matrix-vector multiplications. Furthermore, many expressions in MD-GMP are similar and can be computed efficiently.

The algorithm listing for MD-GMP is longer than for MD-BAMP mostly since a state update for the estimation of $\boldsymbol{x}$ as well as targeted messages have to be computed in each iteration. Furthermore, variance tracking is more complex.

2.5.4 State Evolution

The behavior of classical approximate message passing [23] for large dimension $N, L \to \infty$ is characterized by a framework called *state evolution* [3, 24]. An extension of state evolution which applies to signals with multivariate priors, yielding non-separable denoisers, can be found in [5]. The proof holds for Lipschitz continuous functions F. For finite entries of $\boldsymbol{\sigma}_{\mathbf{x}_k}^{2(l)}$ and $f_{\mathbf{x}_k}(\boldsymbol{x}_k)$ a pdf with limited second order moments, these functions are always Lipschitz, i.e.

$$\left\|\boldsymbol{F}(\boldsymbol{\mu}_1, \sigma^2) - \boldsymbol{F}(\boldsymbol{\mu}_2, \sigma^2)\right\|_2 \leq K \left\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right\|_2 , \tag{2.119}$$

with the Lipschitz constant

$$K = \sup_{\boldsymbol{v},\boldsymbol{\mu}} \left|\boldsymbol{v}^T \mathrm{diag}(\nabla \boldsymbol{F}(\boldsymbol{\mu}, \sigma^2))\right| < \infty \quad \|\boldsymbol{v}\|_2 = 1,\ \sigma^2 > 0 . \tag{2.120}$$

This can be shown using the mean value theorem. State evolution for MD-BAMP is defined by the recursion

$$\sigma_{\mathsf{x},[t]}^2 = \mathbb{E}\left\{\frac{1}{\kappa_k}\left\|\boldsymbol{F}(\boldsymbol{x}_0 + \sigma_{\mathsf{x},[t-1]}^{(l)}\mathbf{n}) - \boldsymbol{x}_0\right\|_2^2\right\} \tag{2.121}$$

$$\sigma_{\mathsf{x},[t]}^{2(l)} = \sigma_{\mathsf{w}}^2 + \rho^{-1}\sigma_{\mathsf{x},[t]}^2 = \sigma_{\mathsf{w}}^2 + \sigma_{\mathsf{z}}^2 , \tag{2.122}$$

where $\mathbf{n}$ is a random vector whose entries n_i are distributed according to the standard normal distribution and $\boldsymbol{x}_0$ is the true value of the random vector $\mathbf{x}$.

2.6 BAMP for Variable Measurement Noise

Motivation. In many practical settings, measurement noise is involved, e.g. in applications like compressive imaging or compressed coded transmission systems [52, 8]. In many of these applications, measurement noise is not i.i.d. Another practically important case is that of compressive sensing systems with measurement matrices where the rows are scaled by individual scaling factors s_a. In this case, the variance of the sensing matrice's entries depends on the row and is in general different from L^{-1}. A simple linear transformation in the form of a diagonal matrix $\boldsymbol{T}$ with entries $T_{a,a} = s_a^{-1}$ applied to the measurements and the sensing matrix transforms the problem such that it satisfies the requirements of MD-BAMP. The entries of the noise vector $\boldsymbol{w}$ are then scaled and the noise appears to have variable variance.

Prior work. The noise sensitivity of CS systems in presence of i.i.d. noise has been discussed extensively for linear measurements of sparse signals [25] and more generally in [66]. While the literature deals with a range of special types of noise, such as in 1-bit compressive sensing [15] and general noisy quantized CS [70], measurement noise is usually assumed to be i.i.d. and algorithms do not exploit more detailed noise models. The MD-GMP algorithm as well as GMP for i.i.d. entries of $\boldsymbol{x}$ (which can be extracted from the derivation of AMP in [43], for example) intrinsically allow for variable (although not correlated) measurement noise, due to the fact that they track the variances of each message separately. These algorithms have high computational complexity, however. It is thus desirable to find a method that is both fast and enables the use of detailed information about the structure of the noise. The algorithm, its derivation, discussion and simulation as well as state evolution are novel results that were *first published in the Proceedings of the 26th European Signal Processing Conference (EUSIPCO-2018) in 2018, published by EURASIP* [7]. The algorithm published at EUSIPCO was formulated as extension for classical Bayesian AMP while this section extends MD-BAMP. Furthermore, a new, simple formalism allowing for correlated noise is presented in this section.

Derivation. In order to perform vectorization in Section 2.5, the dependence of $\sigma^2_{\mathsf{x},i\to a}$ on a was dropped in (2.50). Implicitly, this also introduced the requirement that $\sigma^2_{\mathsf{w},a} = \sigma^2_{\mathsf{w}}$, i.e. that all noise variances are equal. Since this assumption is restrictive, this section shall discuss a modified version of MD-BAMP that is able to handle variable measurement noise.
In this section, the noise vector $\mathbf{w}$ is assumed to be distributed according to

$$\mathbf{w} \sim \prod_a \mathcal{N}(0, \sigma^2_{\mathsf{w},a}) \,. \tag{2.123}$$

For the derivation it is assumed that the number of different noise variances might be smaller than L. It shall be shown that this allows for minimization of compu-

tational complexity. Thus, $|\mathcal{V}|$ different variances $\sigma^2_{\mathsf{w},a}$ are presumed to exist. The set $\mathcal{V}$ is a set of sets, i.e.

$$\mathcal{V} = \{\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_K\}\,, \tag{2.124}$$

with a set $\mathcal{W}_k$ containing all random variables w_a with identical variance. The variance of variables in $\mathcal{W}_k$ shall be denoted $\sigma^2_{\mathsf{w},k}$. Equivalently, and slightly abusing notation, the set $\mathcal{W}_k$ contains all indices a for which the random variables w_a have identical variance $\sigma^2_{\mathsf{w},k}$. Subsequently, the derivation of Bayesian approximate message passing for variable noise variance (BAMP-VN) is presented. The message passing rules as well as the message's Gaussian approximation are identical to the one presented in Section 2.5. For the variances, an iteration according to the message passing rules is defined in (2.48), (2.62) and (2.41), repeated here for convenience:

$$\sigma^{2(l)}_{\mathsf{x},i\to a} = \left(\sum_{b\neq a} \frac{1}{\sigma^2_{\mathsf{x},b\to i}}\right)^{-1} = \left(\sum_{b\neq a} \frac{A^2_{b,i}}{\sigma^2_{\mathsf{z},b\to i} + \sigma^2_{\mathsf{w},b}}\right)^{-1} \tag{2.125}$$

$$\sigma^2_{\mathsf{x},i\to a} = G(\mu^{(l)}_{\mathsf{x},i\to a}, \sigma^{2(l)}_{\mathsf{x},i\to a}) \tag{2.126}$$

$$\sigma^2_{\mathsf{z},a\to i} = \sum_{j\neq i} A^2_{a,j}\sigma^2_{\mathsf{x},j\to a}\,. \tag{2.127}$$

Using $A^2_{a,i} \approx L^{-1}$ again (cf. (2.52) and condition 5 on p.13), it is possible to write

$$\sigma^{2(l)}_{\mathsf{x},i\to a} \approx L\left(\sum_{b\neq a} \frac{1}{\sigma^2_{\mathsf{z},b\to i} + \sigma^2_{\mathsf{w},b}}\right)^{-1} \tag{2.128}$$

$$\sigma^2_{\mathsf{z},a\to i} \approx \frac{1}{L}\sum_{j\neq i} \sigma^2_{\mathsf{x},j\to a}\,. \tag{2.129}$$

Again it is assumed that all $\sigma^2_{\mathsf{z},a\to i}$ and $\sigma^2_{\mathsf{x},i\to a}$ are approximately equal. Then,

$$\sigma^2_{\mathsf{z},a\to i} \approx \sigma^2_{\mathsf{z}} = \frac{N-1}{NL}\sum_i \sigma^2_{\mathsf{x},i} \approx \frac{1}{L}\sum_i \sigma^2_{\mathsf{x},i} \tag{2.130}$$

$$\sigma^{2(l)}_{\mathsf{x},*\to a} = L\left(\sum_b \frac{1}{\sigma^2_{\mathsf{z}} + \sigma^2_{\mathsf{w},b}} - \frac{1}{\sigma^2_{\mathsf{z}} + \sigma^2_{\mathsf{w},a}}\right)^{-1} \tag{2.131}$$

$$\approx \sigma^{2(l)}_{\mathsf{x}} = L\left(\sum_a \frac{1}{\sigma^2_{\mathsf{z}} + \sigma^2_{\mathsf{w},a}}\right)^{-1}. \tag{2.132}$$

Let $L \gg 1$, then the step from (2.131) to (2.132) is a valid approximation. The omitted term is small compared to the $|\mathcal{W}_k| - 1$ identical terms remaining in the sum. The biggest approximation error happens if $\sigma^2_{\mathsf{w},a} = 0$, i.e. a particular measurement sample y_a is unaffected by noise. The approximation (2.132) is then

overly "optimistic", i.e. the estimated value of $\sigma_{\mathsf{x}}^{2(l)}$ is too small. Identical noise variances can be used to minimize computational cost. Expression (2.132) can be written as

$$\sigma_{\mathsf{x}}^{2(l)} = L \left(\sum_{\mathcal{W}_k \in \mathcal{V}} \sum_{b \in \mathcal{W}_k} \frac{1}{\sigma_{w,k}^2 + \sigma_{\mathsf{z}}^2} \right)^{-1} \tag{2.133}$$

$$= L \left(\sum_{\mathcal{W}_k \in \mathcal{V}} \frac{|\mathcal{W}_k|}{\sigma_k^2} \right)^{-1} , \tag{2.134}$$

with $\sigma_k^2 = \sigma_{\mathsf{w},k}^2 + \sigma_{\mathsf{z}}^2$. Note that (2.134) only requires $|\mathcal{V}|+1$ divisions. If $|\mathcal{W}_k| \gg 1$, then due to $\sum_k |\mathcal{W}_k| = L$, $|\mathcal{V}|$ needs to be small and thus the computational complexity is low.

For expectations, one message passing iteration is defined by (2.40), (2.46) and (2.57), repeated here:

$$\mu_{\mathsf{z},a\to i} = y_a - \sum_{j \neq i} A_{a,j} \mu_{\mathsf{x},j\to a} \tag{2.135}$$

$$\mu_{\mathsf{x},a\to i} = \frac{\mu_{\mathsf{z},a\to i}}{A_{a,i}} \approx L(A_{a,i} \mu_{\mathsf{z},a\to i}) \tag{2.136}$$

$$\mu_{\mathsf{x},i\to a}^{(l)} = \sigma_{\mathsf{x},i\to a}^{2(l)} \sum_{b \neq a} \frac{\mu_{\mathsf{x},b\to i}}{\sigma_{\mathsf{x},b\to i}^2} \tag{2.137}$$

$$\approx \sigma_{\mathsf{x},i\to a}^{2(l)} \sum_{b \neq a} \frac{L(A_{a,i}\mu_{\mathsf{z},a\to i})}{L(\sigma_{\mathsf{z}}^2 + \sigma_{\mathsf{w},a}^2)} . \tag{2.138}$$

Writing the sum in (2.138) in terms of $\mathcal{W}_k$ results in

$$\mu_{\mathsf{x},i\to a}^{(l)} \approx \frac{\sigma_{\mathsf{x}}^{2(l)}}{L} \left(\sum_{\mathcal{W}_k \in \mathcal{V}} \sum_{b \in \mathcal{W}_k} \frac{\mu_{\mathsf{x},b\to i}}{\sigma_{\mathsf{w},b}^2 + \sigma_{\mathsf{z}}^2} - \frac{\mu_{\mathsf{x},b\to i}}{\sigma_{\mathsf{w},a}^2 + \sigma_{\mathsf{z}}^2} \right) \tag{2.139}$$

$$= \frac{\sigma_{\mathsf{x}}^{2(l)}}{L} \left(\sum_{\mathcal{W}_k \in \mathcal{V}} \frac{1}{\sigma_k^2} \sum_{b \in \mathcal{W}_k} \mu_{\mathsf{x},b\to i} - \frac{\mu_{\mathsf{x},a\to i}}{\sigma_{\mathsf{w},a}^2 + \sigma_{\mathsf{z}}^2} \right) \tag{2.140}$$

$$\approx \sigma_{\mathsf{x}}^{2(l)} \left(\sum_{\mathcal{W}_k \in \mathcal{V}} \frac{1}{\sigma_k^2} \sum_{b \in \mathcal{W}_k} A_{b,i} \mu_{\mathsf{z},b\to i} - \frac{A_{a,i}\mu_{\mathsf{z},a\to i}}{\sigma_{\mathsf{w},a}^2 + \sigma_{\mathsf{z}}^2} \right) . \tag{2.141}$$

While the variances have already lost their "targeted" message-passing character (cf. (2.130), (2.132)), expectations are still updated using message-passing rules, requiring computation of LN values in each iteration. Similarly to (2.84), the message (2.141) is dissected:

$$\mu_{\mathsf{x},i\to a}^{(l)} = \mu_{\mathsf{x},i}^{(l)} + \delta_{\mathsf{x},i\to a}^{\mu(l)} + O(1/N) , \tag{2.142}$$

using $\mu_{\mathsf{z},a\to i} = \mu_{\mathsf{z},a} + \delta^{\mu}_{\mathsf{z},a\to i} + O(1/N)$:

$$\begin{aligned}\mu^{(l)}_{\mathsf{x},i\to a} &= \sum_{\mathcal{W}_k\in\mathcal{V}} \frac{\sigma^{2(l)}_{\mathsf{x}}}{\sigma^2_k} \sum_{b\in\mathcal{W}_k} A_{b,i}(\mu_{\mathsf{z},b} + \delta^{\mu}_{\mathsf{z},b\to i} + O(1/N)) \\ &\quad - \frac{\sigma^{2(l)}_{\mathsf{x}}}{\sigma^2_k} A_{a,i}(\mu_{\mathsf{z},a} + \delta^{\mu}_{\mathsf{z},a\to i} + O(1/N)) \\ &= \underbrace{\sum_{\mathcal{W}_k\in\mathcal{V}} \frac{\sigma^{2(l)}_{\mathsf{x}}}{\sigma^2_k} \sum_{b\in\mathcal{W}_k} A_{b,i}(\mu_{\mathsf{z},b} + \delta^{\mu}_{\mathsf{z},b\to i})}_{\mu^{(l)}_{\mathsf{x},i}} \qquad (2.143) \\ &\quad \underbrace{-\frac{\sigma^{2(l)}_{\mathsf{x}}}{\sigma^2_k} A_{a,i}\mu_{\mathsf{z},a}}_{\delta^{\mu(l)}_{\mathsf{x},i\to a}} + \underbrace{\dots}_{O(1/N)} . \qquad (2.144)\end{aligned}$$

Again, terms of magnitude $O(1/N)$ are neglected, thereby approximating message passing. Furthermore, since $\delta^{\mu}_{\mathsf{z},a\to i} = A_{a,i}\mu_{\mathsf{x},i}$ (cf. (2.88)), (2.143) can be reformulated as

$$\begin{aligned}\mu^{(l)}_{\mathsf{x},i} &= \sum_{\mathcal{W}_k\in\mathcal{V}} \frac{\sigma^{2(l)}_{\mathsf{x}}}{\sigma^2_k} \sum_{b\in\mathcal{W}_k} A_{b,i}\mu_{\mathsf{z},b} + \sum_{\mathcal{W}_k\in\mathcal{V}} \frac{\sigma^{2(l)}_{\mathsf{x}}}{\sigma^2_k} \sum_{b\in\mathcal{W}_k} \underbrace{A^2_{b,i}}_{\approx L^{-1}} \mu_{\mathsf{x},i} \qquad (2.145) \\ &= \sum_{\mathcal{W}_k\in\mathcal{V}} \frac{\sigma^{2(l)}_{\mathsf{x}}}{\sigma^2_k} \sum_{b\in\mathcal{W}_k} A_{b,i}\mu_{\mathsf{z},b} + \mu_{\mathsf{x},i}\frac{\sigma^{2(l)}_{\mathsf{x}}}{L} \sum_{\mathcal{W}_k\in\mathcal{V}} \frac{|\mathcal{W}_k|}{\sigma^2_k} . \qquad (2.146)\end{aligned}$$

Developing (2.146) by taking into account (2.134), $\sigma^{2(l)}_{\mathsf{x}}$ cancels down and thus

$$\mu^{(l)}_{\mathsf{x},i} = \sum_{\mathcal{W}_k\in\mathcal{V}} \frac{\sigma^{2(l)}_{\mathsf{x}}}{\sigma^2_k} \sum_{b\in\mathcal{W}_k} A_{b,i}\mu_{\mathsf{z},b} + \mu_{\mathsf{x},i} \qquad (2.147)$$

$$\text{with } \mu'_{\mathsf{z},a} = \frac{\sigma^{2(l)}_{\mathsf{x}}}{\sigma^2_k}\mu_{\mathsf{z},a} , \quad a \in \mathcal{W}_k \qquad (2.148)$$

$$\boldsymbol{\mu}^{(l)}_{\mathbf{x}} = \boldsymbol{A}^T\boldsymbol{z}' + \boldsymbol{\mu}_{\mathbf{x}} . \qquad (2.149)$$

The expression (2.147) is similar to $\boldsymbol{\mu}^{(l)}_{\mathbf{x}} = \boldsymbol{A}^T\boldsymbol{z} + \boldsymbol{\mu}_{\mathbf{x}}$ in MD-BAMP and indeed identical for $|\mathcal{V}| = 1$ (i.e. identical noise variances). It remains to deploy the term $\delta^{\mu(l)}_{\mathsf{x},i\to a}$ from (2.144) in the expression for $\mu_{\mathsf{x},i\to a}$:

$$\mu_{\mathsf{x},i\to a} \approx F(\mu^{(l)}_{\mathsf{x},i}, \sigma^{2(l)}_{\mathsf{x}}) + \delta^{\mu(l)}_{\mathsf{x},i\to a}\frac{\partial F(\mu^{(l)}_{\mathsf{x},i}, \sigma^{2(l)}_{\mathsf{x}})}{\partial \mu^{(l)}_{\mathsf{x},i}} + O(1/N) \qquad (2.150)$$

$$= \underbrace{F(\mu_{\mathsf{x},i}^{(l)}, \sigma_{\mathsf{x}}^{2(l)})}_{\mu_{\mathsf{x},i}} \underbrace{-\mu'_{\mathsf{z},a} A_{a,i} \frac{\partial F(\mu_{\mathsf{x},i}^{(l)}, \sigma_{\mathsf{x}}^{2(l)})}{\partial \mu_{\mathsf{x},i}^{(l)}}}_{\delta^{\mu}_{\mathsf{x},i\to a}} + O(1/N) \,. \tag{2.151}$$

Finally, $\delta^{\mu}_{\mathsf{x},i\to a}$ appears in $\mu_{\mathsf{z},a}$:

$$\mu_{\mathsf{z},a} = y_a - \sum_j A_{a,j}(\mu_{\mathsf{x},j} + \delta^{\mu}_{\mathsf{x},j\to a}) \tag{2.152}$$

$$= y_a - \sum_j A_{a,j}\mu_{\mathsf{x},j} + \sum_j A_{a,j}^2 \mu'_{\mathsf{z},a} \frac{\partial F(\mu_{\mathsf{x},j}^{(l)}, \sigma_{\mathsf{x}}^{2(l)})}{\partial \mu_{\mathsf{x},j}^{(l)}} \tag{2.153}$$

$$= y_a - \sum_j A_{a,j}\mu_{\mathsf{x},j} + \frac{\mu'_{\mathsf{z},a}}{L} \sum_j \frac{\partial F(\mu_{\mathsf{x},j}^{(l)}, \sigma_{\mathsf{x}}^{2(l)})}{\partial \mu_{\mathsf{x},j}^{(l)}} \,. \tag{2.154}$$

For multi-dimensional priors, the steps (2.150)-(2.154) are analogous to (2.111)-(2.118) with $\mu_{\mathsf{z},a}$ replaced by $\mu'_{\mathsf{z},a}$ as defined in (2.148).

2.6.1 Correlated Noise

Certain problems might exhibit correlated noise. Let $\mathbb{E}\{\mathbf{w}\} = \mathbf{0}$, then $\boldsymbol{C} = \mathbb{E}\{\mathbf{w}\mathbf{w}^T\}$ is not diagonal if the entries of $\mathbf{w}$ are correlated. It is tempting to use a whitening transformation to decorrelate $\mathbf{w}$, however this would change the statistics of the sensing matrix $\boldsymbol{A}$: let

$$\mathbf{w} = \boldsymbol{U}\mathbf{u} \,, \tag{2.155}$$

where $\mathbf{u}$ is a zero-mean random vector with $\mathbb{E}\{\mathbf{u}\mathbf{u}^T\} = \boldsymbol{I}$ and $\boldsymbol{U}$ describes a (rank-preserving) linear transformation. The covariance matrix of $\mathbf{w}$ is

$$\boldsymbol{C} = \mathbb{E}\{\boldsymbol{w}\boldsymbol{w}^T\} = \boldsymbol{U}\boldsymbol{U}^T \,, \tag{2.156}$$

with Eigenvalue decomposition

$$\boldsymbol{C} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^H = \boldsymbol{D}\boldsymbol{D}^T \tag{2.157}$$

$$\boldsymbol{D} = \boldsymbol{V}\boldsymbol{\Lambda}^{\frac{1}{2}} \,. \tag{2.158}$$

Attempting to whiten the noise results in the vector

$$\boldsymbol{v} = \boldsymbol{D}^{-1}\boldsymbol{y} = \boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{x} + \boldsymbol{D}^{-1}\boldsymbol{w} \,, \tag{2.159}$$

which is the output of a compressive sensing problem where the measurements are affected by white noise. This transformation changes the statistics of $\boldsymbol{A}$ because $\boldsymbol{\Lambda}^{\frac{1}{2}}$ is a diagonal real matrix with varying entries $\sqrt{\lambda_a}$. The of rows of $\boldsymbol{A}' = \boldsymbol{D}^{-1}\boldsymbol{A}$ would exhibit entries of possibly different variance $\sigma^2_{A,a,i} \neq \sigma^2_{A,b,i}$ and also

$\sigma^2_{A,a,i} \neq L^{-1}$. The solution is to use a Karhunen-Loève transformation (KLT) i.e. multiplying with the (unitary) matrix $\boldsymbol{V}^{-1} = \boldsymbol{V}^T$:

$$\boldsymbol{v} = \boldsymbol{V}^T\boldsymbol{y} = \boldsymbol{V}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{V}^T\boldsymbol{w}\,. \tag{2.160}$$

By setting $\boldsymbol{B} = \boldsymbol{V}^T\boldsymbol{A}$ and $\boldsymbol{n} = \boldsymbol{V}^T\boldsymbol{w}$ the CS problem can be written as

$$\boldsymbol{v} = \boldsymbol{B}\boldsymbol{x} + \boldsymbol{n}\,, \tag{2.161}$$

where the variance of the transformed noise $\boldsymbol{n}$ is

$$\mathbb{E}\left\{\mathbf{n}\mathbf{n}^T\right\} = \mathbb{E}\left\{\boldsymbol{V}^T\mathbf{w}\mathbf{w}^T\boldsymbol{V}\right\} = \mathbb{E}\left\{\boldsymbol{V}^T\boldsymbol{U}\mathbf{u}\mathbf{u}^T\boldsymbol{U}^T\boldsymbol{V}\right\} \tag{2.162}$$

$$= \boldsymbol{V}^T\boldsymbol{U}\underbrace{\mathbb{E}\left\{\mathbf{u}\mathbf{u}^H\right\}}_{\boldsymbol{I}}\boldsymbol{U}^T\boldsymbol{V} = \boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{V} \tag{2.163}$$

$$= \boldsymbol{\Lambda}\,. \tag{2.164}$$

Since $\boldsymbol{\Lambda}$ is diagonal, the transformed CS problem (2.161) can be solved using the method described in this section.

2.6.2 Algorithm

The algorithm for Bayesian approximate message passing with variable measurement noise is listed in Appendix A, p.115 as Algorithm 9. This algorithm assumes that the noise samples w_a are independent of each other as well as $\boldsymbol{x}$, $\boldsymbol{A}$ and (thus also) from $\boldsymbol{y}$ but allows for different variances of each w_a. The operation of the algorithm can be explained intuitively by regarding the modification of $\boldsymbol{\mu}_\mathbf{z}$ as a weighting procedure: noisier, and thus, less reliable entries of $\boldsymbol{\mu}_\mathbf{z}$ are multiplied with a smaller factor $\sigma_\mathsf{x}^{2(l)}/(\sigma^2_{\mathsf{w},a} + \sigma^2_\mathsf{z})$ than more reliable entries, for which the noise variance $\sigma^2_{\mathsf{w},a}$ is smaller.

2.6.3 State Evolution

For regular MD-BAMP, the variance $\sigma_\mathsf{x}^{2(l)}$ at the input of the denoiser is defined by (2.122), i.e. the sum of the average noise variance σ^2_w and the variance σ^2_z. In case of variable noise, the variance $\sigma_\mathsf{x}^{2(l)}$ is computed according to (2.134). The state evolution step defined by (2.122) therefore needs to be adjusted and a full state evolution iteration reads

$$\sigma^{2(l)}_{\mathsf{x},[t]} = L\left(\sum_{\mathcal{W}_k\in\mathcal{V}}\frac{|\mathcal{W}_k|}{\sigma^2_{\mathsf{w},k} + \rho^{-1}\sigma^2_{\mathsf{x},[t-1]}}\right)^{-1} \tag{2.165}$$

$$= L\left(\sum_{\mathcal{W}_k\in\mathcal{V}}\frac{|\mathcal{W}_k|}{\sigma^2_{\mathsf{w},k} + \sigma^2_{\mathsf{z},[t-1]}}\right)^{-1} \tag{2.166}$$

$$\sigma^2_{\mathsf{x},[t]} = \mathbb{E}\left\{\frac{1}{\kappa_k}\left\|\boldsymbol{F}(\boldsymbol{x}_0 + \sigma^{(l)}_{\mathsf{x},[t-1]}\mathbf{n}) - \boldsymbol{x}_0\right\|_2^2\right\}\,. \tag{2.167}$$

2.7 BAMP for Non-Zero-Mean $\boldsymbol{A}$

Motivation. The measurement matrix might be affected by a mean. In certain systems a nonzero-mean matrix could simplify hardware implementations, e.g. when it is possible to design the matrix, as is the case for compressed channel codes. A Walsh-Hadamard transform with a suitably chosen matrix mean does not need subtractions, for example. In other cases, it might not be possible to design the measurement matrix at all, because it is defined by a physical system. Sometimes, the sensing matrix can only be designed to a certain extent, as in compressed image acquisition systems using digital light-processing (DLP) based on micro-mirror devices [27]. In these systems, light from a certain region (a pixel) can either be accumulated or discarded, yielding an acquisition matrix with entries $B_{a,i} \in \{0, 1\}$.

Prior work. The problem of nonzero-mean sensing matrices has been examined in several publications. Since neither the AMP algorithm nor most of its descendants work for non-zero-mean sensing matrices $\boldsymbol{A}$, adapted algorithms have been proposed. In [60], mean removal is used in conjuction with an adaptive damping technique to stabilize the generalized approximate message passing (GAMP) algorithm [49]. While the modified GAMP algorithm is usable for more general ill-conditioned measurement matrices, it is still sensitive to the matrix mean. Furthermore, not even the Gaussian message passing algorithm converges for non-zero-mean $\boldsymbol{A}$. Reconstruction is possible using GMP with a modified message passing schedule [16]. GMP is much slower than AMP, however. Another promising algorithm is vector AMP (VAMP), which is stable for a wider variety of sensing matrices [50] but requires computation of an economy SVD of the sensing matrix. It is thus desirable to find an approximate message passing algorithm for the case of nonzero-mean sensing matrix. A subset of the novel results from this section was first presented at the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) [10]. This section extends the algorithm towards multi-dimensional priors and offers a more detailed derivation.

System analysis. One possible explanation as to why GMP does not converge is the fact that already the simplification of $m_{a\to i}(x_i)$ with a Gaussian distribution is not possible anymore. Let

$$c_j = B_{a,j}\mathsf{x}_j \,, \tag{2.168}$$

$$z_{a,i} = y_a - \sum_{j\neq i} c_j \,, \tag{2.169}$$

with $\boldsymbol{B}$ the sensing matrix. For $\mathbb{E}\{\mathsf{B}_{a,i}\} = 0$,

$$\mathbb{E}\{\mathsf{y}_a\} = 0 \tag{2.170}$$

$$\mathbb{E}\{\mathsf{B}_{a,i}x_i\} = 0 \tag{2.171}$$

$$\mathbb{E}\{\mathsf{c}_i\mathsf{c}_j\}|_{i\neq j} = 0\,, \tag{2.172}$$

even for $\mathbb{E}\{\mathsf{x}_i\} \neq 0$ and $\mathbb{E}\{\mathsf{x}_i\mathsf{x}_j\}|_{i\neq j} \neq 0$. This is used in (2.41). For $\mathbb{E}\{\mathsf{B}_{a,i}\} \neq 0$, the expressions (2.170) and (2.171) only hold if $\mathbb{E}\{\mathsf{x}_i\} = 0$ and (2.172) requires $\mathbb{E}\{\mathsf{x}_i\mathsf{x}_j\}|_{i\neq j} = 0$. These conditions are satisfied by many simple CS problems (e.g. the Bernoulli-Gaussian case). Any systematical estimation error of $\boldsymbol{x}$ is amplified by the matrix mean, however. In the following paragraphs, the effect of the matrix mean μ_B is examined in detail.
Consider the linear compressed sensing system

$$\boldsymbol{v} = \boldsymbol{B}\boldsymbol{x} + \boldsymbol{w}\,, \tag{2.173}$$

where $\boldsymbol{B} \in \mathbb{R}^{L\times N}$ with $L < N$. Again, $\boldsymbol{x}$ is an unknown vector and $\boldsymbol{w}$ measurement noise. Furthermore, the noise is assumed to be i.i.d. Gaussian with zero mean and independent of both B and x. In this section, it is assumed that the matrix $\boldsymbol{B}$ consists of entries that are pairwise independent and obey

$$\mathbb{E}\{\mathsf{B}_{a,i}\} = \mu_B \tag{2.174}$$

$$\mathbb{E}\left\{(\mathsf{B}_{a,i} - \mathbb{E}\{\mathsf{B}_{a,i}\})^2\right\} = L^{-1}\,. \tag{2.175}$$

Equivalently, the matrix $\boldsymbol{B}$ can be written as

$$\boldsymbol{B} = \mu_B\mathbf{1}\mathbf{1}^T + \boldsymbol{A}\,, \tag{2.176}$$

with $\mathbb{E}\{\mathsf{A}_{a,i}\} = 0$ and $\mathbb{E}\left\{(\mathsf{A}_{a,i})^2\right\} = L^{-1}$. The mean μ_B can be estimated with sufficient precision and does not need to be known beforehand. Taking into account this decomposition, (2.173) can be written as

$$\boldsymbol{v} = (\mu_B\mathbf{1}\mathbf{1}^T + \boldsymbol{A})\boldsymbol{x} + \boldsymbol{w} \tag{2.177}$$

$$= \mu_B\mathbf{1}\mathbf{1}^T\boldsymbol{x} + \underbrace{\boldsymbol{A}\boldsymbol{x} + \boldsymbol{w}}_{\boldsymbol{y}} \tag{2.178}$$

$$= \mu_B\mathbf{1}\mathbf{1}^T\boldsymbol{x} + \boldsymbol{y} \tag{2.179}$$

$$v_a = \sum_i \mu_B x_i + y_a = N\mu_B\bar{x} + y_a\,. \tag{2.180}$$

In (2.180), the value $\bar{x}$ is defined as

$$\bar{x} = \frac{1}{N}\sum_i x_i\,, \tag{2.181}$$

i.e. the arithmetic mean of $\boldsymbol{x}$'s entries. Note that even if $\mathbb{E}_{\mathsf{x}}\{\bar{\mathsf{x}}\} = 0$, its variance is

$$\sigma^2_{\bar{\mathsf{x}}} = \mathbb{E}_{\mathsf{x}}\left\{\bar{\mathsf{x}}^2\right\} = \frac{1}{N^2}\sum_i\sum_j \mathbb{E}_{\mathsf{x}}\{\mathsf{x}_i\mathsf{x}_j\}\,, \tag{2.182}$$

which is finite for finite N. Thus the expression $N\mu_B\bar{x}$ causes v_a to be different from y_a in general. The offset depends on the problem dimension N, the matrix mean μ_B and the arithmetic mean of a particular sample vector $\boldsymbol{x}$. Expectation and variance of y_a are

$$\mu_{\mathsf{y}_a} = \mathbb{E}\{\mathsf{y}_a\} = \mathbb{E}\left\{\sum_i \mathsf{A}_{a,i}\mathsf{x}_i + \mathsf{w}_a\right\} \tag{2.183}$$

$$= \sum_i \mathbb{E}\{\mathsf{A}_{a,i}\}\,\mathbb{E}\{\mathsf{x}_i\} + \mathbb{E}\{\mathsf{w}_a\} = 0 \tag{2.184}$$

$$\mathbb{E}\{\mathsf{y}_a\mathsf{y}_b\} = \mathbb{E}\left\{\Big(\sum_i \mathsf{A}_{a,i}\mathsf{x}_i + \mathsf{w}_a\Big)\Big(\sum_j \mathsf{A}_{b,j}\mathsf{x}_j + \mathsf{w}_b\Big)\right\} \tag{2.185}$$

$$= \mathbb{E}\left\{\Big(\sum_i\sum_j \mathsf{A}_{a,i}\mathsf{x}_i\mathsf{A}_{b,j}\mathsf{x}_j\Big)\right\} + \mathbb{E}\{\mathsf{w}_a\mathsf{w}_b\} \tag{2.186}$$

$$= I(a-b)\left(\frac{1}{L}\sum_i \mathbb{E}\{\mathsf{x}_i^2\} + \mathbb{E}\{\mathsf{w}_a^2\}\right) \tag{2.187}$$

$$\sigma^2_{\mathsf{y},a} = \frac{1}{L}\sum_i P_{\mathsf{x},i} + \sigma^2_{\mathsf{w}}\,, \tag{2.188}$$

with $I(x) = 1 \iff x = 0$ and $I(x) = 0$ otherwise. In order to compare the magnitude of $\sigma^2_{\bar{\mathsf{x}}}$ and $\sigma^2_{\mathsf{y}_a}$, assume for the moment that $\mathbb{E}\{\mathsf{x}_i\mathsf{x}_j\}\,|_{i\neq j} = 0$. Then,

$$\sigma^2_{\mathsf{y}_a} = \frac{1}{L}\sum_i P_{\mathsf{x}_i} + \sigma^2_{\mathsf{w}} \tag{2.189}$$

$$\sigma^2_{\bar{\mathsf{x}}} = \frac{1}{N^2}\sum_i P_{\mathsf{x}_i}\,. \tag{2.190}$$

Keeping in mind that $\bar{x}$ is scaled with the factor $N\mu_B$, it becomes obvious that offset of $\boldsymbol{v}$ from $\boldsymbol{y}$ can be comparable in magnitude to the standard deviation of $\boldsymbol{y}$ for a particular instance of a CS problem. Therefore, assuming that $\boldsymbol{v} = \boldsymbol{y}$ and running AMP with $\boldsymbol{A}$ instead of $\boldsymbol{B}$ does not work well: it increases the effective noise variance such that

$$\sigma^2_{\mathsf{w}'} = \sigma^2_{\mathsf{w}} + N^2\mu_B^2\sigma^2_{\bar{\mathsf{x}}} \tag{2.191}$$

$$= \sigma^2_{\mathsf{w}} + \mu_B^2\sum_i P_{\mathsf{x}_i}\,. \tag{2.192}$$

It is possible to estimate $\boldsymbol{y}$ by removal of the the arithmetic mean of $\boldsymbol{v}$, which is

$$\bar{v} = \frac{1}{L}\sum_a v_a = \frac{1}{L}\sum_a (N\mu_B\bar{x} + y_a) \tag{2.193}$$

$$= N\mu_B\bar{x} + \bar{y}\,. \tag{2.194}$$

Removing the arithmetic mean of $\boldsymbol{v}$ thus results in removal of both the (undesired) effect of the matrix mean and the arithmetic mean of $\boldsymbol{y}$. Since the latter should be retained, mean removal is a trade-off. The variance of $\bar{y}$ is

$$\mathbb{E}\left\{\bar{\mathsf{y}}\right\} = \mathbb{E}\left\{\frac{1}{L}\sum_a \mathsf{y}_a\right\} = 0 \tag{2.195}$$

$$\mathbb{E}\left\{\bar{\mathsf{y}}^2\right\} = \mathbb{E}\left\{\frac{1}{L^2}\sum_a\sum_b \mathsf{y}_a\mathsf{y}_b\right\} \tag{2.196}$$

$$= \frac{1}{L^2}\sum_a\left(\frac{1}{L}\sum_i P_{\mathsf{x},i} + \sigma_{\mathsf{w}}^2\right) = \frac{1}{L}\sigma_{\mathsf{y},a}^2\,, \tag{2.197}$$

while the moments of $\bar{x}$ are

$$\mathbb{E}\left\{\bar{\mathsf{x}}\right\} = \mathbb{E}\left\{\frac{1}{N}\sum \mathsf{x}_i\right\} = \frac{1}{N}\sum_i \mu_{\mathsf{x},i} \tag{2.198}$$

$$\mathbb{E}\left\{\bar{\mathsf{x}}^2\right\} = \mathbb{E}\left\{\frac{1}{N^2}\sum_i\sum_j \mathsf{x}_i\mathsf{x}_j\right\} = \frac{1}{N^2}\sum_{i,j}(\boldsymbol{R}_{\mathsf{x}})_{i,j}\,. \tag{2.199}$$

For i.i.d., zero mean x_i, the undesired term $N\mu_B\bar{x}$ therefore has a variance of $\mu_B^2 N\sigma_{\mathsf{x}}^2$ while $\bar{y}$ has a variance of $\sigma_{\mathsf{x}}^2 N/L^2$ in the noiseless case. For significant μ_B, removing the mean of $\boldsymbol{v}$ is therefore desirable.
In the noiseless case, the arithmetic mean of $\boldsymbol{y}$ is zero only if the columns of $\boldsymbol{A}$ have an arithmetic mean of zero. This can be used when the problem permits a custom design of the sensing matrix: choosing the matrix such that its columns have an arithmetic mean of zero makes it possible to compute $\boldsymbol{y}$ from $\boldsymbol{v}$ in the noiseless case. The same can be achieved in the presence of noise if the CS problem permits measurement of the arithmetic mean of $\boldsymbol{x}$. In all other cases, subtracting the arithmetic mean from $\boldsymbol{v}$ to obtain $\boldsymbol{y}$ results in an estimation error which is moreover correlated with $\boldsymbol{x}$. The error ε can be written as

$$\varepsilon = y_a - \hat{y}_a = y_a - (v_a - \bar{v}) = -\mu_B\sum_i x_i + \bar{v} \tag{2.200}$$

$$= \frac{1}{L}\left(\sum_i x_i\sum_a A_{a,i} + \sum_a w_a\right) = \bar{y}\,. \tag{2.201}$$

The variance of the estimation error is thus given by (2.197) and the effective signal-to-noise ratio (SNR) is thus[1]

$$\mathrm{SNR}_{\mathrm{dB}}^{\mathrm{eff}} = 10\log_{10}\left(\frac{\mathbb{E}\left\{\mathsf{y}_a^2\right\}}{\mathbb{E}\left\{\varepsilon^2\right\}}\right) = 10\log_{10}\left(\frac{\sigma_{\mathsf{y},a}^2}{\sigma_\varepsilon^2}\right) \tag{2.202}$$

[1] The characterization of the estimation error ε as noise is highly inaccurate – all of its samples are identical and depend on x, A and w. Modeling it as noise is merely useful to obtain a rough estimate of the quality of results that can be expected when using mean removal.

$$= 10 \log_{10} \left(\frac{\sigma_{\mathrm{y},a}^2}{L^{-1} \sigma_{\mathrm{y},a}^2} \right) = 10 \log_{10}(L) \,. \tag{2.203}$$

It is therefore possible to apply MD-BAMP to a modified noiseless CS problem where the mean of $\boldsymbol{v}$ and $\boldsymbol{B}$ were removed by setting the noise variance to $L^{-2} \|\boldsymbol{y}\|_2^2$. It shall be shown below that this stabilizes MD-BAMP for the modified problem and that MD-BAMP follows the convergence predicted by regular state evolution with the noise variance set accordingly. Achievable signal-to-distortion ratio (SDR) of the recovered signal is limited, especially for low-dimensional problems ($N \approx 10^3$) and low-rate CS ($\delta = 10^{-3}, N = 10^5 \implies \mathrm{SNR}_{\mathrm{dB}}^{\mathrm{eff}} = 20\mathrm{dB}$).

Since the BAMP derivation fails at an early stage, the case of non-zero-mean sensing matrix $\boldsymbol{A}$ shall be incorporated as modification of the original algorithm. This has multiple advantages: on the one hand the resulting modification can be applied to existing, BAMP-based algorithms and on the other hand, a number of theoretical results regarding the performance of BAMP under various conditions can be reused with slight modifications. The disadvantage is that any optimizations that are possible due to the non-zero mean of the matrix are lost.

In the following derivation, the dependence of the error term ε on $\boldsymbol{x}$ shall be exploited to yield optimal recovery in the AMP regime. For the remaining analysis, assume that

$$\bar{x} = \frac{1}{N} \sum_i x_i \neq 0 \tag{2.204}$$

$$r_i = \sum_a A_{a,i} \neq 0 \,, \tag{2.205}$$

as well as unknown $\bar{x}$. The arithmetic mean of $\boldsymbol{x}$ effects the original BAMP algorithm as per (2.179). This modification is depicted in a graphical model shown in Fig. 2.2. Note that the setup is similar to GAMP with the crucial difference that the factors $f_{vy,a}$ depend on x via $\bar{\mathsf{x}}$.

Factor $f_{vy,a}$. The factors $f_{vy,a}$ represent the dependency between v_a, y_a and $\bar{\mathsf{x}}$ (and optionally the noise w_a). In the noiseless case, it is

$$f_{vy,a}(\bar{x}, y_a) = \delta(v_a - N \mu_B \bar{x} - y_a) \,. \tag{2.206}$$

For Gaussian noise with parameters $\mu_{\mathsf{w},a} = 0$ and $\sigma_{\mathsf{w},a}^2 > 0$ the factor reads

$$f_{vy,a}(\bar{x}, y_a) \propto \exp\left(-\frac{1}{2\sigma_{\mathsf{w},a}^2} (v_a - N \mu_B \bar{x} - y_a)^2 \right) . \tag{2.207}$$

Factor $f_{\bar{\mathsf{x}}}$. This factor describes the relationship between $\bar{\mathsf{x}}$ and x. Here, $\bar{\mathsf{x}}$ is an auxiliary variable. It is possible to get rid of it by introducing dependencies between the factors $f_{vy,a}$ and x. Note that this results in adding $O(NL)$ edges

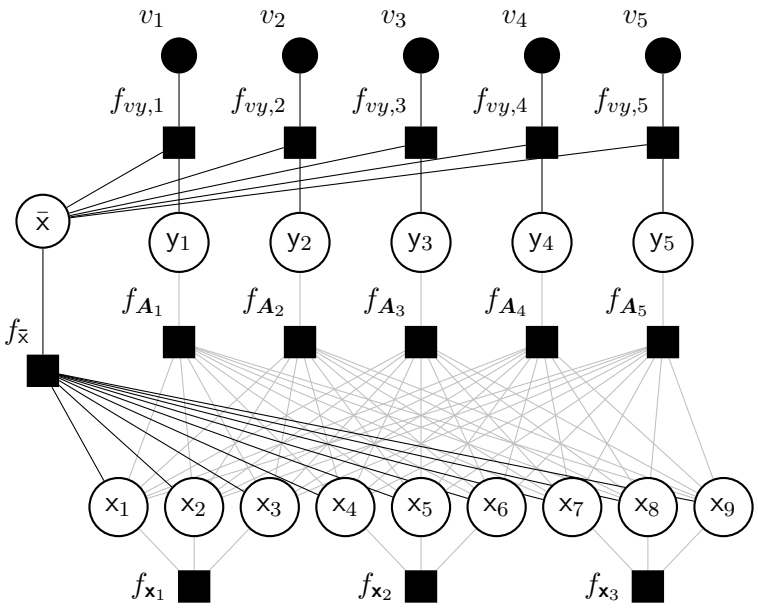


Figure 2.2: Example of a graphical model describing a CS problem with a nonzero-mean sensing matrix $\boldsymbol{B}$. New edges have been emphasized. There are $K = 3$ factors of $f_{\mathbf{x}}(\boldsymbol{x})$, $N = 9$ unknown variables x_i and $L = 5$ known variables v_a.

while with auxiliary variable only $O(N + L)$ edges are necessary. The factor $f_{\bar{\mathsf{x}}}$ is given as

$$f_{\bar{\mathsf{x}}}(\bar{x}, \boldsymbol{x}) = \delta\left(\bar{x} - \frac{1}{N}\sum_i x_i\right). \tag{2.208}$$

Messages. The message passing rules of the sum-product algorithm can be used to derive expressions of messages along the newly introduced edges. There are eight new messages:

- $m_{vy\to\mathsf{y},a}(y_a)$, from factor node $f_{vy,a}$ to y_a,
- $m_{\mathsf{y},a\to vy}(y_a)$, from variable node y_a to $f_{vy,a}$,
- $m_{vy\to\bar{\mathsf{x}}}(\bar{x})$, from factor node $f_{vy,a}$ to $\bar{\mathsf{x}}$,
- $m_{\bar{\mathsf{x}}\to vy}(\bar{x})$, from variable node $\bar{\mathsf{x}}$ to $f_{vy,a}$,
- $m_{\bar{\mathsf{x}}\to f,\bar{x}}(\bar{x})$, from variable node $\bar{\mathsf{x}}$ to $f_{\bar{\mathsf{x}}}$,
- $m_{f,\bar{x}\to\bar{\mathsf{x}}}(\bar{x})$, from factor node $f_{\bar{\mathsf{x}}}$ to $\bar{\mathsf{x}}$,

- $m_{f,\bar{x}\to\mathsf{x},i}(x_i)$, from factor node $f_{\bar{\mathsf{x}}}$ to x_i,
- $m_{\mathsf{x},i\to f,\bar{x}}(x_i)$, from variable node x_i to $f_{\bar{\mathsf{x}}}$.

Following the message passing rules (cf. (2.28), (2.29)), one obtains

$$m_{vy\to\mathsf{y},a}(y_a) = \int_{\bar{\mathsf{x}}} f_{vy,a}(\bar{x},y_a)m_{\bar{x}\to vy}(\bar{x})\mathrm{d}\bar{x} \tag{2.209}$$

$$m_{\mathsf{y},a\to vy}(y_a) = m_{f,a\to\mathsf{y},a}(y_a) \tag{2.210}$$

$$m_{vy\to\bar{\mathsf{x}}}(\bar{x}) = \int_{\bar{\mathsf{x}}} f_{vy,a}(\bar{x},y_a)m_{\mathsf{y},a\to vy}(y_a)\mathrm{d}y_a \tag{2.211}$$

$$m_{\bar{\mathsf{x}}\to vy,a}(\bar{x}) = m_{f,\bar{x}\to\bar{x}}(\bar{x})\prod_{b\neq a} m_{vy,b\to\bar{\mathsf{x}}}(\bar{x}) \tag{2.212}$$

$$m_{\bar{\mathsf{x}}\to f,\bar{x}}(\bar{x}) = \prod_{a} m_{vy,a\to\bar{\mathsf{x}}}(\bar{x}) \tag{2.213}$$

$$m_{f,\bar{x}\to\bar{\mathsf{x}}}(\bar{x}) = \int_{\mathbf{x}} f_{\bar{\mathsf{x}}}(\bar{x},\boldsymbol{x})\prod_{i} m_{\mathsf{x},i\to f,\bar{x}}(x_i)\mathrm{d}\boldsymbol{x} \tag{2.214}$$

$$m_{f,\bar{x}\to\mathsf{x},i}(x_i) = \int_{j\neq i}\int_{\bar{\mathsf{x}}} f_{\bar{\mathsf{x}}}(\bar{x},\boldsymbol{x})m_{\bar{\mathsf{x}}\to f,\bar{x}}(\bar{x})\mathrm{d}\bar{x}\prod_{j\neq i} m_{\mathsf{x},i\to f,\bar{x}}(x_i)\mathrm{d}x_j \tag{2.215}$$

$$m_{\mathsf{x},i\to f,\bar{x}}(x_i) = m_{f,\mathsf{x},i\to\mathsf{x}_i}(x_i)\prod_{a} m_{f,a\to\mathsf{x},i}(x_i)\,. \tag{2.216}$$

Not all messages are required to achieve convergence. Fig. 2.2 shows that it is possible to interpret the factor $f_{\bar{\mathsf{x}}}$ and variable $\bar{\mathsf{x}}$ as a new row in the sensing matrix. It does not correspond to any measured value v_a, however, contrary to the factors $f_{\boldsymbol{A},a}$ and variables y_a. It is possible to omit the updates of x_i originating from $f_{\bar{\mathsf{x}}}$. This is equivalent to removing one row from the sensing matrix. For sensing matrices with a large number of rows, this has negligible effect.
The algorithm can be initialized by starting with the last message, $m_{\mathsf{x},i\to f,\bar{x}}(x_i)$. During iterations, this is the tuple $(\mu_{\mathsf{x},i},\sigma^2_{\mathsf{x},i})$ with $\mu_{\mathsf{x},i} = F(\dots)$ and $\sigma^2_{\mathsf{x},i} = G(\dots)$. For initialization, let

$$\mu_{\mathsf{x},i} = \int_i x_i \int_{\substack{j\neq i\\ j\in\mathcal{K}}} f_{\mathbf{x},k}(\boldsymbol{x}_k)\mathrm{d}x_j\mathrm{d}x_i \tag{2.217}$$

$$\sigma^2_{\mathsf{x},i} = \int_i x_i^2 \int_{\substack{j\neq i\\ j\in\mathcal{K}}} f_{\mathbf{x},k}(\boldsymbol{x}_k)\mathrm{d}x_j\mathrm{d}x_i - \mu^2_{\mathsf{x},i}\,, \tag{2.218}$$

i.e. the prior mean and variance of x_i. Subsequently, $m_{f,\bar{x}\to\bar{\mathsf{x}}}$ can be parameterized using mean and variance with

$$\mu_{f,\bar{\mathsf{x}}\to\bar{\mathsf{x}}} = \frac{1}{N}\sum \mu_{\mathsf{x},i} \tag{2.219}$$

$$\sigma^2_{f,\bar{\mathsf{x}}\to\bar{\mathsf{x}}} = \frac{1}{N^2}\sum \sigma^2_{\mathsf{x},i}\,. \tag{2.220}$$

During initialization, $m_{vy\to\bar{\mathsf{x}}}$ is unknown and thus, $m_{\bar{\mathsf{x}}\to vy,a}(\bar{x})$ consists of the parameters $\mu_{f,\bar{\mathsf{x}}\to\bar{\mathsf{x}}}$ and $\sigma^2_{f,\bar{\mathsf{x}}\to\bar{\mathsf{x}}}$. Once $m_{vy,a\to\bar{\mathsf{x}}}(\bar{x})$ is known, they can be mixed in according to (2.212). To maintain parameterization, the message from $f_{vy,a}$ to $\bar{\mathsf{x}}$ can be assumed to approximate a Gaussian distribution.
Now that $m_{\bar{\mathsf{x}}\to vy,a}(\bar{x})$ is known, the message from $f_{vy,a}$ to y_a can be computed (cf. (2.209)). The mean and variance of y_a compute as (cf. (2.207))

$$\mu_{vy,a\to\mathsf{y},a} = v_a - N\mu_B\mu_{\bar{\mathsf{x}}\to vy,a} \tag{2.221}$$
$$\sigma^2_{vy,a\to\mathsf{y},a} = \sigma^2_{\mathsf{w},a} + N^2\mu_B^2\sigma^2_{\bar{\mathsf{x}}\to vy,a}\,. \tag{2.222}$$

This can be shown by expressing (2.207) in terms of $\bar{\mathsf{x}}$:

$$f_{vy,a}(\bar{x}, y_a) = \mathcal{N}\left(\frac{v_a - y_a}{N\mu_B}, \frac{\sigma^2_{\mathsf{w},a}}{(N\mu_B)^2}\right)\,, \tag{2.223}$$

computing the product with the Gaussian message given by the parameters $\mu_{\bar{\mathsf{x}}\to vy,a}$ and $\sigma^2_{\bar{\mathsf{x}}\to vy,a}$ and finally evaluating the Gaussian integral (2.209) w.r.t. $\bar{\mathsf{x}}$. This results in

$$m_{vy\to\mathsf{y},a}(y_a) \propto \exp\left(\left(2\left(\frac{\sigma^2_{\mathsf{w},a}}{(N\mu_B)^2} + \sigma^2_{\bar{\mathsf{x}}\to vy,a}\right)\right)^{-1}\left(\frac{v_a - y_a}{N\mu_B} - \mu_{\bar{\mathsf{x}}\to vy,a}\right)^2\right) \tag{2.224}$$
$$\propto \exp\left(\frac{(v_a - y_a - \mu_{\bar{\mathsf{x}}\to vy,a})^2}{2\left(\sigma^2_{\mathsf{w},a} + (N\mu_B)^2\sigma^2_{\bar{\mathsf{x}}\to vy,a}\right)}\right)\,. \tag{2.225}$$

The mean and variance w.r.t. y_a can then be identified as (2.221) and (2.222) respectively. Subsequently, one iteration of AMP can be computed with $\mu_{vy,a\to y,a}$ substituted for y_a and $\sigma^2_{vy,a\to y,a}$ substituted for $\sigma^2_{w,a}$. The initial value of $\sigma^2_{\mathsf{z},a}$ can be computed as the empirical variance of $\mu_{vy,a\to y,a}$. After this iteration, the message $(\mu_{y,a\to vy,a}, \sigma^2_{y,a\to vy,a})$ is

$$\mu_{y,a\to vy,a} = \sum_i A_{a,i}\mu_{i\to a} \approx \sum_i A_{a,i}\mu_{\mathsf{x},i} - \frac{\mu_{\mathsf{z},a}}{L}\sum_j \frac{\partial F}{\partial \mu^{(l)}} \tag{2.226}$$
$$\sigma^2_{y,a\to vy,a} = \sum_i A^2_{a,i}\sigma^2_{i\to a} \approx \frac{N}{L}\sigma^2_{\mathsf{x}} = \sigma^2_{y\to vy}\,. \tag{2.227}$$

Given v_a and the message from y_a to $f_{vy,a}$, the message from $f_{vy,a}$ to $\bar{x}$ can be computed. Since $N\mu_B\bar{x} = v_a - y_a$, the message is

$$N\mu_B\mu_{vy,a\to\bar{x}} = v_a - \mu_{y,a\to vy,a} = v_a - \sum_i A_{a,i}\mu_{i\to a} \tag{2.228}$$
$$(N\mu_B)^2\sigma^2_{vy,a\to\bar{x}} = \sigma^2_{y,a\to vy,a} + \sigma^2_{\mathsf{w},a}\,. \tag{2.229}$$

At $\bar{x}$ there are now $L+1$ incoming messages, namely L from the nodes $f_{vy,a}$ and one from $f_{\bar{x}}$. Formally and following (2.212), the message from $\bar{x}$ to the factor nodes $f_{vy,a}$ is

$$\sigma^2_{\bar{x}\to vy,a} = \left(\frac{1}{\sigma^2_{f,\bar{x}\to\bar{x}}} + \sum_{b\neq a} \frac{1}{\sigma^2_{vy,b\to\bar{x}}} \right)^{-1} \tag{2.230}$$

$$\mu_{\bar{x}\to vy,a} = \sigma^2_{\bar{x}\to vy,a} \left(\frac{\mu_{f,\bar{x}\to\bar{x}}}{\sigma^2_{f,\bar{x}\to\bar{x}}} + \sum_{b\neq a} \frac{\mu_{vy,b\to\bar{x}}}{\sigma^2_{vy,b\to\bar{x}}} \right). \tag{2.231}$$

To simplify the derivation, assume from now on that the noise variances are identical, i.e. $\sigma^2_{\mathsf{w},a} = \sigma^2_{\mathsf{w}}$. Since the approximation of $\sigma^2_{y,a\to vy,a}$ in (2.227) does not depend on a either, the variances $\sigma^2_{vy,a\to\bar{\mathsf{x}}}$ and $\sigma^2_{\bar{\mathsf{x}}\to vy,a}$ can also be regarded as independent of a. The notation is simplified by dropping the factor node index where it is superfluous. It is thus possible to rewrite (2.229) and (2.230)

$$\sigma^2_{vy\to\bar{\mathsf{x}}} = \frac{1}{(N\mu_B)^2} \left(\frac{N}{L}\sigma^2_{\mathsf{x}} + \sigma^2_{\mathsf{w}} \right) \tag{2.232}$$

$$\sigma^2_{\bar{\mathsf{x}}\to vy} = \left(\frac{1}{\sigma^2_{f,\bar{x}\to\bar{\mathsf{x}}}} + \frac{L-1}{\sigma^2_{vy\to\bar{\mathsf{x}}}} \right)^{-1} \tag{2.233}$$

$$= \left(\frac{N^2}{\sum_i \sigma^2_{\mathsf{x},i}} + \frac{(L-1)(N\mu_B)^2}{L^{-1}\sum_i \sigma^2_{\mathsf{x},i} + \sigma^2_{\mathsf{w}}} \right)^{-1}. \tag{2.234}$$

Furthermore, (2.231) can be written as

$$\mu_{\bar{\mathsf{x}}\to vy,a} = \sigma^2_{\bar{\mathsf{x}}\to vy} \left(\frac{\mu_{f,\bar{\mathsf{x}}\to\bar{\mathsf{x}}}}{\sigma^2_{f,\bar{\mathsf{x}}\to\bar{\mathsf{x}}}} + \frac{L-1}{L\sigma^2_{vy\to\bar{\mathsf{x}}}} \sum_b \mu_{vy,b\to\bar{\mathsf{x}}} \right) \tag{2.235}$$

$$\approx \sigma^2_{\bar{\mathsf{x}}\to vy} \left(\frac{\mu_{f,\bar{\mathsf{x}}\to\bar{\mathsf{x}}}}{\sigma^2_{f,\bar{\mathsf{x}}\to\bar{\mathsf{x}}}} + \frac{1}{\sigma^2_{vy\to\bar{\mathsf{x}}}} \sum_b \mu_{vy,b\to\bar{\mathsf{x}}} \right). \tag{2.236}$$

2.7.1 Algorithm

The resulting approximate message passing algorithm for nonzero-mean sensing matrix (MEAN-BAMP) is displayed in Appendix A, p.116 as Algorithm 10. While it does look more complicated than regular MD-BAMP, its computational complexity is similar: the additional lines (10 - 17) are operating on scalars. While line 15 contains a matrix-vector multiplication, the term can be reused in line 18. The complexity of the remaining additional expressions grows with N. The number of additional floating point divisions is constant. Thus, both algorithms have a complexity of $O(NL)$ due to matrix-vector multiplications.

2.7.2 State Evolution

It is possible to formulate a state evolution recursion by taking into account modifications to $\sigma_{\mathsf{x}}^{2(l)}$. State evolution for regular BAMP is defined by the recursion (2.121), (2.122). MEAN-BAMP modifies the second line of the state evolution since it introduces an effective noise variance $\sigma_{vy\to y}^2$. Its value is lower bounded by σ_{w}^2 and equal to the value predicted by state evolution for BAMP if $\mu_B = 0$, i.e. a zero-mean sensing matrix. Thus, for a zero-mean sensing matrix $\boldsymbol{A}$, state evolution for MEAN-BAMP reduces to state evolution for standard BAMP. The recursion can be written as

$$\sigma_{\mathsf{x},[t]}^2 = \mathbb{E}\left\{\frac{1}{\kappa_k}\left\|\boldsymbol{F}\left(\boldsymbol{x}_0 + \sigma_{\mathsf{x},[t-1]}^{(l)}\mathbf{n}\right) - \boldsymbol{x}_0\right\|_2^2\right\} \tag{2.237}$$

$$\sigma_{\tau,[t]}^2 = \delta^{-1}\sigma_{\mathsf{x},[t]}^2 + \sigma_{\mathsf{w}}^2 \tag{2.238}$$

$$\sigma_{\kappa,[t]}^2 = \left(\frac{1}{N\mu_B^2\sigma_{\mathsf{x},[t]}^2} + \frac{L-1}{\sigma_{\tau,[t]}^2}\right)^{-1} \tag{2.239}$$

$$\sigma_{\mathsf{x},[t]}^{2(l)} = \sigma_{\tau,[t]}^2 + \sigma_{\kappa,[t]}^2\,. \tag{2.240}$$

Chapter 3

Denoiser Construction

All variants of the Bayesian AMP algorithm use the functions F, G and the derivative of F (cf. Algorithm 5 to 10 in Appendix A). These functions are defined in (2.61), (2.62) and (2.73), (2.74) and repeated here for convenience:

$$Z = \int \cdots \int f_{\mathbf{x}}(\boldsymbol{x}) \mathcal{N}_{\mathbf{x}}(\boldsymbol{x}; \boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \mathrm{d}\boldsymbol{x} \tag{3.1}$$

$$\mu_{\mathrm{x},i} = F_i = \frac{1}{Z} \int \cdots \int x_i f_{\mathbf{x}}(\boldsymbol{x}) \mathcal{N}_{\mathbf{x}}(\boldsymbol{x}; \boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \mathrm{d}\boldsymbol{x} \tag{3.2}$$

$$\sigma^2_{\mathrm{x},i} = G_i = \frac{1}{Z} \int \cdots \int x_i^2 f_{\mathbf{x}}(\boldsymbol{x}) \mathcal{N}_{\mathbf{x}}(\boldsymbol{x}; \boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \mathrm{d}\boldsymbol{x} - \mu^2_{\mathrm{x},i \to a} \,. \tag{3.3}$$

The arguments of these functions are $(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)})$ which is the local belief regarding $\boldsymbol{x}$, computed from incoming messages. The function $f_{\mathbf{x}}(\boldsymbol{x})$ is the prior of the random vector $\mathbf{x}$. The notation is modified for this chapter. The index k is not anymore used for identification of the group of variables connected to a prior. The derivative of F first appears in (2.95). In this chapter, the functions F_i, G_i, F_i' and Z are expressed using the following definition:

$$I_{k,i}(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) = \int \cdots \int x_i^k f_{\mathbf{x}}(\boldsymbol{x}) \mathcal{N}_{\mathbf{x}}(\boldsymbol{x}; \boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \mathrm{d}\boldsymbol{x} \,. \tag{3.4}$$

The expressions (3.1) - (3.3) can then be identified as

$$Z(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) = I_0(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \tag{3.5}$$

$$F_i(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) = \frac{I_{1,i}(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)})}{I_0(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)})} \tag{3.6}$$

$$G_i(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) = \frac{I_{2,i}(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)})}{I_0(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)})} - F_i^2(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \,. \tag{3.7}$$

Note that $I_{k,i}|_{k=0}$ does not depend on i and thus the index is omitted in (3.5). The functions (3.4), (3.6) and (3.7) can be expressed in a vector valued notation,

$$\boldsymbol{I}_k(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) = \int \cdots \int \boldsymbol{x}^k f_{\mathbf{x}}(\boldsymbol{x}) \mathcal{N}_{\mathbf{x}}(\boldsymbol{x}; \boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \mathrm{d}\boldsymbol{x} \tag{3.8}$$

$$\boldsymbol{F}(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) = \frac{\boldsymbol{I}_1(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)})}{I_0(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)})} \tag{3.9}$$

$$\boldsymbol{G}(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) = \frac{\boldsymbol{I}_2(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)})}{I_0(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)})} - \boldsymbol{F}^2(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \,, \tag{3.10}$$

where the exponentiations $\boldsymbol{x}^k$ and $\boldsymbol{F}^2$ are computed component wise. Furthermore, all functions take the arguments $(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)})$ which shall henceforth be omitted, yielding a cleaner notation:

$$Z = I_0 \tag{3.11}$$

$$F_i = \frac{I_{1,i}}{I_0} \qquad \boldsymbol{F} = \frac{\boldsymbol{I}_1}{I_0} \tag{3.12}$$

$$G_i = \frac{I_{2,i}}{I_0} - F_i^2 \qquad \boldsymbol{G} = \frac{\boldsymbol{I}_2}{I_0} - \boldsymbol{F}^2 \,. \tag{3.13}$$

The derivative of F_i is computed w.r.t. an entry of the first argument $\boldsymbol{\mu}^{(l)}$. In most variants of BAMP presented in this thesis, only the diagonal of the Jacobian is needed. For sake of completeness, expressions for off-diagonal elements are presented as well. The derivative of F_i w.r.t. $\mu_j^{(l)}$ is

$$\frac{\partial F_i}{\partial \mu_j^{(l)}} = \frac{1}{I_0} \frac{\partial I_{1,i}}{\partial \mu_{\mathsf{x}_j}^{(l)}} - \frac{I_{1,i}}{I_0^2} \frac{\partial I_0}{\partial \mu_{\mathsf{x}_j}^{(l)}} \,. \tag{3.14}$$

The derivative of $I_{k,i}$ can be written as

$$\frac{\partial I_{k,i}}{\partial \mu_j^{(l)}} = \frac{1}{\sigma_j^{2(l)}} \int \cdots \int x_i^k (x_j - \mu_j^{(l)}) f_{\mathbf{x}}(\boldsymbol{x}) \mathcal{N}_{\mathbf{x}}(\boldsymbol{x}; \boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \mathrm{d}\boldsymbol{x} \tag{3.15}$$

$$= \begin{cases} \frac{1}{\sigma_j^{2(l)}} \left(R_{k,1,i,j} - \mu_j^{(l)} I_{k,i} \right) & \text{if } i \neq j \\ \frac{1}{\sigma_j^{2(l)}} \left(I_{k+1,i} - \mu_j^{(l)} I_{k,i} \right) & \text{if } i = j \,. \end{cases} \tag{3.16}$$

The term $R_{k,l,i,j}$ represents the (un-normalized) mixed moment

$$R_{k,l,i,j} = \int \cdots \int x_i^k x_j^l f_{\mathbf{x}}(\boldsymbol{x}) \mathcal{N}_{\mathbf{x}}(\boldsymbol{x}; \boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \mathrm{d}\boldsymbol{x} \,. \tag{3.17}$$

For the diagonal of $\boldsymbol{F}$'s Jacobian, this results in

$$\frac{\partial F_i}{\partial \mu_i^{(l)}} = \frac{1}{I_0 \sigma_i^{2(l)}} \left(I_{2,i} - \mu_i^{(l)} I_{1,i} \right) - \frac{I_{1,i}}{I_0^2 \sigma_i^{2(l)}} \left(I_{1,i} - \mu_i^{(l)} I_0 \right) \tag{3.18}$$

$$= \frac{1}{\sigma_i^{2(l)}} \left(\frac{I_{2,i}}{I_0} - \left(\frac{I_{1,i}}{I_0} \right)^2 \right) \tag{3.19}$$

$$= \frac{G_i}{\sigma_i^{2(l)}} \,. \tag{3.20}$$

Thus, the diagonal of the Jacobian is proportional to the variance estimation provided by G_i. An extension towards the full Jacobian is possible if $\boldsymbol{G}$ is defined to compute the complete covariance of the estimation. Let

$$\mathfrak{G} = \frac{1}{Z} \int \cdots \int \boldsymbol{x}\boldsymbol{x}^T f_{\mathbf{x}}(\boldsymbol{x}) \mathcal{N}_{\mathbf{x}}(\boldsymbol{x}; \boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \mathrm{d}\boldsymbol{x} - \boldsymbol{F}\boldsymbol{F}^T \,. \tag{3.21}$$

Here, $\mathfrak{G}$ returns a matrix, while evaluation of $\boldsymbol{G}$ results in a vector, namely the diagonal of $\mathfrak{G}$. The Jacobian of $\boldsymbol{I}_k$ w.r.t. $\boldsymbol{\mu}^{(l)}$ can be written as

$$\frac{\partial \boldsymbol{I}_k}{\partial \boldsymbol{\mu}^{(l)}} = \int \cdots \int \boldsymbol{x}^k f_{\mathbf{x}}(\boldsymbol{x}) (\boldsymbol{x} - \boldsymbol{\mu}^{(l)})^T \boldsymbol{\Sigma}^{-1(l)} \mathcal{N}_{\mathbf{x}}(\boldsymbol{x}; \boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \mathrm{d}\boldsymbol{x} \tag{3.22}$$

$$= \left(\boldsymbol{R}_{k,1} - \boldsymbol{I}_k \boldsymbol{\mu}^{(l)T} \right) \boldsymbol{\Sigma}^{-1(l)} \,, \tag{3.23}$$

which is true for $k \geq 1$. Here, $\boldsymbol{R}_{k,l}$ is the mixed moment

$$\boldsymbol{R}_{k,l} = \int \cdots \int \boldsymbol{x}^k (\boldsymbol{x}^l)^T f_{\mathbf{x}}(\boldsymbol{x}) \mathcal{N}_{\mathbf{x}}(\boldsymbol{x}; \boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \mathrm{d}\boldsymbol{x} \,, \tag{3.24}$$

which is matrix-notation for (3.17). Thus, $\mathfrak{G} = \boldsymbol{R}_{1,1}/I_0 - \boldsymbol{F}\boldsymbol{F}^T$. The Jacobian of $\boldsymbol{F}$ is given by

$$\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\mu}^{(l)}} = \frac{1}{I_0} \frac{\partial \boldsymbol{I}_1}{\partial \boldsymbol{\mu}^{(l)}} - \frac{\boldsymbol{I}_1}{I_0^2} \left(\frac{\partial I_0}{\partial \boldsymbol{\mu}^{(l)}} \right)^T \tag{3.25}$$

$$= \frac{1}{I_0} (\boldsymbol{R}_{1,1} - \boldsymbol{I}_1 \boldsymbol{\mu}^{(l)T}) \boldsymbol{\Sigma}^{-1(l)} - \frac{\boldsymbol{I}_1}{I_0^2} \left(\left(\boldsymbol{I}_1 - I_0 \boldsymbol{\mu}^{(l)T} \right) \boldsymbol{\Sigma}^{-1(l)} \right)^T . \tag{3.26}$$

Since $\boldsymbol{\Sigma}^{-1(l)}$ is the inverse of a covariance matrix, it is identical to its transpose. Thus,

$$\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\mu}^{(l)}} = \left(\frac{\boldsymbol{R}_{1,1}}{I_0} - \boldsymbol{F}\boldsymbol{\mu}^{(l)T} - \frac{\boldsymbol{I}_1 \boldsymbol{I}_1^T}{I_0^2} + \frac{\boldsymbol{I}_1 \boldsymbol{\mu}^{(l)T}}{I_0} \right) \boldsymbol{\Sigma}^{-1(l)} \tag{3.27}$$

$$= \left(\frac{\boldsymbol{R}_{1,1}}{I_0} - \boldsymbol{F}\boldsymbol{F}^T \right) \boldsymbol{\Sigma}^{-1(l)} \tag{3.28}$$

$$= \mathfrak{G} \boldsymbol{\Sigma}^{-1(l)} \,. \tag{3.29}$$

Algorithms like VAMP [50] might need to make use of these general expressions. While both (3.3) and the more general relationship (3.29) also hold for non-diagonal covariance matrices $\boldsymbol{\Sigma}^{(l)}$, this chapter focuses on formulations for use in Bayesian AMP algorithms. Thus, diagonal $\boldsymbol{\Sigma}^{(l)}$ is assumed. Results presented here are optimized for this case and focus on $\boldsymbol{G}$ instead of the matrix-valued $\mathfrak{G}$.

3.1 Composite Priors

Deriving an expression for complicated priors which might arise from particular problems can prove difficult. Often, it is more appropriate to approximate this prior by a linear combination of simple priors, for which expressions of $I_{k,i}$ are known. Moreover, high-dimensional priors can be constructed as a product of low-dimensional or even scalar priors. A combination of both constructions can be used as a powerful tool to construct complicated priors.

Linear Mixture. Note that the functions $I_{k,i}$ are linear towards the prior $f_{\mathbf{x}}(\boldsymbol{x})$, cf. (3.4). Let $f_{\mathbf{x}}(\boldsymbol{x})$ be a linear mixture,

$$f_{\mathbf{x}}(\boldsymbol{x}) = \sum_{m=1}^{M} a_m f_{\mathbf{x}}^{(m)}(\boldsymbol{x}) \,, \tag{3.30}$$

where $f_{\mathbf{x}}^{(m)}(\boldsymbol{x})$ are pdfs of identical dimension and the variables a_m are weights, with $a_m > 0$, $\sum_m a_m = 1$. Then, the function $I_{k,i}$ for the linear combination can itself be written as linear mixture

$$I_{k,i} = \sum_{m=1}^{M} a_m I_{k,i}^{(m)} \,. \tag{3.31}$$

Note that the functions F, F' and G *cannot* be written as linear mixture. For $I_{k,i}$, the derivatives are

$$\frac{\partial I_{k,i}}{\partial \mu_j^{(l)}} = \sum_{m=1}^{M} a_m \frac{\partial I_{k,i}^{(m)}}{\partial \mu_j^{(l)}} \,. \tag{3.32}$$

Product Densities. It is possible to obtain a high-dimensional pdf as the product of multiple low-dimensional or scalar pdfs. Since the normal distribution representing the local beliefs in $I_{k,i}$ is uncorrelated, it can be factored ($\boldsymbol{\sigma}^{2(l)}$ is the diagonal of the diagonal covariance matrix $\boldsymbol{\Sigma}^{(l)}$). Let

$$f_{\mathbf{x}}(\boldsymbol{x}) = \prod_{m=1}^{M} f_{\mathbf{x}_m}(\boldsymbol{x}_m) \,, \tag{3.33}$$

where the functions $f_{\mathbf{x}_m}(\boldsymbol{x}_m)$ are pdfs of arbitrary dimension for which expressions $I_{k,i}^{(m)}$ are known. The random vectors $\mathbf{x}_m$ are distinct. Then the function $I_{k,i}$ for the prior $f_{\mathbf{x}}(\boldsymbol{x})$ can be written as

$$I_{k,i}(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) = I_{k,j}^{(n)}(\boldsymbol{\mu}_n^{(l)}, \boldsymbol{\sigma}_n^{2(l)}) \prod_{\substack{m=1 \\ m \neq n}}^{M} I_0^{(m)}(\boldsymbol{\mu}_m^{(l)}, \boldsymbol{\sigma}_m^{2(l)}) \,, \tag{3.34}$$

where the i^{th} entry of a vector $\boldsymbol{x}$ is the j^{th} entry of $\boldsymbol{x}_m$ and it was assumed that the vectors $\boldsymbol{x}$, $\boldsymbol{\mu}^{(l)}$ and $\boldsymbol{\sigma}^{2(l)}$ can be written as

$$\boldsymbol{x} = (\boldsymbol{x}_1^T, \ldots, \boldsymbol{x}_m^T, \ldots, \boldsymbol{x}_M^T)^T \tag{3.35}$$

$$\boldsymbol{\mu}^{(l)} = (\boldsymbol{\mu}_1^{(l)T}, \ldots, \boldsymbol{\mu}_m^{(l)T}, \ldots, \boldsymbol{\mu}_M^{(l)T})^T \tag{3.36}$$

$$\boldsymbol{\sigma}^{2(l)} = (\boldsymbol{\sigma}_1^{2(l)T}, \ldots, \boldsymbol{\sigma}_m^{2(l)T}, \ldots, \boldsymbol{\sigma}_M^{2(l)T})^T \,. \tag{3.37}$$

The functions F, G and F' can be expressed in terms of the component functions $I_{k,i}^{(m)}$. Let

$$\boldsymbol{\mu}_{\mathbf{x}} = \boldsymbol{F}(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \tag{3.38}$$

$$\boldsymbol{\sigma}_{\mathbf{x}}^2 = \boldsymbol{G}(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \tag{3.39}$$

$$\boldsymbol{\mu}_{\mathbf{x}_m} = \boldsymbol{F}^{(m)}(\boldsymbol{\mu}_m^{(l)}, \boldsymbol{\sigma}_m^{2(l)}) = \frac{\boldsymbol{I}_1^{(m)}}{I_0^{(m)}} \tag{3.40}$$

$$\boldsymbol{\sigma}_{\mathbf{x}_m}^2 = \boldsymbol{G}^{(m)}(\boldsymbol{\mu}_m^{(l)}, \boldsymbol{\sigma}_m^{2(l)}) = \frac{\boldsymbol{I}_2^{(m)}}{I_0^{(m)}} - \boldsymbol{F}^{2(m)} \,, \tag{3.41}$$

where $\boldsymbol{F}^{2(m)}$ is squared component-wise. Then, the vectors $\boldsymbol{\mu}_{\mathbf{x}}$ and $\boldsymbol{\sigma}_{\mathbf{x}}^2$ can be written as

$$\boldsymbol{\mu}_{\mathbf{x}} = (\boldsymbol{\mu}_{\mathbf{x}_1}^T, \ldots, \boldsymbol{\mu}_{\mathbf{x}_m}^T, \ldots, \boldsymbol{\mu}_{\mathbf{x}_M}^T) \tag{3.42}$$

$$\boldsymbol{\sigma}_{\mathbf{x}}^2 = (\boldsymbol{\sigma}_{\mathbf{x}_1}^{2T}, \ldots, \boldsymbol{\sigma}_{\mathbf{x}_m}^{2T}, \ldots, \boldsymbol{\sigma}_{\mathbf{x}_M}^{2T})^T. \tag{3.43}$$

Similarly, the derivatives of $I_{k,i}$ and F can be computed component-wise with the derivative w.r.t. $\mu_j^{(l)}$ only depending on the component $I_{k,i}^{(m)}$ or $F_i^{(m)}$ for which $\mu_j^{(l)}$ is an entry of the vector $\boldsymbol{\mu}_m^{(l)}$. The derivatives w.r.t. dimensions which are not argument to a particular component function are always zero. Product densities are therefore only useful in conjunction with other compositions, e.g. linear mixtures.

3.2 Transformations

Sometimes it is useful to apply simple transformations to a prior in order to obtain a different prior without having to re-derive the functions I_k. The most common of these transformations are discussed here for the one-dimensional case.
Some of the expressions involve a number of terms which cancel out arithmetically. Numerically, these lead to inaccuracies. These transformations should therefore only be used if the distribution does not support them intrinsically. The normal, uniform and Dirac distributions all support translation (by a change of the mean) and normal, uniform and exponential distributions support linear scaling of the x-axis (by change of the variance, boundaries or decay parameter respectively).

The aim of this section is to derive the functions I_k^t, F^t and G^t for transformed priors. These functions are expressions of I_k, F and G, i.e. functions of the non-transformed priors. Since this section discusses the one-dimensional case, the index i is not needed for the functions $I_{k,i}$, F_i and G_i. Whenever arguments are missing, they are $(\mu^{(l)}, \sigma^{2(l)})$ for the transformed variant and (u, s^2) for the regular one. The variables u and s^2 are defined separately for each transformation.

Translation. A distribution $f_\mathsf{x}(x)$ can be shifted on the x-axis by a constant value v. This amounts to convolution with the distribution $\delta(x - v)$, which shifts the distribution towards larger x by v. The functions $I_k(\dots)$ are

$$I_k(\mu^{(l)}, \sigma^{2(l)}) = \int x^k f_\mathsf{x}(x) \mathcal{N}_\mathsf{x}(\mu^{(l)}, \sigma^{2(l)}) \mathrm{d}x \,. \tag{3.44}$$

Shifting the prior amounts to

$$I_k^t(\mu^{(l)}, \sigma^{2(l)}) = \int x^k f_\mathsf{x}(x - v) \mathcal{N}_\mathsf{x}(x; \mu^{(l)}, \sigma^{2(l)}) \mathrm{d}x \,, \tag{3.45}$$

$$\text{with } \chi = x - v \tag{3.46}$$

$$\mathrm{d}\chi = \mathrm{d}x \tag{3.47}$$

$$= \int (\chi + v)^k f_\mathsf{x}(\chi) \mathcal{N}_\mathsf{x}(\chi + v; \mu^{(l)}, \sigma^{2(l)}) \mathrm{d}\chi \,. \tag{3.48}$$

The normal distribution can be expressed as

$$\mathcal{N}_\mathsf{x}(\chi + v; \mu^{(l)}, \sigma^{2(l)}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\chi + v - \mu^{(l)})^2}{2\sigma^{2(l)}} \right) \tag{3.49}$$

$$= \mathcal{N}_\mathsf{x}(\chi; \mu - v, \sigma^2) \,. \tag{3.50}$$

For a shift along the x-axis, the values u and s^2 are thus

$$u = \mu^{(l)} - v \tag{3.51}$$

$$s^2 = \sigma^{2(l)} \,. \tag{3.52}$$

Thus, the functions I_k^t for shifted priors can be expressed in terms of the original functions I_k:

$$I_k^t(\mu^{(l)}, \sigma^{2(l)}) = \int (\chi + v)^k f_\mathsf{x}(\chi) \mathcal{N}_\mathsf{x}(\chi; u, s^2) \mathrm{d}\chi \tag{3.53}$$

$$= \sum_{m=0}^{k} \binom{k}{m} v^m I_{k-m}(u, s^2) \tag{3.54}$$

$$I_0^t(\mu^{(l)}, \sigma^{2(l)}) = \int f_\mathsf{x}(\chi) \mathcal{N}_\mathsf{x}(\chi; u, s^2) \mathrm{d}\chi \tag{3.55}$$

$$= I_0(u, s^2) \tag{3.56}$$

$$I_1^t(\mu^{(l)}, \sigma^{2(l)}) = \int (\chi + v) f_{\mathrm{x}}(\chi) \mathcal{N}_{\mathrm{x}}(\chi; u, s^2) \mathrm{d}\chi \tag{3.57}$$

$$= I_1(u, s^2) + v I_0(u, s^2) \tag{3.58}$$

$$I_2^t(\mu^{(l)}, \sigma^{2(l)}) = \int (\chi + v)^2 f_{\mathrm{x}}(\chi) \mathcal{N}_{\mathrm{x}}(\chi; u, s^2) \mathrm{d}\chi \tag{3.59}$$

$$= I_2(u, s^2) + 2v I_1(u, s^2) + v^2 I_0(u, s^2) \tag{3.60}$$

$$F^t(\mu^{(l)}, \sigma^{2(l)}) = \frac{I_1^t}{I_0^t} = \frac{I_1(u, s^2)}{I_0(u, s^2)} + v = F(u, s^2) + v \tag{3.61}$$

$$G^t(\mu^{(l)}, \sigma^{2(l)}) = \frac{I_2^t}{I_0^t} - F^{t2} \tag{3.62}$$

$$= \frac{I_2(u, s^2)}{I_0(u, s^2)} + 2vF(u, s^2) + v^2 - F^{t2}(\dots) \tag{3.63}$$

$$= \frac{I_2(u, s^2)}{I_0(u, s^2)} - F^2(u, s^2) \tag{3.64}$$

$$= G(u, s^2)\,. \tag{3.65}$$

To compute the derivative w.r.t. $\mu^{(l)}$ of the transformed functions, the derivative of $\mathcal{N}_{\mathrm{x}}(\chi; u, s^2)$ needs to be computed. With $u = \mu^{(l)} - v$ and $\partial u = \partial \mu^{(l)}$ this evaluates to

$$\frac{\partial \mathcal{N}(u, s^2)}{\partial u} = \frac{\chi - u}{s^2} \mathcal{N}_{\mathrm{x}}(\chi; u, s^2) \tag{3.66}$$

$$= \frac{\chi + v - \mu^{(l)}}{\sigma^{2(l)}} \mathcal{N}_{\mathrm{x}}(\chi; \mu^{(l)} - v, \sigma^{2(l)})\,. \tag{3.67}$$

The derivatives are therefore obtained as

$$\frac{\partial I_k^t}{\partial \mu^{(l)}} = \frac{1}{\sigma^{2(l)}} \left(\int (\chi + v)^k (\chi + v) f_{\mathrm{x}}(\chi) \mathcal{N}_{\mathrm{x}}(\chi; u, s^2) \mathrm{d}\chi \right. \tag{3.68}$$

$$\left. + \mu^{(l)} \int (\chi + v)^k f_{\mathrm{x}}(\chi) \mathcal{N}_{\mathrm{x}}(\chi; u, s^2) \mathrm{d}\chi \right) \tag{3.69}$$

$$= \frac{1}{\sigma^{2(l)}} \left(I_{k+1}^t - \mu^{(l)} I_k^t \right) \tag{3.70}$$

$$\frac{\partial I_0^t(\dots)}{\partial \mu^{(l)}} = \frac{1}{\sigma^{2(l)}} \left(I_1 + v I_0 - \mu^{(l)} I_0 \right) \tag{3.71}$$

$$\frac{\partial I_1^t}{\partial \mu^{(l)}} = \frac{1}{\sigma^{2(l)}} \left(I_2 + 2v I_1 + v^2 I_0 - \mu^{(l)} (I_1 + v I_0) \right) \tag{3.72}$$

$$\frac{\partial F^t}{\partial \mu^{(l)}} = \frac{1}{I_0^{t2}} \left(I_0^t \frac{\partial I_1^t}{\partial \mu^{(l)}} - I_1^t \frac{\partial I_0^t}{\partial \mu^{(l)}} \right) \tag{3.73}$$

$$= \frac{1}{\sigma^{2(l)}} \left(\frac{I_2 + 2v I_1 + v^2 I_0 - \mu^{(l)} I_1 - \mu^{(l)} v I_0}{I_0} \right. \tag{3.74}$$

$$-\frac{(I_1+vI_0)(I_1+vI_0-\mu^{(l)}I_0)}{I_0^2}\Bigg) \tag{3.75}$$

$$= \frac{1}{\sigma^{2(l)}}\left(\frac{I_2}{I_0}+2vF+v^2-\mu^{(l)}F-\mu^{(l)}v \right. \tag{3.76}$$

$$\left. -(F^2+vF-\mu^{(l)}F+vF+v^2-\mu^{(l)}v)\right) \tag{3.77}$$

$$= \frac{1}{\sigma^{2(l)}}\left(\frac{I_2}{I_0}-F^2\right) = \frac{1}{\sigma^{2(l)}}G(u,s^2)\,. \tag{3.78}$$

Scaling. This transformation has the effect of "stretching" (or compressing) the prior. Let χ be the scaled variable

$$\chi = ax\,, \tag{3.79}$$

with $a \in \mathbb{R}\setminus\{0\}$. The transformed pdf needs to be normalized, i.e.

$$1 = \int \gamma f_{\mathsf{x}}(ax)\mathrm{d}x\,. \tag{3.80}$$

With $\mathrm{d}\chi = a\mathrm{d}x$, the constant γ is determined as

$$1 = \gamma\int f_{\mathsf{x}}(\chi)\frac{1}{a}\mathrm{d}\chi \tag{3.81}$$

$$= \frac{\gamma}{a}\underbrace{\int f_{\mathsf{x}}(\chi)\mathrm{d}\chi}_{=1} \tag{3.82}$$

$$\gamma = a\,. \tag{3.83}$$

The functions I_k^t with a scaled prior can be written as

$$I_k^t(\mu^{(l)},\sigma^{2(l)}) = \int x^k a f_{\mathsf{x}}(ax)\mathcal{N}_{\mathsf{x}}(x;\mu^{(l)},\sigma^{2(l)})\mathrm{d}x \tag{3.84}$$

$$= \int \left(\frac{\chi}{a}\right)^k a f_{\mathsf{x}}(\chi)\mathcal{N}_{\mathsf{x}}(\frac{\chi}{a};\mu^{(l)},\sigma^{2(l)})\frac{1}{a}\mathrm{d}\chi\,. \tag{3.85}$$

The normal distribution $\mathcal{N}_{\mathsf{x}}(\frac{\chi}{a};\mu^{(l)},\sigma^{2(l)})$ needs to be expressed in terms of χ:

$$\mathcal{N}_{\mathsf{x}}(\frac{\chi}{a};\mu^{(l)},\sigma^{2(l)}) = \frac{1}{\sqrt{2\pi\sigma^{2(l)}}}\exp\left(-\frac{\left(\frac{\chi}{a}-\mu^{(l)}\right)^2}{2\sigma^{2(l)}}\right) \tag{3.86}$$

$$= \frac{1}{\sqrt{2\pi\sigma^{2(l)}}}\exp\left(-\frac{\left(\chi-a\mu^{(l)}\right)^2}{2a^2\sigma^{2(l)}}\right). \tag{3.87}$$

With $u = a\mu^{(l)}$ and $s^2 = a^2\sigma^{2(l)}$, the normal distribution can be written as

$$\mathcal{N}_{\mathsf{x}}(\chi;u,s^2) = \frac{1}{\sqrt{2\pi s^2}}\exp\left(-\frac{(\chi-u)^2}{2s^2}\right). \tag{3.88}$$

Note that (3.88) integrates to one because $\sigma^{2(l)}$ was replaced by s^2 in the normalization factor. This is not necessary for the Bayesian AMP algorithm to work, however, since messages are always normalized. The functions $I_k^t(\dots)$ can be written as

$$I_k^t(\mu^{(l)}, \sigma^{2(l)}) = a^{-k} \int \chi^k f_{\mathrm{x}}(\chi) \mathcal{N}_{\mathrm{x}}(\chi; u, s^2) \mathrm{d}\chi = a^{-k} I_k(u, s^2) \tag{3.89}$$

$$F^t(\mu^{(l)}, \sigma^{2(l)}) = \frac{I_1^t}{I_0^t} = \frac{a^{-1} I_1}{I_0} = a^{-1} F(u, s^2) \tag{3.90}$$

$$G^t(\mu^{(l)}, \sigma^{2(l)}) = \frac{I_2^t}{I_0^t} - F^{t2} = \frac{a^{-2} I_2}{I_0} - a^{-2} F^2 \tag{3.91}$$

$$= a^{-2} G(u, s^2)\,. \tag{3.92}$$

The derivatives are

$$\frac{\partial I_k^t(\mu^{(l)}, \sigma^{2(l)})}{\partial \mu^{(l)}} = \frac{\partial}{\partial \mu^{(l)}} a^{-k} I_k(u, s^2) \tag{3.93}$$

$$= \frac{a^{-k}}{a^2 \sigma^{2(l)}} \left(I_{k+1} - a\mu^{(l)} I_k \right) a \tag{3.94}$$

$$= \frac{1}{\sigma^{2(l)}} \left(a^{-(k+1)} I_{k+1} - a^{-k} \mu^{(l)} I_k \right) \tag{3.95}$$

$$= \frac{1}{\sigma^{2(l)}} \left(I_{k+1}^t - \mu^{(l)} I_k^t \right) \tag{3.96}$$

$$\frac{\partial F^t(\mu^{(l)}, \sigma^{2(l)})}{\partial \mu^{(l)}} = \frac{1}{I_0^t} \frac{\partial I_1^t}{\partial \mu^{(l)}} - \frac{I_1^t}{I_0^{m2}} \frac{\partial I_0^t}{\partial \mu^{(l)}} \tag{3.97}$$

$$= \frac{1}{\sigma^{2(l)}} \left(\frac{a^{-2} I_2 - a^{-1} \mu^{(l)} I_1}{I_0} - \frac{a^{-1} I_1 (a^{-1} I_1 - \mu^{(l)} I_0)}{I_0^2} \right) \tag{3.98}$$

$$= \frac{1}{\sigma^{2(l)}} \left(a^{-2} \frac{I_2}{I_0} - a^{-1} \mu^{(l)} F - a^{-2} F^2 + a^{-1} \mu^{(l)} F \right) \tag{3.99}$$

$$= \frac{1}{\sigma^{2(l)}} G^t(\mu^{(l)}, \sigma^{2(l)}) = \frac{1}{s^2} G(u, s^2)\,. \tag{3.100}$$

Mirroring across the y-Axis. Mirroring priors across the y-axis is equivalent to scaling with $a = -1$. Since it is an important case and allows for some simplification, the associated expressions are explicitly stated here:

$$u = -\mu^{(l)} \tag{3.101}$$

$$s = \sigma^{2(l)} \tag{3.102}$$

$$I_k^t(\mu^{(l)}, \sigma^{2(l)}) = (-1)^k I_k(u, s^2) \tag{3.103}$$

$$F^t(\mu^{(l)}, \sigma^{2(l)}) = -F(u, s^2) \tag{3.104}$$

$$G^t(\mu^{(l)}, \sigma^{2(l)}) = G(u, s^2) \tag{3.105}$$

$$\frac{\partial I_k^t(\mu^{(l)}, \sigma^{2(l)})}{\partial \mu^{(l)}} = \frac{1}{\sigma^{2(l)}} \left((-1)^{-(k+1)} I_{k+1} - (-1)^{-k} \mu^{(l)} I_k \right) \tag{3.106}$$

$$\frac{\partial F^t(\mu^{(l)}, \sigma^{2(l)})}{\partial \mu^{(l)}} = \frac{1}{\sigma^{2(l)}} G^t(\mu^{(l)}, \sigma^{2(l)}) = \frac{1}{s^2} G(u, s^2) \,. \tag{3.107}$$

Extensions for Multivariate Priors. For multivariate priors, the functions $\boldsymbol{I}_k$, $\boldsymbol{F}(\dots)$ and $\boldsymbol{G}(\dots)$ are vector-valued, i.e. their arguments and resulting values are vectors. They are evaluated component-wise however, thus extension of the expressions above towards multivariate priors is straightforward. For example, he transformed function $\boldsymbol{I}_k^t$ for a prior $f_{\mathbf{x}}(\boldsymbol{x})$ shifted by $\boldsymbol{v}$ is

$$\boldsymbol{I}_k^t(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) = \int \cdots \int \boldsymbol{x}^k f_{\mathbf{x}}(\boldsymbol{x} - \boldsymbol{v}) \mathcal{N}_{\mathbf{x}}(\boldsymbol{x}; \boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \mathrm{d}\boldsymbol{x} \tag{3.108}$$

$$\text{with } \boldsymbol{\chi} = \boldsymbol{x} - \boldsymbol{v} \tag{3.109}$$

$$\mathrm{d}\boldsymbol{\chi} = \mathrm{d}\boldsymbol{x} \tag{3.110}$$

$$= \int \cdots \int (\boldsymbol{\chi} + \boldsymbol{v})^k f_{\mathbf{x}}(\boldsymbol{\chi}) \mathcal{N}_{\mathbf{x}}(\boldsymbol{\chi} + \boldsymbol{v}; \boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \mathrm{d}\boldsymbol{\chi} \,. \tag{3.111}$$

The expressions (3.108) to (3.111) again slightly abuse notation and define vector exponentiation in a component-wise context, e.g. the expression $\boldsymbol{x}^k = (\dots, x_i^k, \dots)^T$ and $\boldsymbol{x}^0 = \mathbf{1}$. The element-wise approach is also valid for all diagonal entries of the Jacobian, i.e. the derivatives of I_k's i^{th} dimension w.r.t. $\mu_i^{(l)}$. The off-diagonal entries cannot easily be obtained by extension, however. Let $\mathcal{T}$ be the inverse of a particular transformation $\mathcal{U}$, e.g. for $\mathcal{U}\{x\} : x \mapsto x - v$, $\mathcal{T}\{\chi\} : \chi \mapsto \chi + v$. Furthermore, let $\boldsymbol{u}$ and $\boldsymbol{s}^2$ be transformed versions of $\boldsymbol{\mu}^{(l)}$ and $\boldsymbol{\sigma}^{2(l)}$. Then the Jacobian of $\boldsymbol{I}_k^t$ w.r.t. $\boldsymbol{\mu}^{(l)}$ is

$$\frac{\partial \boldsymbol{I}_k^t(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)})}{\partial \boldsymbol{\mu}^{(l)}} = \int \cdots \int \mathcal{T}\{\boldsymbol{\chi}\}^k f_{\mathbf{x}}(\boldsymbol{\chi}) \frac{\partial}{\partial \boldsymbol{\mu}^{(l)}} \mathcal{N}_{\mathbf{x}}(\boldsymbol{\chi}; \boldsymbol{u}, \boldsymbol{s}^2) \mathrm{d}\boldsymbol{\chi} \,. \tag{3.112}$$

The derivative of the multivariate normal distribution $\mathcal{N}_{\mathbf{x}}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ w.r.t. $\boldsymbol{\mu}$ is a vector:

$$\frac{\partial \mathcal{N}_{\mathbf{x}}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}} = \mathcal{N}_{\mathbf{x}}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \,. \tag{3.113}$$

In the context of AMP, the covariance matrix $\boldsymbol{\Sigma}$ is always diagonal and (3.112) becomes

$$\frac{\partial \boldsymbol{I}_k^t(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)})}{\partial \boldsymbol{\mu}^{(l)}} = \boldsymbol{s}^{-2} \int \cdots \int \mathcal{T}\{\boldsymbol{\chi}\}^k f_{\mathbf{x}}(\boldsymbol{\chi}) \left((\boldsymbol{\chi} - \boldsymbol{u}) \otimes \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\mu}^{(l)}} \right)^T \mathcal{N}_{\mathbf{x}}(\boldsymbol{\chi}; \boldsymbol{u}, \boldsymbol{s}^2) \mathrm{d}\boldsymbol{\chi} \,. \tag{3.114}$$

In (3.114), the operator $\otimes$ denotes the component-wise multiplication, the division $\partial \boldsymbol{u} / \partial \boldsymbol{\mu}^{(l)}$ is performed component-wise and $\boldsymbol{s}^{-2}$ is the inverse of the diagonal covariance matrix with entries s_i^2. Since $\mathcal{N}_{\mathbf{x}}(\boldsymbol{\chi}; \boldsymbol{u}, \boldsymbol{s}^2) = \mathcal{N}_{\mathbf{x}}(\mathcal{T}\{\boldsymbol{\chi}\}; \boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)})$, however,

an off-diagonal entry can also be written as

$$\frac{\partial I^t_{k,i}(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)})}{\partial \mu^{(l)}_j} = \frac{1}{\sigma^{2(l)}_j} \int \cdots \int f_{\mathbf{x}}(\boldsymbol{\chi}) \mathcal{T}\{\chi_i\}^k (\mathcal{T}\{\chi_j\} - \mu_j) \mathcal{N}_{\mathbf{x}}(\mathcal{T}\{\boldsymbol{\chi}\}; \boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)}) \mathrm{d}\boldsymbol{\chi} \tag{3.115}$$

$$= \frac{1}{\sigma^{2(l)}_j} \left(\mathbb{E}\left\{ \mathcal{T}\{\chi_i\}^k \mathcal{T}\{\chi_j\} \right\} - \mu_j \mathbb{E}\left\{ \mathcal{T}\{\chi_i\}^k \right\} \right), \tag{3.116}$$

where the expectation is w.r.t. the (unnormalized) pdf $f_{\mathbf{x}}(\boldsymbol{\chi}) \mathcal{N}_{\mathbf{x}}(\mathcal{T}\{\boldsymbol{\chi}\}; \boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)})$. For multivariate priors with independent dimensions, the above becomes

$$\frac{\partial I^t_{k,i}(\boldsymbol{\mu}^{(l)}, \boldsymbol{\sigma}^{2(l)})}{\partial \mu^{(l)}_j} = \frac{1}{\sigma^{2(l)}_j} \left(\mathbb{E}\left\{ \mathcal{T}\{\chi_i\}^k \right\} \mathbb{E}\left\{ \mathcal{T}\{\chi_j\} \right\} - \mu_j \mathbb{E}\left\{ \mathcal{T}\{\chi_i\}^k \right\} \right) \tag{3.117}$$

$$= \frac{1}{\sigma^{2(l)}_j} I^t_{k,i} \left(I^t_{1,j} - \mu_j \right) . \tag{3.118}$$

For translations, $\boldsymbol{u} = \boldsymbol{\mu}^{(l)} - \boldsymbol{v}$, $\boldsymbol{s}^2 = \boldsymbol{\sigma}^{2(l)}$, $\partial \boldsymbol{u} / \partial \boldsymbol{\mu}^{(l)} = \mathbf{1}$ and $\mathcal{T}\{\boldsymbol{\chi}\} : \boldsymbol{\chi} \mapsto \boldsymbol{\chi} + \boldsymbol{v}$. The off-diagonal elements thus evaluate to

$$\frac{\partial I_{k,i}}{\partial \mu^{(l)}_j} = s_j^{-2} (R^t_{k,i,j} - u_j I^t_{k,i}) . \tag{3.119}$$

Here, the mixed moment $R^t_{k,i,j}$ is defined as

$$R^t_{k,i,j} = \int \cdots \int \mathcal{T}\{\chi_i\}^k \chi_j f_{\mathbf{x}}(\boldsymbol{\chi}) \mathcal{N}_{\mathbf{x}}(\boldsymbol{\chi}; \boldsymbol{u}, \boldsymbol{s}^2) \mathrm{d}\boldsymbol{\chi} \tag{3.120}$$

$$= \int \cdots \int (\chi_i + v_i)^k \chi_j f_{\mathbf{x}}(\boldsymbol{\chi}) \mathcal{N}_{\mathbf{x}}(\boldsymbol{\chi}; \boldsymbol{u}, \boldsymbol{s}^2) \mathrm{d}\boldsymbol{\chi} \tag{3.121}$$

$$= \sum_{m=0}^{k} \binom{k}{m} v_i^m R_{k-m,1,i,j}(\boldsymbol{u}, \boldsymbol{s}^2) , \tag{3.122}$$

where $R_{k,l,i,j}$ as defined in (3.17). Note that $R_{0,1,i,j} = I_{1,j}$. For the scaling transformation, $\boldsymbol{u} = \boldsymbol{a} \otimes \boldsymbol{\mu}^{(l)}$, $\boldsymbol{s}^2 = \boldsymbol{a}^2 \otimes \boldsymbol{\sigma}^{2(l)}$, $\partial \boldsymbol{u} / \partial \boldsymbol{\mu}^{(l)} = \boldsymbol{a}$ and $\mathcal{T}\{\boldsymbol{\chi}\} : \boldsymbol{\chi} \mapsto \boldsymbol{\chi} / \boldsymbol{a}$, with both "vector-divisions" to be applied component-wise. The off-diagonals are therefore obtained as

$$\frac{\partial I_{k,i}}{\partial \mu^{(l)}_j} = a_j s_j^{-2} (R^t_{k,i,j} - u_j I^t_{k,i}) . \tag{3.123}$$

In this case, the mixed moment $R^t_{k,i,j}$ resolves to

$$R^t_{k,i,j} = \int \cdots \int \mathcal{T}\{\chi_i\}^k \chi_j f_{\mathbf{x}}(\boldsymbol{\chi}) \mathcal{N}_{\mathbf{x}}(\boldsymbol{\chi}; \boldsymbol{u}, \boldsymbol{s}^2) \mathrm{d}\boldsymbol{\chi} \tag{3.124}$$

$$= a_j s_j^{-2} a_i^{-k} R_{k,1,i,j} , \tag{3.125}$$

with $R_{k,l,i,j}$ defined as in (3.17).

3.3 Dirac Distribution

The distribution $f_{\mathsf{x}}(x)$ defined as

$$f_{\mathsf{x}}(x) = \delta(x - \mu_p) \tag{3.126}$$

is the Dirac delta with mean μ_p. It is zero everywhere except at μ_p and integrates to 1. This distribution is crucial in compressed sensing since it can be used to model sparse signals. Furthermore, discrete distributions can be constructed using linear mixes of Dirac deltas, which is of particular interest when compressive sensing is used in conjunction with channel codes (e.g. [52, 8]). The functions I_k, F and G are

$$I_k = \mu_p^k \left(2\pi\sigma^{2(l)}\right)^{-\frac{1}{2}} \exp\left(-\frac{(\mu_p - \mu^{(l)})^2}{2\sigma^{2(l)}}\right) \tag{3.127}$$

$$F = \mu_p \tag{3.128}$$

$$G = 0\,. \tag{3.129}$$

The M-dimensional Dirac delta distribution $f_{\mathbf{x}}(\boldsymbol{x})$ can be written as the product of scalar Dirac deltas,

$$f_{\mathbf{x}}(\boldsymbol{x}) = \delta(\boldsymbol{x} - \boldsymbol{\mu}_p) = \prod_{m=1}^{M} \delta(x_m - \mu_{p,m})\,. \tag{3.130}$$

The vector valued versions of I_0, $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ are

$$I_0 = \left((2\pi)^M \det(\boldsymbol{\Sigma}^{(l)})\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{\mu_p} - \boldsymbol{\mu}^{(l)})^T \boldsymbol{\Sigma}^{-1(l)} (\boldsymbol{\mu}_p - \boldsymbol{\mu}^{(l)})\right) \tag{3.131}$$

$$\boldsymbol{I}_1 = \boldsymbol{\mu}_p I_0 \tag{3.132}$$

$$\boldsymbol{I}_2 = \operatorname{diag}(\boldsymbol{\mu}_p \boldsymbol{\mu}_p^T) I_0\,. \tag{3.133}$$

In (3.131), the matrix $\boldsymbol{\Sigma}^{(l)}$ is a diagonal covariance matrix with entries $\Sigma_{i,i}^{(l)} = \sigma_i^{2(l)}$ and $\operatorname{diag}(\boldsymbol{U})$ is the vector at the main diagonal of the matrix $\boldsymbol{U}$. The derivative of I_k can be written as

$$\frac{\partial I_k}{\partial \mu^{(l)}} = \frac{\mu_p - \mu^{(l)}}{\sigma^{2(l)}} I_k\,. \tag{3.134}$$

For the multivariate case, the derivative is

$$\frac{\partial I_{k,i}}{\partial \mu_j^{(l)}} = \frac{\mu_{p,j} - \mu_j^{(l)}}{\sigma_j^{2(l)}} I_{k,i}\,. \tag{3.135}$$

The Dirac distribution becomes interesting in conjunction with linear mixtures since the functions F, G and F' are otherwise constant, i.e. they do not depend on

their arguments. Two examples are presented to graphically illustrate the operation of the functions F and G. These are on the one hand the (one-dimensional) binary prior

$$f_{\mathrm{x,Bi}}(x) = \frac{1}{2}\left(\delta(x+1) + \delta(x-1)\right) \ , \tag{3.136}$$

on the other hand the two-dimensional quaternary prior

$$f_{\mathbf{x},\mathrm{Qr}}(\boldsymbol{x}) = \frac{1}{4}\left(\delta(\boldsymbol{x} - (1,0)^T) + \delta(\boldsymbol{x} - (0,1)^T)\right. \tag{3.137}$$

$$\left. + \delta(\boldsymbol{x} - (-1,0)^T) + \delta(\boldsymbol{x} - (0,-1)^T)\right) \ . \tag{3.138}$$

These functions are plotted in Fig. 3.1 for the binary prior. Note that F can be expressed as a hyperbolic tangent and G is proportional to $1-\tanh^2(x)$. In Fig. 3.2 and Fig. 3.3, results for the quaternary prior are plotted. The "decision regions" can clearly be distinguished. Note that κ-dimensional priors yield denoisers of the same dimension, i.e. the input and output for $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{F}'$, $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ are κ-dimensional, which makes graphical descriptions hard for $\kappa > 2$.

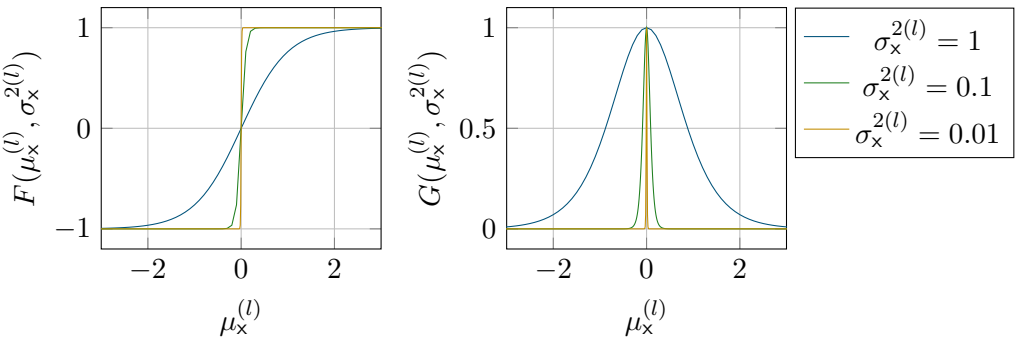


Figure 3.1: Functions F (left) and G (right) for the binary prior $f_{\mathrm{x,Bi}}(x)$ and various local variances $\sigma_{\mathrm{x}}^{2(l)}$.

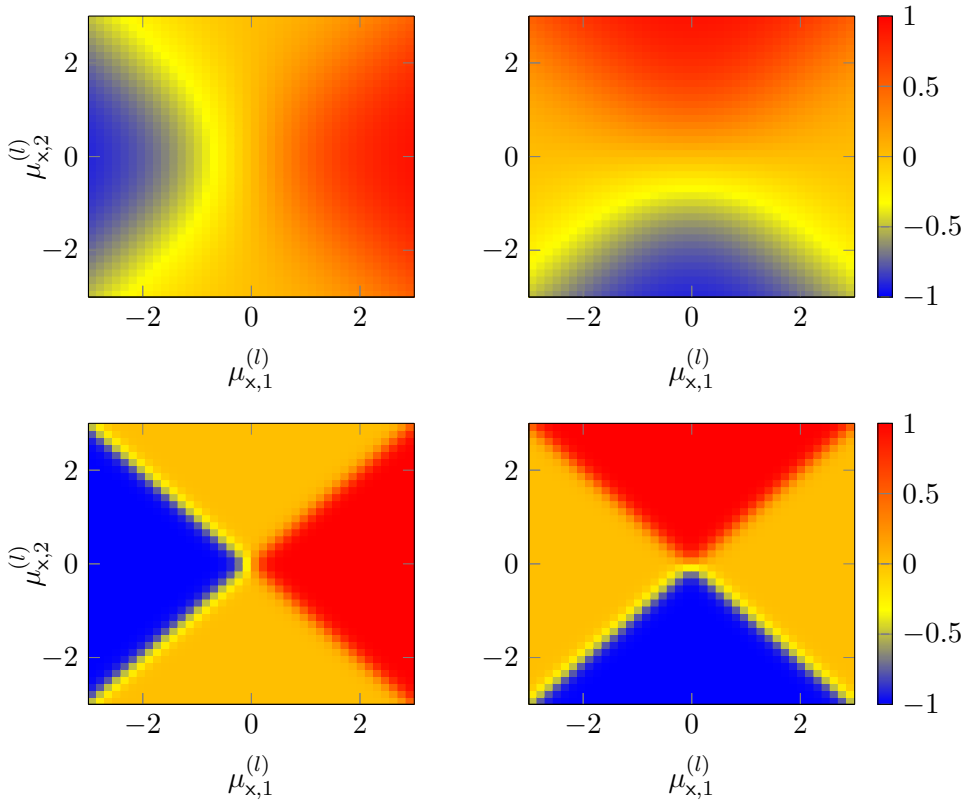


Figure 3.2: Function F (color coded) for the quaternary prior $f_{\mathbf{x},\mathrm{Qr}}(\boldsymbol{x})$. The left column shows the output for the first dimension, while the right column shows the result for the second dimension. Variances $\sigma_{\mathrm{x}}^{2(l)}$ are 1 (top row) and 0.1 (bottom row) for both dimensions.

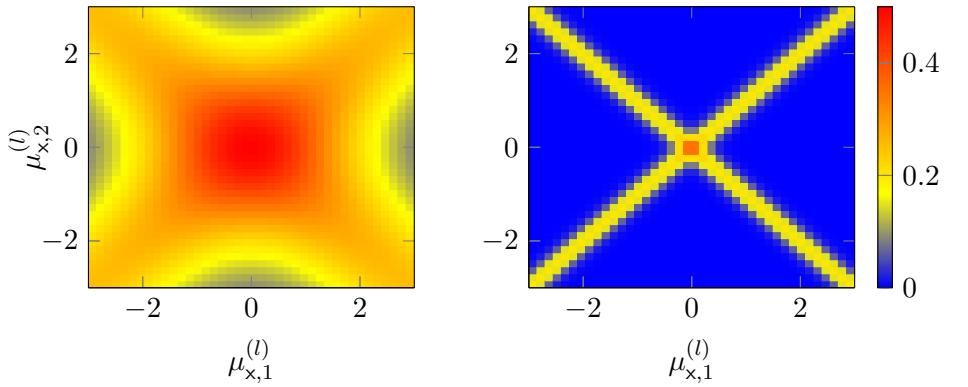


Figure 3.3: Function G (color coded) for the quaternary prior $f_{\mathbf{x},\mathrm{Qr}}(\boldsymbol{x})$. Depicted are the values for input variance $\sigma_{\mathrm{x}}^{2(l)} = 1$ (left) and $\sigma_{\mathrm{x}}^{2(l)} = 0.1$ (right). The input variance (and in this case, also the output) is identical for both dimensions.

3.4 Gaussian Distribution

The M-dimensional Gaussian distribution is given by

$$f_{\mathbf{x}}(\boldsymbol{x}) = \left((2\pi)^M \det(\boldsymbol{\Sigma}_p)\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_p)^T \boldsymbol{\Sigma}_p^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_p)\right), \tag{3.139}$$

where $\boldsymbol{\mu}_p$ is the prior mean and $\boldsymbol{\Sigma}_p$ is the prior covariance matrix. For a Gaussian prior, the function $\boldsymbol{I}_k$ computes the non-centered, non-normalized k^{th} moment of the product of two Gaussian distributions. This product is again a Gaussian function, however. Expressions for the parameters of this product can be found in [48]. The mean and covariance matrix of the resulting Gaussian function are

$$\boldsymbol{\Sigma} = \left(\boldsymbol{\Sigma}_p^{-1} + \boldsymbol{\Sigma}^{-1(l)}\right)^{-1} \tag{3.140}$$

$$\boldsymbol{\mu} = \boldsymbol{\Sigma}\left(\boldsymbol{\Sigma}_p^{-1}\boldsymbol{\mu}_p + \boldsymbol{\Sigma}^{-1(l)}\boldsymbol{\mu}^{(l)}\right). \tag{3.141}$$

The computationally intensive steps are the three matrix inversions. One of these is trivial since $\boldsymbol{\Sigma}^{(l)}$ is diagonal. Thus, the inversion only involves L floating-point divisions; for algorithms that track only a single variance, one floating-point division is required. These divisions need to be performed in each iteration, however. If the parameters of the Gaussian prior are constant, the inverse of its covariance matrix $\boldsymbol{\Sigma}_p$ can be precomputed. However, the inverse of the sum of inverses needs to be evaluated during each iteration of the algorithm. Using (3.140), (3.141), the functions F and G can be identified as

$$\boldsymbol{F} = \boldsymbol{\mu} \tag{3.142}$$

$$\boldsymbol{G} = \operatorname{diag}(\boldsymbol{\Sigma}). \tag{3.143}$$

The normalization term I_0 is

$$I_0 = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{\mu}_p - \boldsymbol{\mu}^{(l)})^T(\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}^{(l)})^{-1}(\boldsymbol{\mu}_p - \boldsymbol{\mu}^{(l)})\right)}{\left((2\pi)^M \det(\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}^{(l)})\right)^{\frac{1}{2}}}. \tag{3.144}$$

Thus, the function $\boldsymbol{I}_1$ can be expressed using (3.144) and (3.141):

$$\boldsymbol{I}_1 = \boldsymbol{\mu} I_0. \tag{3.145}$$

The function $\boldsymbol{I}_2$ is obtained as

$$\boldsymbol{I}_2 = \left(\operatorname{diag}(\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T)\right) I_0, \tag{3.146}$$

with $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ defined in (3.140) and (3.141) respectively. Since the inverse of sums (cf. (3.144)) as well as the inverse of the sum of inverses (cf. (3.140)) are required, the identity

$$(\boldsymbol{A}+\boldsymbol{B})^{-1} = \boldsymbol{A}^{-1}(\boldsymbol{A}^{-1}+\boldsymbol{B}^{-1})^{-1}\boldsymbol{B}^{-1} \tag{3.147}$$

can be used to avoid computation of the left-hand side at the cost of two extra matrix multiplications. Numerically, this optimization is sub-optimal. For the Gaussian distribution, only the derivatives of I_0 and $\boldsymbol{I}_1$ are discussed. Derivatives of $\boldsymbol{I}_k$ with $k > 1$ can be obtained by applying Isserli's theorem [34]. The derivative of I_0 can be written as

$$\frac{\partial I_0}{\partial \boldsymbol{\mu}^{(l)}} = \boldsymbol{D}(\boldsymbol{\mu}_p - \boldsymbol{\mu}^{(l)})I_0 \,, \tag{3.148}$$

where $\boldsymbol{D} = (\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}^{(l)})^{-1}$. The derivative of $\boldsymbol{I}_1$ is

$$\frac{\partial \boldsymbol{I}_1}{\partial \boldsymbol{\mu}^{(l)}} = \boldsymbol{\mu}\left(\frac{\partial I_0}{\partial \boldsymbol{\mu}^{(l)}}\right)^T + I_0\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1(l)} \,, \tag{3.149}$$

which is obtained by applying the product rule to (3.145). The expression (3.149) also contains the derivative of F, which is

$$\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\mu}^{(l)}} = \boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1(l)} = (\boldsymbol{\Sigma}_p^{-1} + \boldsymbol{\Sigma}^{-1(l)})^{-1}\boldsymbol{\Sigma}^{-1(l)} \tag{3.150}$$

$$= \boldsymbol{\Sigma}_p(\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}^{(l)})^{-1} \,, \tag{3.151}$$

where (3.147) was used. Note that (3.150) agrees with (3.29). Gaussian distributions are frequently employed in models of sparse signals, using the linear combination with a Dirac delta. A one-dimensional "Bernoulli-Gaussian" prior can be expressed as

$$f_{\mathsf{x},\mathrm{BG}}(x) = \frac{1}{2}\left(\delta(x) + \mathcal{N}_{\mathsf{x}}(\mu, \sigma^2)\right) . \tag{3.152}$$

The associated functions F and G are plotted in Fig. 3.4 while the functions I_0, I_1 and I_2 are shown in Fig. 3.5. Note that the denoiser F becomes similar to *both* the hard and soft thresholding functions as $\sigma_{\mathsf{x}}^{2(l)}$ becomes small. Since the Bayesian denoiser approaches identity (i.e. $x = F(\mu_{\mathsf{x}}^{(l)}, \sigma_{\mathsf{x}}^{2(l)}) = \mu_{\mathsf{x}}^{(l)}$) as $|\mu_{\mathsf{x}}^{(l)}|$ becomes large, the soft thresholder maintains an offset, however.

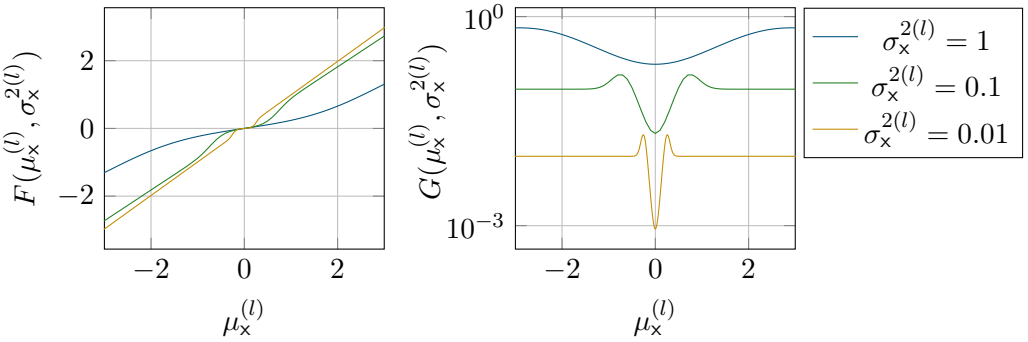


Figure 3.4: Functions F (left) and G (right) for the Bernoulli-Gaussian prior $f_{\mathrm{x,BG}}(x)$ and several local variances $\sigma_{\mathrm{x}}^{2(l)}$.

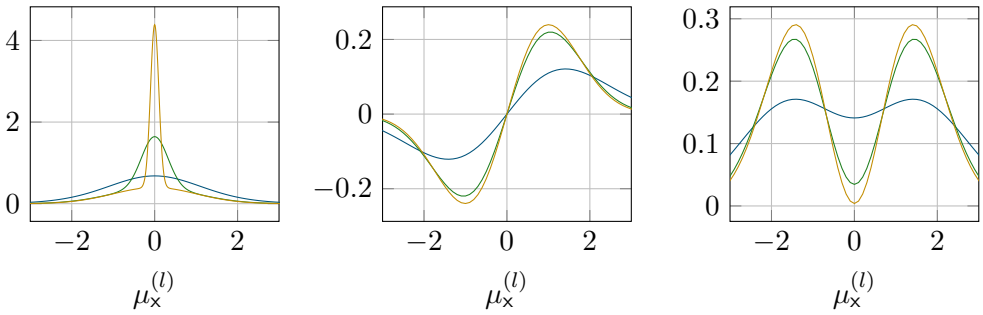


Figure 3.5: Functions I_0 (left), I_1 (center) and I_2 (right) for the Bernoulli-Gaussian prior $f_{\mathrm{x,BG}}(x)$ and several local variances $\sigma_{\mathrm{x}}^{2(l)}$.

3.5 Uniform Distribution

The uniform distribution is given by

$$f_{\mathrm{x}}(x) = \begin{cases} 0 & x < a \\ \frac{1}{b-a} & a \leq x \leq b \\ 0 & b < x \,, \end{cases} \tag{3.153}$$

with the boundaries $a, b \in \mathbb{R}$ where $a < b$. A multivariate version with vector-valued $\boldsymbol{a}, \boldsymbol{b}$ can be defined as product of one-dimensional uniform distributions. An M-dimensional uniform distribution can be thought of as a hyperrectangle. In this section, it is assumed that the distribution's dimensions are independent. They become dependent if the hyperrectangle is rotated and its edges no longer align with the M-dimensional Cartesian coordinate system. In these cases, the functions $\boldsymbol{I}_k$ involve a definite integral of an M-dimensional Gaussian, for which no closed-form solutions exist. A numerical approach based on belief propagation can be found in [20]. This section exclusively discusses multivariate uniform distributions

with independent dimensions. The normalizing factor I_0 is given as

$$I_0 = \frac{\operatorname{erf}(B) - \operatorname{erf}(A)}{2(b-a)}\,, \tag{3.154}$$

with A and B defined as

$$A = Q(a, \mu^{(l)}, \sigma^{2(l)}) \tag{3.155}$$

$$B = Q(b, \mu^{(l)}, \sigma^{2(l)}) \tag{3.156}$$

$$Q(q, \mu^{(l)}, \sigma^{2(l)}) = \frac{q - \mu^{(l)}}{\sqrt{2}\sigma^{(l)}}\,. \tag{3.157}$$

The functions I_1 and I_2 evaluate to

$$I_1 = \frac{1}{b-a}\left(\frac{\sigma^{(l)}}{\sqrt{2\pi}}\left(\exp(-A^2) - \exp(-B^2)\right)\right. \tag{3.158}$$

$$\left. + \frac{\mu^{(l)}}{2}\left(\operatorname{erf}(B) - \operatorname{erf}(A)\right)\right) \tag{3.159}$$

$$I_2 = \frac{1}{b-a}\left(\sqrt{2}\pi\sigma^{(l)}\mu^{(l)}\left(\exp(-A^2) - \exp(-B^2)\right)\right. \tag{3.160}$$

$$\frac{\sigma^{2(l)}}{\sqrt{\pi}}\left(A\exp(-A^2) - B\exp(-B^2)\right) \tag{3.161}$$

$$\left. \frac{\sigma^{2(l)} + \mu^{2(l)}}{2}\left(\operatorname{erf}(B) - \operatorname{erf}(A)\right)\right)\,. \tag{3.162}$$

To obtain the derivatives, the following identities are useful:

$$\frac{\partial \operatorname{erf}(x)}{\partial x} = \frac{2}{\sqrt{\pi}}\exp(-x^2) \tag{3.163}$$

$$\frac{\partial Q(\mu^{(l)}, \dots)}{\partial \mu^{(l)}} = -\frac{1}{\sqrt{2}\sigma^{(l)}} \tag{3.164}$$

$$\frac{\partial \operatorname{erf}\left(Q(\mu^{(l)}, \dots)\right)}{\partial \mu^{(l)}} = -\sqrt{\frac{2}{\pi\sigma^{2(l)}}}\exp(-Q^2) \tag{3.165}$$

$$\frac{\partial \exp\left(-Q^2(\mu^{(l)}, \dots)\right)}{\partial \mu^{(l)}} = \exp\left(-Q^2\right)\frac{q - \mu^{(l)}}{\sigma^{2(l)}}\,. \tag{3.166}$$

The derivatives are thus obtained as

$$\frac{\partial I_0}{\partial \mu^{(l)}} = \frac{\exp(-A^2) - \exp(-B^2)}{(b-a)\sqrt{2\pi}\sigma^{(l)}} \tag{3.167}$$

$$\frac{\partial I_1}{\partial \mu^{(l)}} = \frac{1}{b-a}\left(\frac{1}{\sqrt{2\pi}\sigma^{(l)}}\left(a\exp(-A^2) - b\exp(-B^2)\right)\right. \tag{3.168}$$

$$\frac{1}{2}\left(\operatorname{erf}(B)-\operatorname{erf}(A)\right)\Bigg). \tag{3.169}$$

Two graphical examples are given to showcase the operation of the denoiser for uniform priors. These are expressed by the pdfs

$$f_{\mathrm{x,U}}(x) = \mathcal{U}(-1,1) \tag{3.170}$$

$$f_{\mathbf{x},\mathrm{BU}}(\boldsymbol{x}) = \frac{1}{2}\delta(\boldsymbol{x}) + \frac{1}{4}\left(\mathcal{U}((-1,-1)^T,(0,0)^T) + \mathcal{U}((0,0)^T,(1,1)^T)\right), \tag{3.171}$$

where $f_{\mathrm{x,U}}(x)$ describes a one-dimensional uniform prior with bounds -1 and 1 while $f_{\mathbf{x},\mathrm{BU}}(\boldsymbol{x})$ is a two-dimensional mixture of two uniform distributions and a Dirac delta. The two-dimensional uniform "squares" have edge length 1 and lie southwest and northeast of the origin, where the Dirac delta is situated.
In Fig. 3.6, the functions F and G for $f_{\mathrm{x,U}}(x)$ are plotted. As $\sigma_{\mathrm{x}}^{2(l)}$ becomes small, the denoiser F approximates a straight line with slope 1 on the interval $[-1,1]$, while it saturates at -1 and 1 outside the range. For the two-dimensional prior $f_{\mathbf{x},\mathrm{BU}}(\boldsymbol{x})$, the variance is particularly large on the off-diagonal, e.g. for $\boldsymbol{\mu}_{\mathbf{x}}^{(l)} = (-1,1)^T$. This follows intuition since it is particularly "hard" to associate such a noisy value with one allowed by the prior. Close to the origin, the decision is particularly "easy" since with high probability, the noiseless vector was drawn from the Dirac delta, i.e. it is $(0,0)^T$.

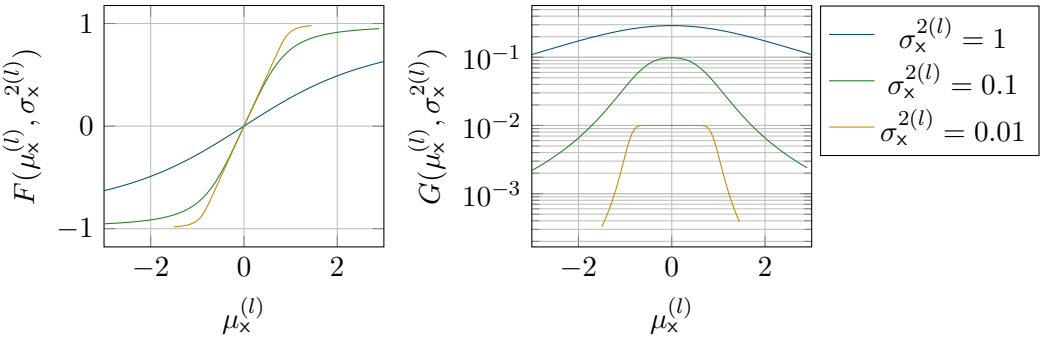


Figure 3.6: Functions F (left) and G (right) for the uniform prior $f_{\mathrm{x,U}}(x)$ and several local variances $\sigma_{\mathrm{x}}^{2(l)}$.

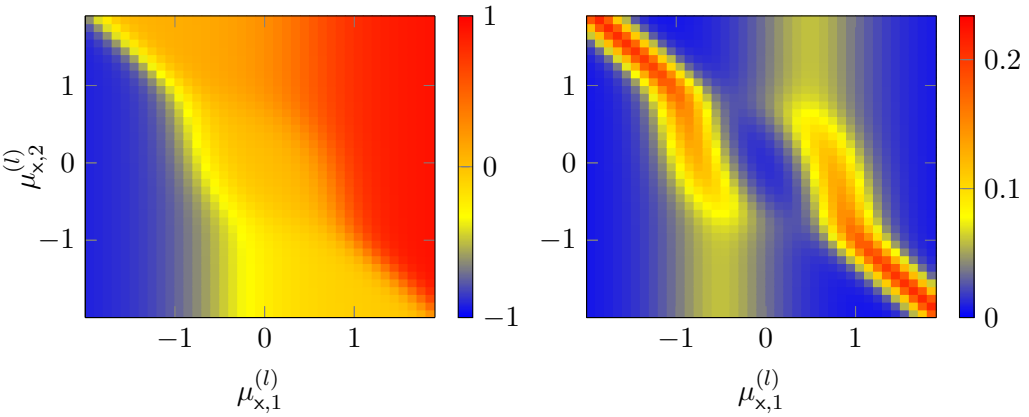


Figure 3.7: First dimension of functions F (left) and G (right) for the two-dimensional Bernoulli-Uniform prior $f_{\mathbf{x},\mathrm{BU}}(\boldsymbol{x})$. Depicted are the values (color coded) for input variance $\sigma_{\mathrm{x}}^{2(l)} = 0.1$.

3.6 Exponential Distribution

The exponential distribution is given by

$$f_{\mathrm{x}}(x) = \begin{cases} \lambda \exp\left(-\lambda x\right) & x \geq 0 \\ 0 & x < 0\,, \end{cases} \tag{3.172}$$

where λ is the rate of a Poisson point process. It is useful to define the shorthand expressions

$$A = \frac{\lambda\sigma^{2(l)} - \mu^{(l)}}{\sqrt{2\sigma^{2(l)}}} \tag{3.173}$$

$$B = \frac{\lambda}{2}(\lambda\sigma^{2(l)} - 2\mu^{(l)})\,. \tag{3.174}$$

The following expressions of I_k have been formulated to provide numerical stability:

$$I_0 = \frac{\lambda}{2}\exp(B)\operatorname{erfc}(A) \tag{3.175}$$

$$I_1 = \frac{\lambda\sigma^{(l)}}{\sqrt{2\pi}}\exp\left(-\frac{\mu^{2(l)}}{2\sigma^{2(l)}}\right) - \left(B + \frac{\lambda\mu^{(l)}}{2}\right)\exp(B)\operatorname{erfc}(A) \tag{3.176}$$

$$I_2 = \frac{\lambda}{2}\left((\lambda\sigma^{2(l)} - \mu^{(l)})^2 + \sigma^{2(l)}\right)\exp(B)\operatorname{erfc}(A) \tag{3.177}$$

$$+ \frac{\lambda}{\sqrt{2\pi}}\exp\left(-\frac{\mu^{2(l)}}{2\sigma^{2(l)}}\right)\left(\sigma^{(l)}\mu^{(l)} - \lambda\sigma^{3(l)}\right) \tag{3.178}$$

$$\frac{\partial I_0}{\partial \mu^{(l)}} = \frac{\lambda}{\sqrt{2\pi}\sigma^{(l)}}\exp\left(-\frac{\mu^{2(l)}}{2\sigma^{2(l)}}\right) - \frac{\lambda^2}{2}\exp(B)\operatorname{erfc}(A) \tag{3.179}$$

$$\frac{\partial I_1}{\partial \mu^{(l)}} = \frac{\lambda}{2}(\lambda^2\sigma^{2(l)} - \lambda\mu^{(l)} + 1)\exp(B)\operatorname{erfc}(A) - \frac{\lambda^2\sigma^{(l)}}{\sqrt{2\pi}}\exp\left(-\frac{\mu^{2(l)}}{2\sigma^{2(l)}}\right). \quad (3.180)$$

Simple expressions for F and F' are

$$F = \mu^{(l)} - \lambda\sigma^{2(l)} + \frac{\exp(-A^2)\sqrt{\frac{2\sigma^{2(l)}}{\pi}}}{\operatorname{erfc}(A)} \quad (3.181)$$

$$\frac{\partial F}{\partial \mu^{(l)}} = 1 - \frac{\exp(-A^2)\left(2\sigma^{(l)}\exp(-A^2) - \sqrt{2\pi}(\lambda\sigma^{2(l)} - \mu^{(l)})\operatorname{erfc}(A)\right)}{\pi\sigma^{(l)}(\operatorname{erfc}(A))^2}. \quad (3.182)$$

For illustration purposes, the functions for the exponential prior with $\lambda = 1$ are plotted in Fig. 3.8. Since the prior is one-sided (i.e. $f(x)|_{x<0} = 0$), the function $F \geq 0 \quad \forall \mu_{\mathsf{x}}^{(l)} \in \mathbb{R}$.

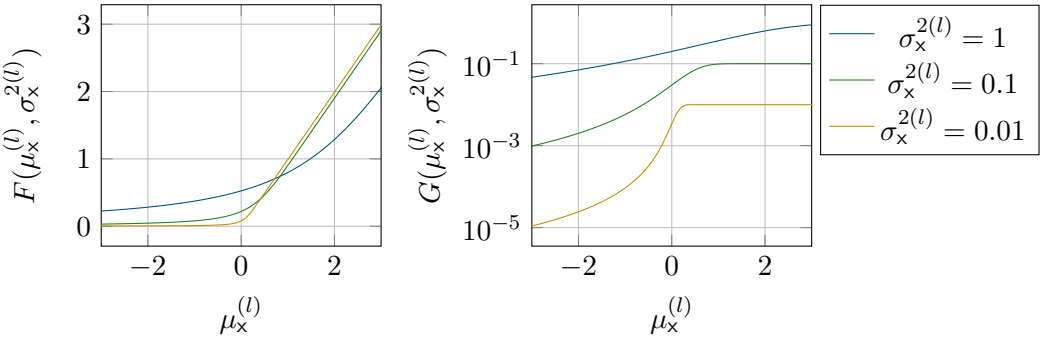


Figure 3.8: Functions F (left) and G (right) for the exponential prior with $\lambda = 1$ and various local variances $\sigma_{\mathsf{x}}^{2(l)}$.

3.7 Chi-Squared Distribution

The Chi-squared distribution is given by

$$f_{\mathsf{x}}(x) = \begin{cases} \frac{x^{\frac{k}{2}-1}e^{-\frac{x}{2}}}{2^{\frac{k}{2}}\Gamma\left(\frac{k}{2}\right)} & x \geq 0 \quad (k = 1 : x > 0) \\ 0 & x < 0 \quad (k = 1 : x \leq 0), \end{cases} \quad (3.183)$$

where k are the "degrees of freedom" and $\Gamma(z)$ is the Gamma function. If $\mathsf{z_k}$ are distributed according to the standard normal distribution, then $\mathsf{x} = \sum_k \mathsf{z}_k^2$ is chi-squared with k degrees of freedom. The Gamma function is defined as

$$\Gamma(z) = \int_0^\infty x^{z-1}e^{-x}\mathrm{d}x. \quad (3.184)$$

For $n \in \mathbb{N}$, $\Gamma(n) = (n-1)!$. It is useful to define the shorthand expressions

$$A = \frac{(\sigma^{2(l)} - 2\mu^{(l)})^2}{8\sigma^{2(l)}} \tag{3.185}$$

$$B = \exp\left(-\frac{\mu^{2(l)}}{2\sigma^{2(l)}}\right) \tag{3.186}$$

$$k_n = \frac{n+k}{4}\,. \tag{3.187}$$

Using the confluent hypergeometric function ${}_1\mathrm{F}_1(a,b,z)$ (for a detailled discussion, see e.g. [2, Chapter 2]), expressions for I_0, I_1 and I_2 can be written as

$$I_0 = 2^{-\frac{4+3k}{4}}\sigma^{-\frac{4-k}{2}(l)}B\left[\frac{\sqrt{2}\sigma^{(l)}}{\Gamma(k_2)}\,{}_1\mathrm{F}_1\left(k_0,\frac{1}{2},A\right)\right. \tag{3.188}$$

$$\left. -\frac{\sigma^{2(l)} - 2\mu^{(l)}}{\Gamma(k_0)}\,{}_1\mathrm{F}_1\left(k_2,\frac{3}{2},A\right)\right] \tag{3.189}$$

$$I_1 = \frac{2^{-k_6}\sigma^{-\frac{2-k}{2}}B}{\sqrt{\pi}\Gamma\left(\frac{k}{2}\right)}\left[\sqrt{2}\sigma^{(l)}\Gamma(k_2)\,{}_1\mathrm{F}_1\left(k_2,\frac{1}{2},A\right)\right. \tag{3.190}$$

$$\left. -\left(\sigma^{2(l)} - 2\mu^{(l)}\right)\Gamma(k_4)\,{}_1\mathrm{F}_1\left(k_4,\frac{3}{2},A\right)\right] \tag{3.191}$$

$$I_2 = \frac{2^{-k_6}\sigma^{\frac{k}{2}(l)}B}{\sqrt{\pi}\Gamma\left(\frac{k}{2}\right)}\left[2\sigma^{(l)}\Gamma(k_4)\,{}_1\mathrm{F}_1\left(k_4,\frac{1}{2},A\right)\right. \tag{3.192}$$

$$\left. -\sqrt{2}\left(\sigma^{2(l)} - 2\mu^{(l)}\right)\Gamma(k_6)\,{}_1\mathrm{F}_1\left(k_6,\frac{3}{2},A\right)\right]. \tag{3.193}$$

The derivatives are

$$\frac{\partial I_0}{\partial \mu^{(l)}} = 2^{-3k_4}\sigma^{-\frac{8-k}{2}}B\left\{-\frac{\sqrt{2}\sigma^{(l)}}{\Gamma(k_2)}\left[4\mu^{(l)}\,{}_1\mathrm{F}_1\left(k_0,\frac{1}{2},A\right)\right.\right. \tag{3.194}$$

$$\left. +k(\sigma^{2(l)} - 2\mu^{(l)})\,{}_1\mathrm{F}_1\left(k_4,\frac{3}{2},A\right)\right] \tag{3.195}$$

$$+\frac{1}{3\Gamma(k_0)}\left[12\left(\sigma^{2(l)}(2+\mu^{(l)}) - 2\mu^{2(l)}\right)\,{}_1\mathrm{F}_1\left(k_2,\frac{3}{2},A\right)\right. \tag{3.196}$$

$$\left.\left. +\left(\sigma^{2(l)} - 2\mu^{(l)}\right)^2(2+k)\,{}_1\mathrm{F}_1\left(k_6,\frac{5}{2},A\right)\right]\right\} \tag{3.197}$$

$$\frac{\partial I_1}{\partial \mu^{(l)}} = \frac{2^{-k_{14}}\sigma^{-\frac{6-k}{2}(l)}B}{3\sqrt{\pi}\Gamma\left(\frac{k}{2}\right)}\left\{-3\sqrt{2}\sigma^{(l)}\Gamma(k_2)\left[4\mu^{(l)}\,{}_1\mathrm{F}_1\left(k_2,\frac{1}{2},A\right)\right.\right. \tag{3.198}$$

$$\left. +\left(\sigma^{2(l)} - 2\mu^{(l)}\right)(2+k)\,{}_1\mathrm{F}_1\left(k_6,\frac{3}{2},A\right)\right] \tag{3.199}$$

$$+\Gamma(k_4)\left[12\left(\sigma^{2(l)}(2+\mu^{(l)}) - 2\mu^{2(l)}\right)\,{}_1\mathrm{F}_1\left(k_4,\frac{3}{2},A\right)\right. \tag{3.200}$$

$$+\left(\sigma^{2(l)}-2\mu^{(l)}\right)^2(4+k)\,{}_1\mathrm{F}_1\left(k_8,\frac{5}{2},A\right)\Bigg]\Bigg\}. \tag{3.201}$$

The confluent hypergeometric function ${}_1\mathrm{F}_1(a,b,z)$ grows with z at a rate faster than the exponential function. The expressions above are thus unsuitable for implementation. The confluent hypergeometric function can however be written as

$${}_1\mathrm{F}_1(a,b,z)=e^z\,{}_1\mathrm{F}_1(b-a,b,-z)\,. \tag{3.202}$$

The argument $z=A$ and all expressions contain the factor B, resulting in terms of the form

$$Be^A=\exp\left(-\frac{\mu^{2(l)}}{2\sigma^{2(l)}}\right)\exp\left(\frac{(\sigma^{2(l)}-2\mu^{(l)})^2}{8\sigma^2(l)}\right) \tag{3.203}$$

$$=\exp\left(\frac{\sigma^{2(l)}(\sigma^{2(l)}-4\mu^{(l)})}{8\sigma^{2(l)}}\right) \tag{3.204}$$

$$=\exp\left(\frac{\sigma^{2(l)}-4\mu^{(l)}}{8}\right), \tag{3.205}$$

which can be used to obtain expressions more suitable for numerical evaluation. Critical implementations should consider piece-wise approximations, however. Like the exponential distribution, the chi-squared distribution is one-sided. Their denoiser functions are similar. In Fig. 3.9, the functions F and G for a chi-squared distribution with one degree of freedom is plotted.

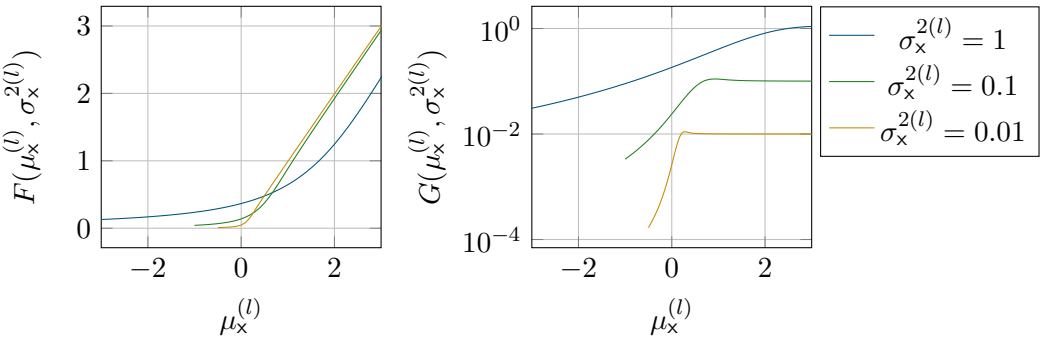


Figure 3.9: Functions F (left) and G (right) for the chi-squared prior with $k=1$ degrees of freedom and various local variances $\sigma_x^{2(l)}$.

Chapter 4

Compressed LDPC Codes

Low density parity check (LDPC) codes are popular error correction codes. They were proposed by Robert G. Gallager [31] in 1962 and rediscovered in the 1990s [54, 42, 41]. Today, they can be found in applications ranging from wired [33] and wireless [32] local area networks to digital video broadcasting [28], storage technologies [69] and also in the 5^{th} generation mobile radio access network [29].

Motivation and related work. Recovery algorithms derived in the context of compressed sensing have been used as detectors in transmission systems before. The applications range from multi-user detection [63, 36, 37, 8] to detectors of capacity-achieving codes [52]. Prior work on compressed LDPC codes was presented in [6], where we constructed the detector by simply combining the LDPC sum-product detector and BAMP, which yielded an algorithm that is similar to the one obtained here. The analysis here provides greater detail and better explains the behavior and limitations of MD-BAMP as detector for compressed LDPC codes. Compressing code words yields a higher rate, i.e. the fraction of bits per symbol is larger. The idea is to be able to dynamically adjust the rate for a given channel quality using a single high-performance code. This goal is achieved by the system presented in this chapter: the code rate can easily be adjusted by choosing the size of the matrix $\boldsymbol{A}$. Another motivation was the approximately Gaussian distribution of transmit symbols $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, which should provide a shaping gain. This gain is not realized by the system presented here. Possible reasons and inspiration for future work is given in Section 5.6 and Chapter 6.

Notation. Some additional notation is required for this chapter. A code word $\boldsymbol{c}$ of length N_c represents ("encodes") a vector $\boldsymbol{u}$ of K_u user data symbols (or "information symbols"). The vector $\boldsymbol{u} \in \{\mathrm{GF}(q)\}^{K_u}$ while $\boldsymbol{c} \in \{\mathrm{GF}(q)\}^{N_c}$, where $q = p^m$, $m \in \mathbb{N} \setminus \{0\}$ and p prime. The algorithms in this chapter are concerned with binary LDPC codes and thus, $p = q = 2$. There are 2^{K_u} different user data words $\boldsymbol{u}$ and thus equally many code words $\boldsymbol{c}$ in a code $\mathcal{C}$. The code $\mathcal{C}$ is the set of all code words $\boldsymbol{c}$ and its size is written as $|\mathcal{C}| = 2^{K_u}$. There exists a bijective mapping

between user data words and code words. This is only possible if $N_c \geq K_u$. All user data words and thus also all code words are assumed to be equally likely to occur. An LDPC code is a linear block code. It is therefore possible to find a generator matrix $\boldsymbol{G} \in \{\mathrm{GF}(q)\}^{N_c \times K_u}$ and a check matrix $\boldsymbol{H} \in \{\mathrm{GF}(q)\}^{(N_c-K_u)\times N_c}$. Note that these matrices are not unique. It can be shown that $\forall \boldsymbol{c} \in \mathcal{C} : \boldsymbol{Hc} = \boldsymbol{0}$. Furthermore, $\boldsymbol{HG} = \boldsymbol{0}$. LDPC codes obtain their name from the fact that there exists a representation of the check matrix $\boldsymbol{H}$ which is sparse. This particular form of the check matrix may have more than $N_c - K_u$ rows. The number of nonzero entries in the a^{th} row of $\boldsymbol{H}$ is called the row weight $\rho_{\boldsymbol{H},a}$. The equivalent for the i^{th} column of $\boldsymbol{H}$ is called the column weight $\gamma_{\boldsymbol{H},i}$. It is possible to distinguish between regular and irregular LDPC codes. For the former, the row weight and column weight is identical for all rows and columns respectively. The latter allows for variation of the row and column weights. In this chapter, regular LDPC codes are assumed. It is possible to extend the presented techniques to irregular LDPC codes, however.

Before transmission, it is necessary to map the code words $\boldsymbol{c}$ to transmit symbol vectors $\boldsymbol{v}$. Depending on the choice of transmission system, the transmit symbol vector $\boldsymbol{v} \in \mathbb{R}^{N_t}$ or $\boldsymbol{v} \in \mathbb{C}^{N_t}$. It is possible to use symbol-wise mappings, map groups of symbols or the whole code word to a transmit symbol. The choice of this mapping determines N_t. In this chapter, a general zero-mean mapping from $\mathrm{GF}(2) \mapsto \mathbb{R}$

$$0 \mapsto t \tag{4.1}$$

$$1 \mapsto -t \tag{4.2}$$

is used and thus $N_c = N_t$. The value t may depend on the position of the code word symbol. In this chapter, it is assumed that there is an additional compression step before transmission. The vectors $\boldsymbol{v}$ are thus referred to as the "inner transmit symbol vectors", while the vector $\boldsymbol{y}^{(tx)}$ obtained after the compression step is called the "outer transmit symbol vector". Before compression, a number K of inner transmit vectors $\boldsymbol{v}$ are concatenated:

$$\boldsymbol{x} = (\boldsymbol{v}_1^T, \boldsymbol{v}_2^T, \ldots, \boldsymbol{v}_K^T)^T \,. \tag{4.3}$$

Thus, a long inner transmit vector $\boldsymbol{x}$ of length $N = KN_t$ is obtained. This vector $\boldsymbol{x}$ can be regarded as a code word of a product code mapped to the transmit symbol set. The outer transmit symbol vector $\boldsymbol{y}^{(tx)} \in \mathbb{R}^L$ is then obtained by compressing $\boldsymbol{x}$:

$$\boldsymbol{y}^{(tx)} = \boldsymbol{A}\boldsymbol{x} \,. \tag{4.4}$$

The received symbol vector $\boldsymbol{y}$ shall be affected by additive white Gaussian noise:

$$\boldsymbol{y} = \boldsymbol{y}^{(tx)} + \boldsymbol{w} \,. \tag{4.5}$$

A block diagram of the transmission system can be seen in Fig. 4.1.

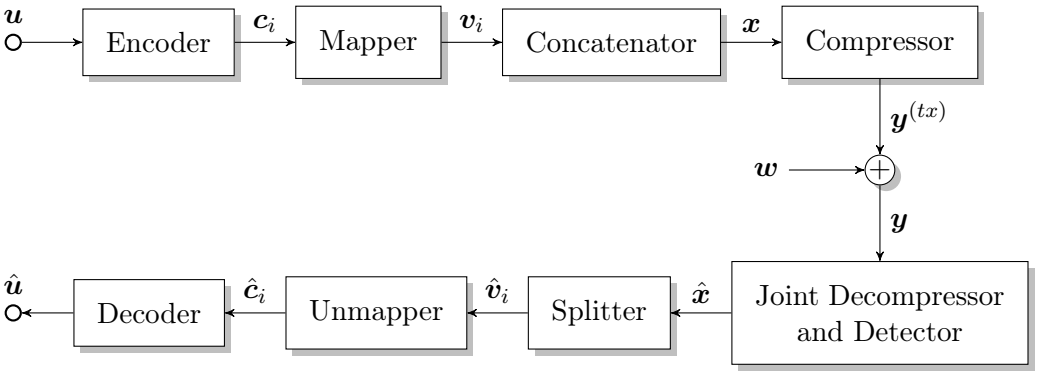


Figure 4.1: Model describing transmission system using a compressed LDPC channel code.

4.1 Denoiser Construction

To obtain expressions for the functions F, G and F' using the formalism outlined in Section 3.1, it is necessary to know the factors $f_{\mathbf{x}_k}(\boldsymbol{x}_k)$ of the prior $f_{\mathbf{x}}(\boldsymbol{x})$. These have dimension N_c and are determined by the code and the mapper. The prior factor is thus constructed by a linear combination of multivariate, equally weighted Dirac deltas. Each code word $\boldsymbol{c}_j \in \mathcal{C}$ can be associated with a Dirac delta at $\delta(\boldsymbol{x} - \boldsymbol{s}_j)$. The prior is thus given by

$$f_{\mathbf{x}}^{\mathrm{LDPC}}(\boldsymbol{x}) = 2^{-K_u} \sum_{\boldsymbol{s}_j \in \mathcal{C}^{\mathbb{R}}} \delta(\boldsymbol{x} - \boldsymbol{s}_j)\,, \tag{4.6}$$

where $\boldsymbol{s}_j$ is the j^{th} code word from $\mathcal{C}$ mapped to $\mathbb{R}$. The number of terms in the sum (4.6) grows exponentially with the length K_u of the user data vector. A straightforward implementation of the denoiser following Section 3.1 for any reasonably sized code will be unusable due to limited computational resources. It is thus desirable to exploit the code's structure.
For a binary LDPC code, each row in the check matrix $\boldsymbol{H}$ can be regarded as a parity check equation. Any code word $\boldsymbol{c} \in \mathcal{C}$ satisfies

$$\boldsymbol{h}_a^T \boldsymbol{c} = 0\,, \tag{4.7}$$

where $\boldsymbol{h}_a$ is the column vector representing the a^{th} row of the check matrix $\boldsymbol{H}$. The code word symbols identified by the non-zero entries of $\boldsymbol{h}_a$ thus form an even parity check code. The popular sum-product algorithm (SPA) detector [41] for LDPC codes represents every row of $\boldsymbol{H}$ by a single parity check factor node. A graphical description of this relationship can be seen in Fig. 4.2. In the context of compressive sensing, each row of $\boldsymbol{H}$ identifies a sub-vector of symbols with a common prior factor, expressed as a single parity check code of length $\rho_{\boldsymbol{H}}$ (i.e. the row weight). Note that each code word symbol depends on exactly $\gamma_{\boldsymbol{H}}$ single

parity check factors. These prior factors are "compatible" since their marginal w.r.t. a particular symbol x_i is always

$$f_{\mathsf{x}_i}(x_i) = \frac{1}{2}\left(\delta(x_i + t_i) + \delta(x_i - t_i)\right) , \tag{4.8}$$

where $t_i \in \mathbb{R} \setminus \{0\}$ is the inner transmit symbol used by the mapper for the code word symbol at position i. Since each code word symbol appears in multiple prior factors, these factors are not disjoint. A graphical model of this setting can be seen in Fig. 4.3. For each prior factor, a denoiser function F computes the estimated value $\mu_{\mathsf{x},i}$ of dependent variables x_i. Similarly, a function G computes the variance of the estimate for each prior factor. Since each variable depends on multiple prior factors (i.e. parity check functions), it is necessary to achieve a consensus among the outputs of the functions F and G pertaining to a particular variable.

It is possible to regard the outputs of the $\gamma_{\boldsymbol{H}}$ different functions F, G w.r.t. a particular symbol x_i as separate variables constrained by a binary repetition code. It will be shown that the output of the functions F, G obtained from a prior representing a binary repetition code are identical for each dimension; they are thus suitable to obtain a consensus.

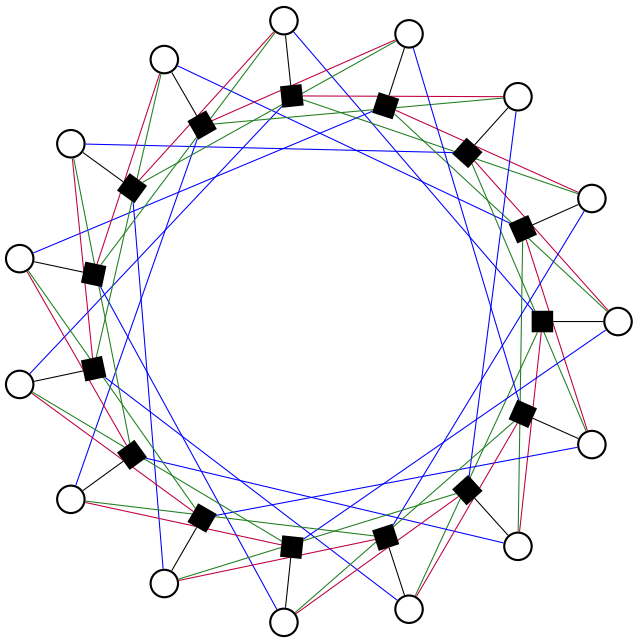


Figure 4.2: Depiction of the relationship between parity check constraints (black squares) and code word symbols (circles) of a cyclic LDPC code with $N_c = 15$ and $K_u = 7$.

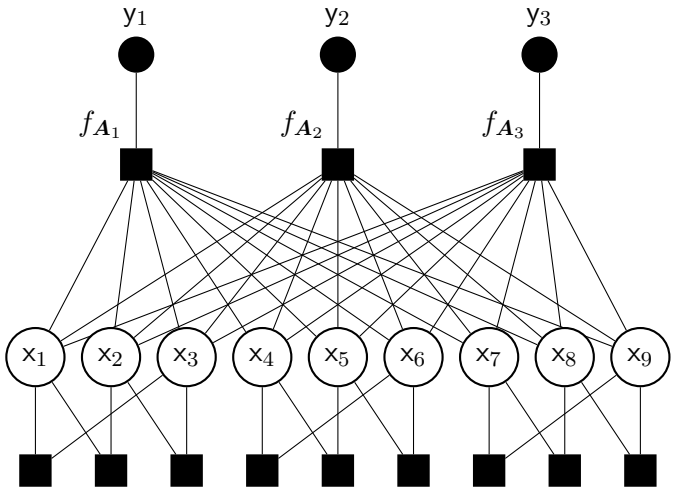


Figure 4.3: Graphical model of compressed LDPC code words with code word length $N_c = 3$ and user data vector length $K_u = 1$. A total of $K = 3$ code words are concatenated to form an inner transmit symbol vector of length $N = 9$. The outer transmit symbol vector consists of $L = 3$ symbols y_a.

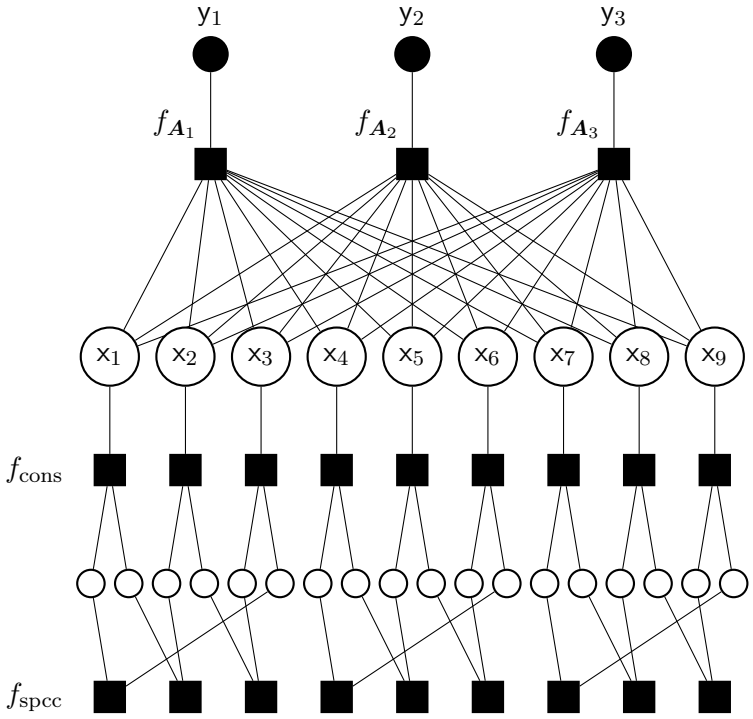


Figure 4.4: This graphic extends Fig. 4.3 by adding consensus factor nodes f_{cons} and auxiliary variables for the parity check nodes f_{spcc}.

4.2 Single Parity Check Code

A single parity check code (SPCC) can be obtained by appending a single parity symbol to a vector of K_u user data symbols. Thus, the length of a code word $\boldsymbol{c}$ is $N_c = K_u + 1$. The parity symbol is chosen such that

$$\mathbf{1}_{N_c}^T \boldsymbol{c} = 0\,. \tag{4.9}$$

Such a code is called an even parity check code. If the SPCC is defined over GF(2), a second practically important case exists. Let the parity symbol be chosen such that

$$\mathbf{1}_{N_c}^T \boldsymbol{c} = 1\,, \tag{4.10}$$

then the code is an odd parity check code. Note that, while the even parity check code is a linear code, the odd parity check code is not (the all-zero code word $\boldsymbol{c} = \mathbf{0}$ is not in $\mathcal{C}$). Let the i^{th} code word symbol c_i of a binary SPCC be mapped to a transmit symbol $\pm t_i \in \mathbb{R}$. Without loss of generality the concrete mapping from GF(2) $\to \mathbb{R}$ shall be $0 \to t_i$, $1 \to -t_i$. Then, a prior $f_{\mathbf{x}}(\boldsymbol{x})$ describing the SPCC in the transmit symbol domain can be obtained as

$$f_{\mathbf{x}}^{\text{SPCC}}(\boldsymbol{x}) = 2^{-K_u} \sum_{\boldsymbol{s}_a \in \mathcal{C}^{\mathbb{R}}} \delta(\boldsymbol{x} - \boldsymbol{s}_a)\,, \tag{4.11}$$

where $\boldsymbol{s}_a$ is the a^{th} code word in the single parity check code $\mathcal{C}$ mapped to $\mathbb{R}$, i.e. the i^{th} entry of $\boldsymbol{s}_a$ is $s_{a,i} \in \{+t_i, -t_i\}$. Using the expressions (3.131) - (3.133) and applying the composition rules from Section 3.1, the function F_i is obtained as

$$F_i = \frac{\sum_a I_{1,i}^{(a)}}{\sum_a I_0^{(a)}} = \frac{\sum_a s_{a,i} I_0^{(a)}}{\sum_a I_0^{(a)}}\,. \tag{4.12}$$

In (4.12), the functions $I_{k,i}^{(a)}$ correspond to the a^{th} element of the sum (4.11). The function I_0 is given by (3.131). The normalization term $((2\pi)^{N_c} \det(\boldsymbol{\Sigma}^{(l)}))^{-\frac{1}{2}}$ appears in all terms of the sums in (4.12) and thus cancels out. Such a modified function $I_0^{(a)\text{mod}}$ can be written as

$$I_0^{(a)\text{mod}} = \exp\left(-\frac{1}{2}\sum_j \frac{(s_{a,j} - \mu_j^{(l)})^2}{\sigma_j^{2(l)}}\right) \tag{4.13}$$

$$= \exp\left(-\frac{1}{2}\sum_j \frac{1}{\sigma_j^{2(l)}}\left(s_{a,j}^2 - 2 s_{a,j}\mu_j^{(l)} + \mu_j^{2(l)}\right)\right) \tag{4.14}$$

$$= \exp\left(-\frac{1}{2}\sum_j \frac{s_{a,j}^2 + \mu_j^{2(l)}}{\sigma_j^{2(l)}}\right) \exp\left(-\frac{1}{2}\sum_j \frac{-2 s_{a,j}\mu_j^{(l)}}{\sigma_j^{2(l)}}\right)\,. \tag{4.15}$$

Note that the left-hand factor in (4.15) appears in all elements of the sums in (4.12). This is due to $s_{a,j}^2$ being independent of the code word: squaring the j^{th} symbol of any code word always results in t_j^2. The modified function $I_0^{(a)\text{mod}}$ can thus be written as

$$I_0^{(a)\text{mod}} = \exp\left(\sum_j \frac{s_{a,j}\mu_j^{(l)}}{\sigma_j^{2(l)}}\right). \tag{4.16}$$

It shall prove useful to define

$$\alpha_j = \frac{t_j \mu_j^{(l)}}{\sigma_j^{2(l)}}. \tag{4.17}$$

It is then possible to write (4.16) as

$$I_0^{(a)\text{mod}} = \exp\left(\sum_j \alpha_{a,j}\right) = \prod_j e^{\alpha_{a,j}}, \tag{4.18}$$

where $\alpha_{a,j} = \pm\alpha_j$ depending on the sign of $s_{a,j}$. Inserting (4.18) into (4.12) results in

$$F_i = \frac{\sum_a s_{a,i} \prod_j e^{\alpha_{a,j}}}{\sum_a \prod_j e^{\alpha_{a,j}}}. \tag{4.19}$$

Since $s_{a,i}$ can only take on two different values, the sums can be partitioned depending on its value:

$$F_i = t_i \frac{\sum_{a:s_{a,i}=t_i} \prod_j e^{\alpha_{a,j}} - \sum_{a:s_{a,i}=-t_i} \prod_j e^{\alpha_{a,j}}}{\sum_{a:s_{a,i}=t_i} \prod_j e^{\alpha_{a,j}} + \sum_{a:s_{a,i}=-t_i} \prod_j e^{\alpha_{a,j}}} \tag{4.20}$$

$$= t_i \frac{e^{\alpha_i} \sum_{a:s_{a,i}=t_i} \prod_{j\neq i} e^{\alpha_{a,j}} - e^{-\alpha_i} \sum_{a:s_{a,i}=-t_i} \prod_{j\neq i} e^{\alpha_{a,j}}}{e^{\alpha_i} \sum_{a:s_{a,i}=t_i} \prod_{j\neq i} e^{\alpha_{a,j}} + e^{-\alpha_i} \sum_{a:s_{a,i}=-t_i} \prod_{j\neq i} e^{\alpha_{a,j}}}. \tag{4.21}$$

In (4.21), the reformulation was obtained by making use of the fact that all factors e^{α_i} are identical within a particular sum. Note that (4.21) has the form

$$F_i = t_i \frac{e^{\alpha_i} X - e^{-\alpha_i} Y}{e^{\alpha_i} X + e^{-\alpha_i} Y} \tag{4.22}$$

$$X = \sum_{a:s_{a,i}=t_i} \prod_{j\neq i} e^{\alpha_{a,j}} \tag{4.23}$$

$$Y = \sum_{a:s_{a,i}=-t_i} \prod_{j\neq i} e^{\alpha_{a,j}}. \tag{4.24}$$

The following reformulation yields a more suitable expression:

$$F_i = t_i \frac{e^{\alpha_i} X - e^{-\alpha_i} Y}{e^{\alpha_i} X + e^{-\alpha_i} Y} \tag{4.25}$$

$$= t_i \frac{(e^{\alpha_i} - e^{-\alpha_i})(X+Y) + (e^{\alpha_i} + e^{-\alpha_i})(X-Y)}{(e^{\alpha_i} + e^{-\alpha_i})(X+Y) + (e^{\alpha_i} - e^{-\alpha_i})(X-Y)} \tag{4.26}$$

$$= t_i \frac{e^{\alpha_i}X - e^{-\alpha_i}X + e^{\alpha_i}Y - e^{-\alpha_i}Y + e^{\alpha_i}X + e^{-\alpha_i}X - e^{\alpha_i}Y - e^{-\alpha_i}Y}{e^{\alpha_i}X + e^{-\alpha_i}X + e^{\alpha_i}Y + e^{-\alpha_i}Y + e^{\alpha_i}X - e^{-\alpha_i}X - e^{\alpha_i}Y + e^{-\alpha_i}Y} \tag{4.27}$$

$$= t_i \frac{2(e^{\alpha_i}X - e^{-\alpha_i}Y)}{2(e^{\alpha_i}X + e^{-\alpha_i}Y)} \tag{4.28}$$

$$= t_i \frac{\frac{e^{\alpha_i} - e^{-\alpha_i}}{e^{\alpha_i} + e^{-\alpha_i}} + \frac{X-Y}{X+Y}}{1 + \frac{e^{\alpha_i} - e^{-\alpha_i}}{e^{\alpha_i} + e^{-\alpha_i}} \frac{X-Y}{X+Y}} \tag{4.29}$$

$$= t_i \frac{\tanh(\alpha_i) + \frac{X-Y}{X+Y}}{1 + \tanh(\alpha_i)\frac{X-Y}{X+Y}} \,. \tag{4.30}$$

Note that the derivation did not make use of the structure of the code up to now. If the prior describes a single parity check code, the variables X and Y describe single parity check codes of length $N_c - 1$ and thus their prior is composed of $2^{K_u - 1}$ Dirac deltas of dimension $N_c - 1$. One of these codes will have odd, the other one even parity. For the particular symbol mapping chosen above, the codes described by X and Y have even and odd parity respectively. Nonetheless it is still possible to split these codes into smaller single parity check codes using a similar procedure as applied before, i.e. separating the code words based on the value of one particular coordinate (symbol). Any symbol can be chosen:

$$X = e^{\alpha_k} \sum_{\substack{a: s_{a,k} = t_k \\ \wedge s_{a,i} = t_i}} \prod_{j \neq k,i} e^{\alpha_{a,j}} + e^{-\alpha_k} \sum_{\substack{a: s_{a,k} = -t_k \\ \wedge s_{a,i} = t_i}} \prod_{j \neq k,i} e^{\alpha_{a,j}} \tag{4.31}$$

$$= e^{\alpha_k} U + e^{-\alpha_k} V \tag{4.32}$$

$$Y = e^{\alpha_k} \sum_{\substack{a: s_{a,k} = t_k \\ \wedge s_{a,i} = -t_i}} \prod_{j \neq k,i} e^{\alpha_{a,j}} + e^{-\alpha_k} \sum_{\substack{a: s_{a,k} = -t_k \\ \wedge s_{a,i} = -t_i}} \prod_{j \neq k,i} e^{\alpha_{a,j}} \tag{4.33}$$

$$= e^{\alpha_k} V + e^{-\alpha_k} U \,. \tag{4.34}$$

Two single parity check codes of size $2^{K_u - 2}$ have thus been created, identified by U and V. Since X had even and Y had odd parity, selecting all code words in X with $s_{a,k} = t_k$ and removing the k^{th} symbol results in an even parity check code of length $N_c - 2$. Conversely, selecting all code words in Y with $s_{a,k} = -t_k$ and removing k^{th} symbol results in an even single parity check code of length $N_c - 2$ which is identical to the one obtained from X. These codes are named U while the odd single parity check codes were named V. Again it is possible to obtain a more suitable expression:

$$X = e^{\alpha_k} U + e^{-\alpha_k} V \tag{4.35}$$

$$\propto (e^{\alpha_k} + e^{-\alpha_k})(U+V) + (e^{\alpha_k} - e^{-\alpha_k})(U-V) \tag{4.36}$$

$$= e^{\alpha_k}U + e^{-\alpha_k}U + e^{\alpha_k}V + e^{-\alpha_k}V + e^{\alpha_k}U - e^{-\alpha_k}U - e^{\alpha_k}V + e^{-\alpha_k}V \quad (4.37)$$

$$= 2(e^{\alpha_k}U + e^{-\alpha_k}V) \quad (4.38)$$

$$Y = e^{\alpha_k}V + e^{-\alpha_k}U \quad (4.39)$$

$$=\propto (e^{\alpha_k} + e^{-\alpha_k})(U + V) - (e^{\alpha_k} - e^{-\alpha_k})(U - V)\,. \quad (4.40)$$

Furthermore,

$$X - Y = \frac{1}{2}(e^{\alpha_k} - e^{-\alpha_k})(U - V) \quad (4.41)$$

$$X + Y = \frac{1}{2}(e^{\alpha_k} + e^{-\alpha_k})(U + V) \quad (4.42)$$

$$\frac{X - Y}{X + Y} = \tanh(\alpha_k)\frac{U - V}{U + V}\,. \quad (4.43)$$

This procedure can then be repeated until the smallest parity check codes are obtained. These codes have length two and consist of two code words. The codes are given by

$$U = e^{\alpha_0}e^{\alpha_1} + e^{-\alpha_0}e^{-\alpha_1} \quad (4.44)$$

$$\equiv \left\{(t_0, t_1)^T, (-t_0, -t_1)^T\right\} = \mathcal{C}_U^{\mathbb{R}} \quad (4.45)$$

$$V = e^{-\alpha_0}e^{\alpha_1} + e^{\alpha_0}e^{-\alpha_1} \quad (4.46)$$

$$\equiv \left\{(-t_0, t_1)^T, (t_0, -t_1)^T\right\} = \mathcal{C}_V^{\mathbb{R}}. \quad (4.47)$$

Then,

$$U - V = e^{\alpha_0}(e^{\alpha_1} - e^{-\alpha_1}) + e^{-\alpha_0}(e^{-\alpha_1} - e^{\alpha_1}) \quad (4.48)$$

$$= (e^{\alpha_0} - e^{-\alpha_0})(e^{\alpha_1} - e^{-\alpha_1}) \quad (4.49)$$

$$U + V = e^{\alpha_0}(e^{\alpha_1} + e^{-\alpha_1}) + e^{-\alpha_0}(e^{-\alpha_1} + e^{\alpha_1}) \quad (4.50)$$

$$= (e^{\alpha_0} + e^{-\alpha_0})(e^{\alpha_1} + e^{-\alpha_1}) \quad (4.51)$$

$$\frac{U - V}{U + V} = \tanh(\alpha_0)\tanh(\alpha_1)\,. \quad (4.52)$$

An expression for F_i of a single parity check code is thus given by

$$F_i = t_i \frac{\tanh(\alpha_i) + \prod_{j\neq i}\tanh(\alpha_j)}{1 + \prod_j \tanh(\alpha_j)}\,. \quad (4.53)$$

The expression (4.53) is problematic due to possible division by zero, in which case the numerator also vanishes, yielding the undefined form $0/0$. Using the relation

$$\tanh(x + y) = \frac{\tanh(x) + \tanh(y)}{1 + \tanh(x)\tanh(y)}\,, \quad (4.54)$$

a practically more usable expression for F_i is obtained as

$$F_i = t_i \tanh\left(\alpha_i + \operatorname{atanh}\left(\prod_{j\neq i}\tanh(\alpha_j)\right)\right)\,. \quad (4.55)$$

The derivation did not include any approximations and thus the formulation of F_i in (4.53) is exact. The function G_i can therefore be found as

$$G_i = \frac{I_{2,i}}{I_0} - F_i^2 = \frac{\sum_a s_{a,i}^2 I_0^{(a)}}{\sum_a I_0^{(a)}} - F_i^2 \,. \tag{4.56}$$

Again, $s_{a,i}^2 = t_i \,\forall a$ and it can thus be pulled out of the sum, yielding

$$G_i = t_i^2 \frac{\sum_a I_0^{(a)}}{\sum_a I_0^{(a)}} - F_i^2 = t_i^2 - F_i^2 \,. \tag{4.57}$$

This function follows intuition: as the algorithm converges towards the true value of the i^{th} symbol, which is $\pm t_i$, the variance becomes zero. The derivatives w.r.t. $\mu_j^{(l)}$ are

$$\frac{\partial I_0^{(a)}}{\partial \mu_j^{(l)}} = \frac{1}{\sigma_j^{2(l)}} \left(I_{1,j}^{(a)} - \mu_j^{(l)} I_0^{(a)} \right) \tag{4.58}$$

$$\frac{\partial I_{1,i}^{(a)}}{\partial \mu_j^{(l)}} = \frac{1}{\sigma_j^{2(l)}} \left(R_{1,1,i,j}^{(a)} - \mu_j^{(l)} I_{1,i}^{(a)} \right) \tag{4.59}$$

$$I_{1,i}^{(a)} = s_{a,i} I_0^{(a)} \tag{4.60}$$

$$R_{1,1,i,j}^{(a)} = s_{a,i} s_{a,j} I_0^{(a)} \,. \tag{4.61}$$

The derivative of F_i (4.12) computes as

$$\frac{\partial F_i}{\partial \mu_j^{(l)}} = \frac{1}{\sum_a I_0^{(a)}} \sum_a \frac{\partial I_{1,i}^{(a)}}{\partial \mu_j^{(l)}} - \frac{\sum_a I_{1,i}^{(a)}}{\left(\sum_a I_0^{(a)} \right)^2} \sum_a \frac{\partial I_0^{(a)}}{\partial \mu_j^{(l)}} \tag{4.62}$$

$$= \frac{\sum_a s_{a,i} s_{a,j} I_0^{(a)} - \mu_j^{(l)} \sum_a I_{1,i}^{(a)}}{\sigma_j^{2(l)} \sum_a I_0^{(a)}} \tag{4.63}$$

$$- \frac{F_i}{\sigma_j^{2(l)} \sum_a I_0^{(a)}} \left(\sum_a I_{1,j}^{(a)} - \mu_j^{(l)} \sum_a I_0^{(a)} \right) . \tag{4.64}$$

For the diagonal of the Jacobian, i.e. $i = j$, this results in

$$\frac{\partial F_i}{\partial \mu_i^{(l)}} = \frac{1}{\sigma_i^{2(l)}} \left(t_i^2 - \mu_i^{(l)} F_i - F_i^2 + \mu_i^{(l)} F_i \right) \tag{4.65}$$

$$= \frac{t_i^2 - F_i^2}{\sigma_i^{2(l)}} \,, \tag{4.66}$$

which agrees with (3.20). The off-diagonal terms are

$$\left.\frac{\partial F_i}{\partial \mu_j^{(l)}}\right|_{i\neq j} = \frac{1}{\sigma_j^{2(l)}}\left(\frac{\sum_a s_{a,i}s_{a,j}I_0^{(a)}}{\sum_a I_0^{(a)}} - \mu_j^{(l)}F_i - F_iF_j + \mu_j^{(l)}F_i\right) \tag{4.67}$$

$$= \frac{1}{\sigma_j^{2(l)}}\left(\underbrace{\frac{\sum_a s_{a,i}s_{a,j}I_0^{(a)}}{\sum_a I_0^{(a)}}}_{H_{i,j}} - F_iF_j\right). \tag{4.68}$$

The expression $H_{i,j}$ shall be examined more closely. Note that the sum in the numerator depends on the product of $s_{a,i}$ and $s_{a,j}$, which is $\pm t_i t_j$. For parity check codes of length $N_c = 2$, the values $s_{a,i}, s_{a,j}$ are highly correlated: the product $s_{a,i}s_{a,j} = t_i t_j$ for even parity check codes and $s_{a,i}s_{a,j} = -t_i t_j$ for odd parity check codes. For single parity check codes with $N_c > 2$, any two coordinates are pairwise uncorrelated, i.e. there is an equal number of code words where $s_{a,i} = s_{a,j}$ and $s_{a,i} \neq s_{a,j}$. The sum does not evaluate to zero however, since the functions $I_0^{(a)}$ can have different values. In a similar procedure as above, the sum is partitioned based on the sign of the product $s_{a,i}s_{a,j}$. The expression $H_{i,j}$ can therefore be written as

$$H_{i,j} = \frac{\sum_a s_{a,i}s_{a,j}I_0^{(a)}}{\sum_a I_0^{(a)}} = t_i t_j \frac{\sum_{a:s_{a,i}=s_{a,j}} I_0^{(a)} - \sum_{a:s_{a,i}\neq s_{a,j}} I_0^{(a)}}{\sum_{a:s_{a,i}=s_{a,j}} I_0^{(a)} + \sum_{a:s_{a,i}\neq s_{a,j}} I_0^{(a)}}. \tag{4.69}$$

Again, $I_0^{(a)}$ can be written as product of exponentials, cf. (4.18). The sums can be further partitioned, depending on the particular values of $s_{a,i}$ and $s_{a,j}$. Furthermore, the exponentials associated with the i^{th} and j^{th} coordinate can be extracted. For the individual sums, this yields

$$\sum_{a:s_{a,i}=s_{a,j}} I_0^{(a)} = \begin{cases} e^{\alpha_i+\alpha_j}\overbrace{\sum_a \prod_{k\neq i,j} e^{\alpha_{a,k}}}^{U} & a : s_{a,i} = t_i \wedge s_{a,j} = t_j \\ e^{-(\alpha_i+\alpha_j)}\underbrace{\sum_a \prod_{k\neq i,j} e^{\alpha_{a,k}}}_{U} & a : s_{a,i} = -t_i \wedge s_{a,j} = -t_j \end{cases} \tag{4.70}$$

$$\sum_{a:s_{a,i}\neq s_{a,j}} I_0^{(a)} = \begin{cases} e^{\alpha_i-\alpha_j}\overbrace{\sum_a \prod_{k\neq i,j} e^{\alpha_{a,k}}}^{V} & a : s_{a,i} = t_i \wedge s_{a,j} = -t_j \\ e^{-(\alpha_i-\alpha_j)}\underbrace{\sum_a \prod_{k\neq i,j} e^{\alpha_{a,k}}}_{V} & a : s_{a,i} = -t_i \wedge s_{a,j} = t_j\,. \end{cases} \tag{4.71}$$

It can be shown that the codes obtained by the expressions marked as U are identical. They are obtained by selecting all code words of a longer single parity

check code where two particular code word symbols (namely those at positions i and j) are equal. The new code of length $N_c - 2$ represented by U thus has the same parity as the code of length N_c. The codes marked as V are similarly obtained by selecting all code words whose symbols differ at positions i and j. The new code of length $N_c - 2$ thus has the opposite parity of the longer code of length N_c. Therefore, (4.69) can be written as

$$H_{i,j} = t_i t_j \frac{\left(e^{\alpha_i+\alpha_j} + e^{-(\alpha_i+\alpha_j)}\right) \sum\limits_{a:s_{a,i}=s_{a,j}} I_0^{(a)} - \left(e^{\alpha_i-\alpha_j} + e^{-(\alpha_i-\alpha_j)}\right) \sum\limits_{a:s_{a,i}\neq s_{a,j}} I_0^{(a)}}{\underbrace{\left(e^{\alpha_i+\alpha_j} + e^{-(\alpha_i+\alpha_j)}\right)}_{u} \sum\limits_{a:s_{a,i}=s_{a,j}} I_0^{(a)} + \underbrace{\left(e^{\alpha_i-\alpha_j} + e^{-(\alpha_i-\alpha_j)}\right)}_{v} \sum\limits_{a:s_{a,i}\neq s_{a,j}} I_0^{(a)}} \tag{4.72}$$

$$= t_i t_j \frac{uU - vV}{uU + vV} = t_i t_j \frac{(u-v)(U+V) + (u+v)(U-V)}{(u+v)(U+V) + (u-v)(U-V)} \tag{4.73}$$

$$= t_i t_j \frac{\dfrac{u-v}{u+v} + \dfrac{U-V}{U+V}}{1 + \dfrac{u-v}{u+v}\dfrac{U-V}{U+V}} . \tag{4.74}$$

Furthermore, the terms involving u and v can be conveniently dissected into

$$\frac{u-v}{u+v} = \frac{e^{\alpha_i} - e^{-\alpha_i}}{e^{\alpha_i} + e^{-\alpha_i}} \frac{e^{\alpha_j} - e^{-\alpha_j}}{e^{\alpha_j} + e^{-\alpha_j}} \tag{4.75}$$

$$= \tanh(\alpha_i) \tanh(\alpha_j) . \tag{4.76}$$

Assuming even parity check codes, the code represented by U is even and the one represented by V is odd. Analogous to the procedure described in (4.31) to (4.34), the terms involving U and V can therefore be factored into products of hyperbolic tangents. The function $H_{i,j}$ is thus defined as

$$H_{i,j} = t_i t_j \frac{\tanh(\alpha_i) \tanh(\alpha_j) + \prod_{k\neq i,j} \tanh(\alpha_k)}{1 + \prod_k \tanh(\alpha_k)} . \tag{4.77}$$

The derivative of F_i w.r.t. $\mu_j^{(l)}$ (cf. (4.68)) can therefore be written as

$$\left.\frac{\partial F_i}{\partial \mu_j^{(l)}}\right|_{i\neq j} = \frac{1}{\sigma_j^{2(l)}} \left(t_i t_j \frac{\tanh(\alpha_i) \tanh(\alpha_j) + \prod_{k\neq i,j} \tanh(\alpha_k)}{1 + \prod_k \tanh(\alpha_k)} - F_i F_j \right) . \tag{4.78}$$

Note that the expression in the brackets of (4.78) remains identical if i and j are swapped, which can be used for efficient implementation.

4.3 Consensus Function

It has been mentioned above that for each symbol x_i, there are outputs from $\gamma_{\boldsymbol{H}}$ different functions F, G, i.e. one for each parity check that a symbol x_i is associated with. To obtain a consensus among these results, a binary repetition code can be used. The functions F and G obtained by deriving the denoiser of the repetition code will prove to be suited as consensus function, i.e. for obtaining an estimate of x_i and its variance $\sigma^2_{\mathrm{x},i}$ as required by the MD-BAMP algorithm. The prior of a repetition code is

$$f_{\mathbf{x}}(\boldsymbol{x}) = \frac{1}{2}\prod_{k}^{K}\delta(x_k+\tau_k) + \frac{1}{2}\prod_{k}^{K}\delta(x_k-\tau_k)\,. \tag{4.79}$$

This prior describes a binary code in K-dimensional space, with the two possible code words being $\boldsymbol{c}_0 = (\tau_1,\ldots,\tau_k,\ldots,\tau_K)^T$ and $\boldsymbol{c}_1 = -\boldsymbol{c}_0$. This section describes the use as consensus function, thus all symbols τ_k are equal and the variables x_k are replicas of x_i. To obtain consensus for the symbol x_i of the LDPC code, let $\tau_k = t_i$. Since (4.79) is a linear mixture of two multivariate Dirac deltas, the expression (4.12) can be used to obtain the denoiser F, yielding

$$F_i = F_k = \frac{\sum_a s_{a,k} I_0^{(a)}}{\sum_a I_0^{(a)}} \tag{4.80}$$

$$= t_i \frac{\prod_{k=1}^{K} e^{\alpha_k} - \prod_{k=1}^{K} e^{-\alpha_k}}{\prod_{k=1}^{K} e^{\alpha_k} + \prod_{k=1}^{K} e^{-\alpha_k}}\,, \tag{4.81}$$

with the sum in (4.80) consisting of two terms and all $s_{a,k} \in \pm\tau_k$. Note that the expression for the k^{th} dimension of the denoiser (4.81) does not depend on k. Using the shorthand $\beta = \sum_k \alpha_k$, the fraction (4.81) becomes

$$F_i = t_i \frac{e^{\beta} - e^{-\beta}}{e^{\beta} + e^{-\beta}} = t_i \tanh(\beta)\,. \tag{4.82}$$

The function G can be obtained by following the steps performed for single parity check codes (cf. (4.56)). Thus, G is

$$G_i = G_k = t_i^2 - F_i^2 = t^2(1-\tanh^2(\beta))\,. \tag{4.83}$$

At this point it is necessary to find the definition of α_k. While it seems logical to use an expression like (4.17), with $\mu_k^{(l)}$ and $\sigma_k^{2(l)}$ set to the results of the functions F_i (cf. (4.55)) and G_i (cf. (4.57)) of the K single parity check nodes neighboring the symbol x_i, this choice yields poor performance. A heuristic approach inspired by the original SPA detector for LDPC codes gives better results. The SPA detector [41] operates on log-likelihood values l_i. Messages are passed between factor nodes,

which correspond to parity checks, and variable nodes, which represent the code word symbols. These messages are given by

$$l_{i\to k} = l_i + \sum_{m\neq k} l_{m\to i} \tag{4.84}$$

$$l_{k\to i} = 2\,\mathrm{atanh}\left(\prod_{j\neq i}\tanh\left(\frac{l_{j\to k}}{2}\right)\right). \tag{4.85}$$

Here, l_i is the log-likelihood value of the i^{th} symbol, while $l_{i\to k}$ and $l_{k\to i}$ are the messages passed from variable to factor and vice versa. Both messages can be found in the denoiser (4.55). The message $l_{k\to i}$ appears in (4.55) (up to a factor of 2) when setting $\alpha_i = 0$ and removing the outer tanh function. In (4.55), the sum covers just one parity check node, namely the node for which the denoiser was derived. Finally, the hyperbolic tangent is the denoiser of the repetition code, where the two values α_k are on the one hand α_i and on the other hand $\mathrm{atanh}(\prod_{j\neq i}\tanh(\alpha_j))$. Following (4.84), the variable β in (4.82) shall be defined as

$$\beta = \alpha_i + \sum_k \alpha_{k\to i}\,, \tag{4.86}$$

where the message $\alpha_{k\to i}$ is

$$\alpha_{k\to i} = \mathrm{atanh}\left(\prod_{j\neq i}\tanh(\alpha_{j\to k})\right) \tag{4.87}$$

and finally, the message $\alpha_{j\to k}$ is (cf. (4.84))

$$\alpha_{i\to k} = \alpha_i + \sum_{m\neq k} \alpha_{m\to i}\,. \tag{4.88}$$

In (4.86), α_i is defined as in (4.17), all other terms do not depend on $\mu_i^{(l)}$. Thus, the derivative of F_i w.r.t. $\mu_i^{(l)}$ is obtained as

$$\frac{\partial F_i}{\partial \mu_i^{(l)}} = \frac{t_i^2}{\sigma_i^{2(l)}}(1-\tanh(\beta))\,, \tag{4.89}$$

which is consistent with (4.83) and (3.20). This formulation is also justified by the graphical model in Fig. 4.4 (p.67). Setting α_i to zero in the denoiser of the single parity check nodes neighboring the symbol x_i is equivalent to excluding the message from the target variable in the computation of the reply, as required by the sum-product message passing rules (cf. (2.28)).
The variables α_j appearing in the denoiser (4.55) can be thought of as messages from the variable node representing the symbol x_j towards the parity check node f_k. The definition of α_j does not take into account messages from other parity check nodes. It is computed using a belief about the "received" symbol represented

by $\mu_j^{(l)}$, $\sigma_j^{2(l)}$ only. The formulation (4.88) correctly takes into account messages from other parity check nodes.
It should be pointed out that the denoiser obtained using definitions (4.82), (4.83) and (4.86)-(4.88) results in a "stateful" denoiser, i.e. it depends on previous iterations. It is therefore not straightforward to obtain results via state evolution estimation using Monte-Carlo methods.

Chapter 5

Results

In this chapter, results of numerical simulations are presented and interpreted. The performance criteria of choice is the signal-to-distortion ratio in decibel, defined as

$$\mathrm{SDR}_{\mathrm{dB}} = 10 \log_{10} \left(\frac{\|\boldsymbol{x}\|_2^2}{\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_2^2} \right) , \tag{5.1}$$

which is the negative normalized mean-squared error (NMSE) in decibel. Note that this definition has the unfortunate side-effect of making comparison of SDR values difficult if $\mathbf{x}$ has nonzero mean and the estimation of $\hat{\boldsymbol{x}}$ was unbiased. The estimated (recovered) signal is denoted $\hat{\boldsymbol{x}}$ while the original signal vector is written as $\boldsymbol{x}$. The subsampling ratio ρ is defined as

$$\rho = \frac{L}{N} . \tag{5.2}$$

Signals $\boldsymbol{x}$ are sampled from multi-dimensional priors $f_{\mathbf{x}}(\boldsymbol{x})$. Samples from these priors are concatenated into a vector of size N, i.e.

$$\boldsymbol{x} = (\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, \ldots, \boldsymbol{x}_{N/K}^T)^T , \tag{5.3}$$

where K is the dimension of the prior and the vectors $\boldsymbol{x}_i$ are samples of the prior $f_{\mathbf{x}}(\boldsymbol{x})$. The dimension N of the vector $\boldsymbol{x}$ is chosen such that it is an integer multiple of K, typically $N = 1000$. For algorithms that use an abort threshold ε, it is set as $\varepsilon = 10^{-4}$. In many cases, this threshold also limits the achievable SDR. The number of iterations is limited to $t_{\max} = 300$, unless specified otherwise. The algorithms used in this chapter are listed in Appendix A, p.109ff.
All simulations were performed using double-precision (IEEE 754-1985 [1]) floating point numbers and a custom simulation framework written in C++. For linear algebra, the "Eigen" template library (version 3) was used. Multi-threading and clustering features combined with the execution speed of optimized C++ code enabled high-resolution simulations.

5.1 Multi-dimensional Sparse Signals

Most classical recovery algorithms, like iterative thresholding and matching pursuit are applicable to sparse signals only. Furthermore they, as well as AMP, assume zero mean, i.i.d. coefficients of the unknown vector $\boldsymbol{x}$. To compare the performance of these algorithms, test signal vectors are drawn from a Bernoulli-Gaussian prior,

$$f_{\mathbf{x},\mathrm{BG}}(\boldsymbol{x}) = (1-\gamma)\delta(\boldsymbol{x}) + \gamma\mathcal{N}_{\mathbf{x}}(\mathbf{0},\boldsymbol{\Sigma})\,, \tag{5.4}$$

where, γ is a measure of the sparsity, and $\boldsymbol{\Sigma}$ is the covariance matrix. For one-dimensional priors the standard normal distribution is used. For multi-dimensional priors the covariance matrix is obtained by rotation of an uncorrelated multivariate Gaussian with unit variance in the first dimension and a variance of 0.1 in all other dimensions. The rotation matrix is chosen such that the axes become aligned with the unit vectors defined by the columns of a Hadamard matrix.
All simulations were performed with $N = 1000$ (unless specified otherwise) and L such that the subsampling ratio ρ ranges from $\rho = 0.01$ to $\rho = 0.985$ in increments of $\Delta\rho = 0.025$. The sparsity γ ranges from $\gamma = 0.025$ to $\gamma = 1$ in increments of $\Delta\gamma = 0.025$. The SDR is plotted for a range of $-5\mathrm{dB} \leq \mathrm{SDR}_{\mathrm{dB}} \leq 50\mathrm{dB}$, which is a reasonable trade-off maintaining visibility of regions where algorithms diverge (which results in low SDR), regions where excellent recovery is achieved and preserving sufficient contrast for phase transitions. Values outside this range are clamped towards the nearest value in the range. Both (MD-)BAMP and (MD-)GMP are supplied with the prior of the signal for recovery, while AMP is provided with the signal's variance. The iterative thresholding algorithms as well as matching pursuit (MP) and orthogonal matching pursuit (OMP) are provided with the sparsity γ of the signal. The abort threshold is set as $\varepsilon = 10^{-4}$ and the maximum number of iterations is limited to 100 except for MP and OMP, where the maximum number of iterations is γN. The algorithms are listed in Appendix A. For these simulations, the variant of MD-BAMP with simplified Onsager term (Algorithm 8, p.114) was used. Any state evolution estimations were performed using Monte-Carlo sampling to compute the expectation in (2.121).
Classical BAMP exhibits comparatively good performance in terms of SDR at a given subsampling ratio ρ for i.i.d. entries of $\boldsymbol{x}$, as shown in Fig. 5.1. It outperforms all other algorithms except Gaussian message passing, which offers similar performance (cf. Fig. 5.2a) but is significantly slower. Approximate message passing as well as orthogonal matching pursuit offer reasonable recovery. Their SDR is plotted in Fig. 5.2b and Fig. 5.2f respectively. Note that OMP achieves superior SDR for very sparse signals ($\gamma < 0.2$) since in many cases it recovers $\boldsymbol{x}$ exactly. This is possible if its choice of active entries of $\boldsymbol{x}$ is correct in each iteration and $\rho \geq \gamma$, i.e. there are at least as many measurements as active entries in $\boldsymbol{x}$. Its recovery performance for dense signals $\gamma > 0.7$ is inferior to AMP. Our implementation of AMP exhibits instability at small ρ, i.e. for a small number of measurements, as can be seen in Fig. 5.2b. Finally, iterative hard thresholding and matching pursuit are suitable only for very for sparse signals ($\gamma < 0.3$) and achieve comparably

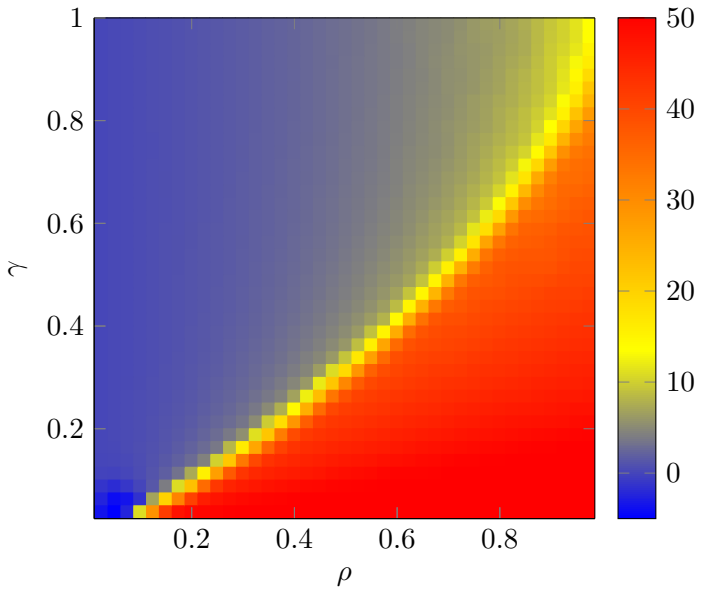


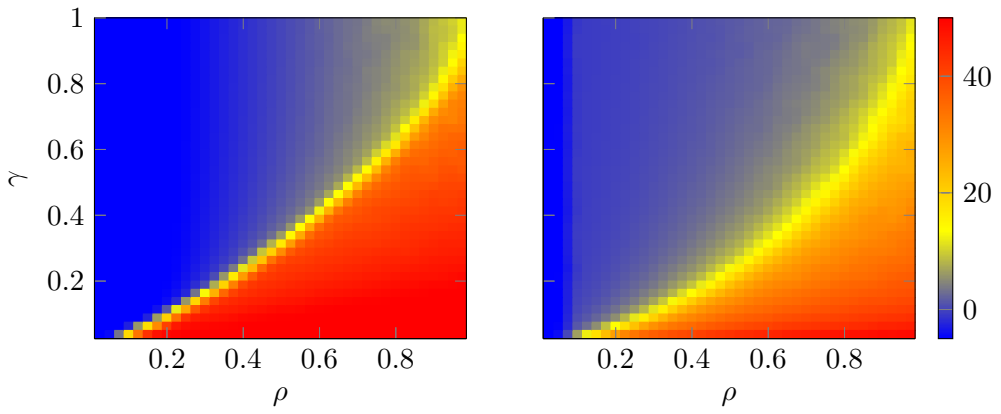
Figure 5.1: Recovery performance (SDR, color coded) of BAMP vs. sparsity γ and subsampling ratio ρ for i.i.d. entries of $\boldsymbol{x}$ with Bernoulli-Gaussian prior.

mediocre SDR, shown in Fig. 5.2d and Fig. 5.2e. Iterative soft thresholding (IST) has similar performance. Note that our particular variant of the IST algorithm becomes unstable for $\gamma > 0.5$ (i.e. rather dense signals).
Comparison is simplified by plotting the "phase transition" vs. subsampling ratio ρ and sparsity γ. Since the phase transition is not sharp for finite N, a particular recovery SDR value has to be chosen. Here, a threshold of 20dB SDR was applied. The phase transition becomes steeper for larger signal length N. An example of this effect can be see in Fig. 5.3. Thus, 20dB were chosen as slightly above the point where the recovery SDR is identical for a fixed subsampling ratio ρ and variable signal length N. The phase transitions for one-dimensional priors are plotted in Fig. 5.4. Note that the theoretical limit for recovery, i.e. the minimum subsampling ratio required for recovery of a signal of a given sparsity is not achieved by any algorithm. This limit can be found using the Rényi information dimension $d(\mathsf{x})$. Let the random variable x be distributed according to the measure ν,

$$\nu = (1-\gamma)\nu_d + \gamma\nu_c \,, \tag{5.5}$$

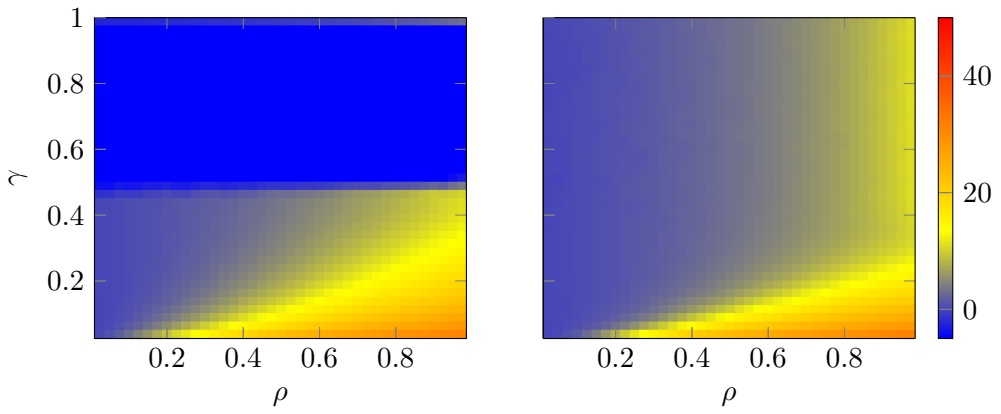
where $0 \leq \gamma \leq 1$, ν_d is a discrete measure and ν_c is an absolutely continuous measure. Then [51, Theorem 3]

$$d(\mathsf{x}) = \gamma \,. \tag{5.6}$$



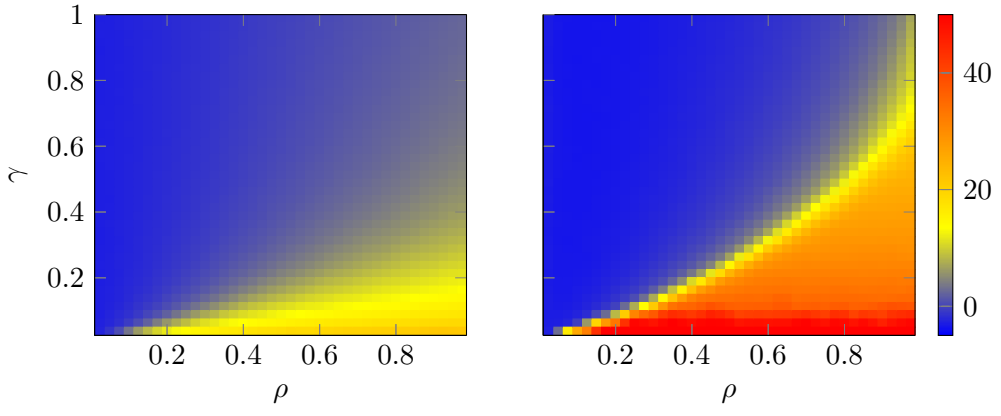
(a) Gaussian message passing

(b) Approximate message passing



(c) Iterative soft thresholding

(d) Iterative hard thresholding



(e) Matching pursuit

(f) Orthogonal matching pursuit

Figure 5.2: Performance of several recovery algorithms in terms of SDR (color coded) vs. sparsity γ and subsampling ratio ρ for i.i.d. entries of $\boldsymbol{x}$ with Bernoulli-Gaussian prior. The color coding is identical to Fig. 5.1 for all plots.

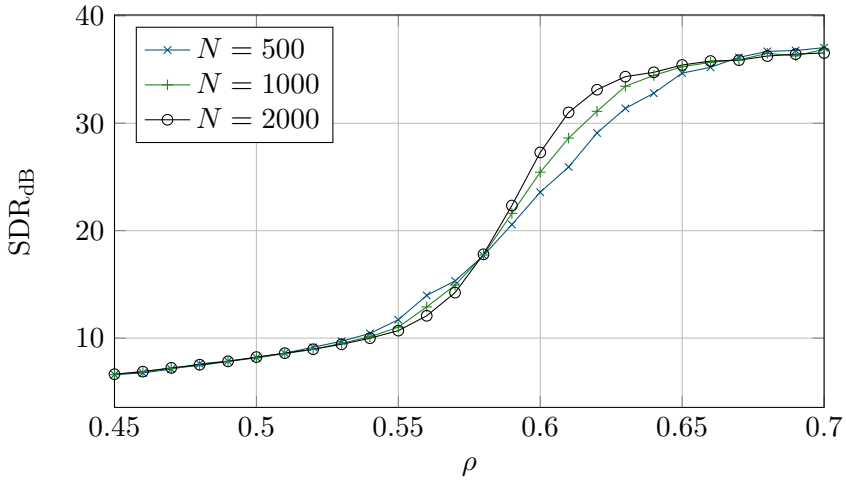


Figure 5.3: Performance (SDR) of MD-BAMP at various dimensions N and sub-sampling factors ρ for a two-dimensional prior. The sparsity is constant at $\gamma = 0.5$. The granularity of the measurement dimension is $0.01N$. Each point is obtained as average of 1000 simulations.

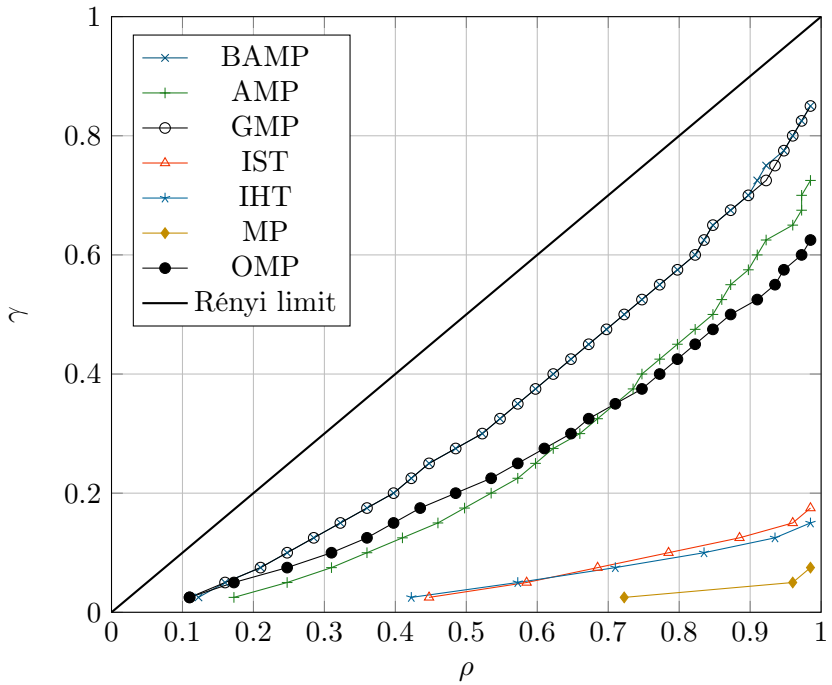
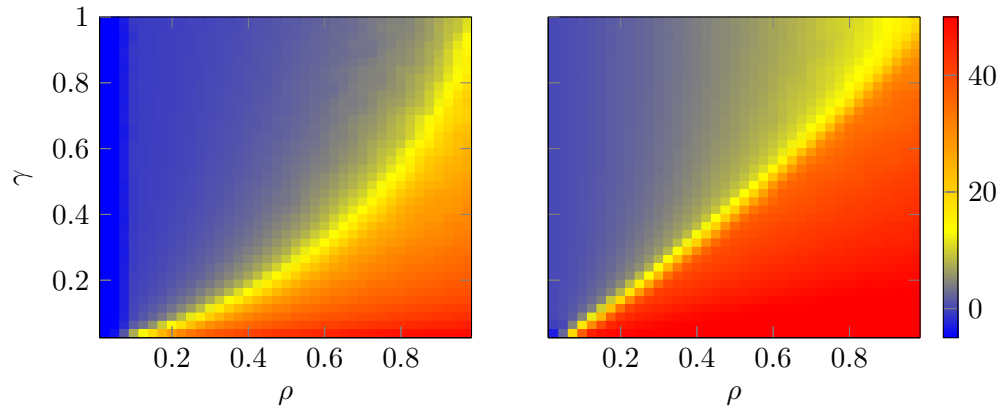


Figure 5.4: Phase transitions of several algorithms. The SDR of a particular algorithm is larger than 20dB southeast of the associated curve.

Wu and Verdú show [64, 65, 67] that under the assumption of linear encoding, a signal originating from a memoryless source distributed according to a discrete-continuous mixture can be reconstructed if the rate of the encoder is at least $d(\mathsf{x})$. The theoretical subsampling limit is therefore $\rho = \gamma$.
For multi-dimensional priors, MD-BAMP achieves better recovery compared to all other algorithms, which assume i.i.d. components of the unknown vector $\boldsymbol{x}$. For a $K = 2$-dimensional prior, the recovered SDR can be seen in Fig. 5.5a for AMP and Fig. 5.5b for MD-BAMP. Phase transitions for increasing prior dimension K can be seen in Fig. 5.6a. For increased dimension, MD-BAMP approaches the Rényi limit. This behavior is predicted by the state evolution recursion defined by equations (2.121), (2.122). The 20dB-thresholds estimated by state evolution are plotted as dotted lines in Fig. 5.6b.
The state evolution recursion can also be plotted in the style of an "Exit-plot", where the evolution of σ_{z}^2 is plotted as function of $\sigma_{\mathsf{x}}^{2(l)}$ and vice versa, pertaining to equations (2.121), (2.122). Extrinsic information transfer plots ("ExIT") were initially developed by Stephan ten Brink [56, 57] to help with analysis and design of iteratively decoded codes. Even though different quantities are plotted in case of AMP-based algorithms, the name "Exit" plot is reused here due to the similar style and operation. Plots of this type are shown in Fig. 5.7. If there is no measurement noise σ_{w}^2, then $\sigma_{\mathsf{x}}^{2(l)} = \sigma_{\mathsf{z}}^2$ and the equation describes a straight line. The behavior of the denoiser F determines the variance σ_{z}^2 depending on $\sigma_{\mathsf{x}}^{2(l)}$. Note that the factor ρ^{-1} is accounted for in the equation of the denoiser in this section's plots. This effectively separates the effects of additive measurement noise and the action of the denoiser at a given subsampling ratio and allows for easier comparison. Clearly, once the condition

$$\rho^{-1} \underbrace{\mathbb{E}\left\{(F(\ldots, \sigma_{\mathsf{x}}^{2(l)}) - x_0)^2\right\}}_{=G(\ldots,\sigma_{\mathsf{x}}^{2(l)})} = \sigma_{\mathsf{x}}^{2(l)} \tag{5.7}$$

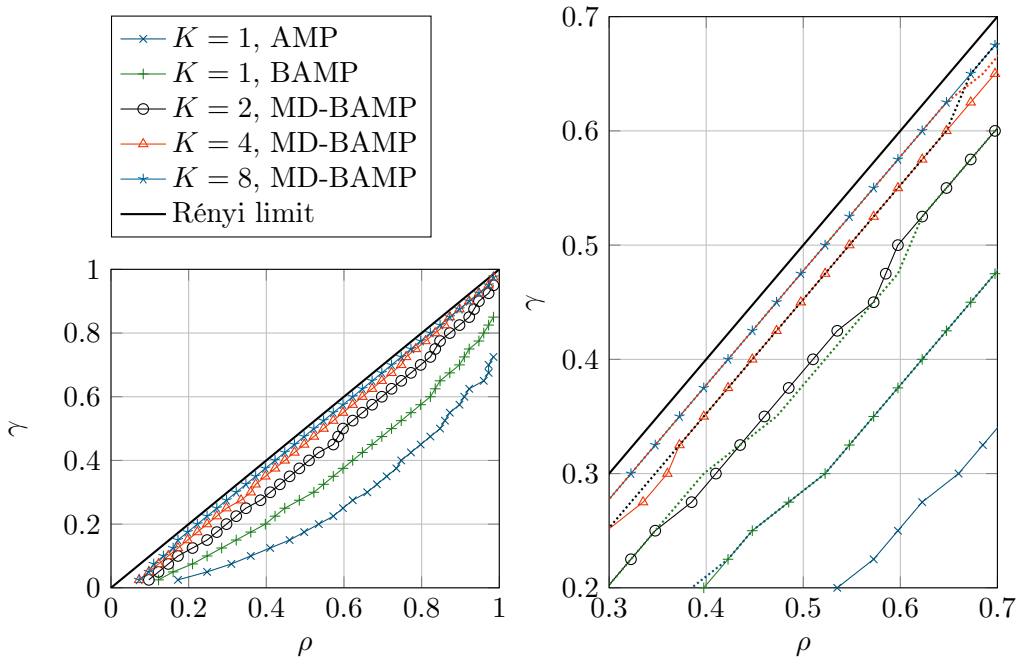
is fulfilled, the estimate cannot be improved any more and the algorithm converges towards a fixed point. The achievable SDR is determined by the point where the two curves meet. To achieve convergence of $\hat{\boldsymbol{x}}$ towards the true value of $\boldsymbol{x}$, a "channel" must remain open between the curves describing the behavior of the denoiser for a certain subsampling ratio ρ and the action of the matrix projection and possible measurement noise. For large $\sigma_{\mathsf{x}}^{2(l)}$, the MSE of the denoiser F saturates at the prior variance of the signal. Decreasing the subsampling ratio ρ amounts to larger σ_{z}^2 at the output of the denoiser F for a given variance $\sigma_{\mathsf{x}}^{2(l)}$. In a logarithmic plot, halving ρ amounts to a shift of the curve describing the denoiser by 3dB. For priors with Rényi dimension $d(\mathsf{x}) > 0$ it is therefore possible to indicate the Rényi limit: it is impossible for any denoiser to perform better than this limit for all $\sigma_{\mathsf{x}}^{2(l)}$. For $d(\mathsf{x}) = 1$ (i.e. signals without discrete components), the Rényi limit coincides with the diagonal describing the matrix projection.



(a) Approximate message passing

(b) Bayesian approximate message passing for multi-dimensional priors

Figure 5.5: Performance in terms of SDR of AMP and MD-BAMP for a two-dimensional Bernoulli-Gaussian prior with sparsity γ and subsampling ratio ρ.



(a) (b)

Figure 5.6: Phase transitions for priors of different dimensions K. The SDR of a particular algorithm is larger than 20dB southeast of the associated curve. The phase transitions as estimated by state evolution are plotted as dotted lines (right side only). The corresponding curves lie on top of each other, thus different colors were selected to enhance visibility.

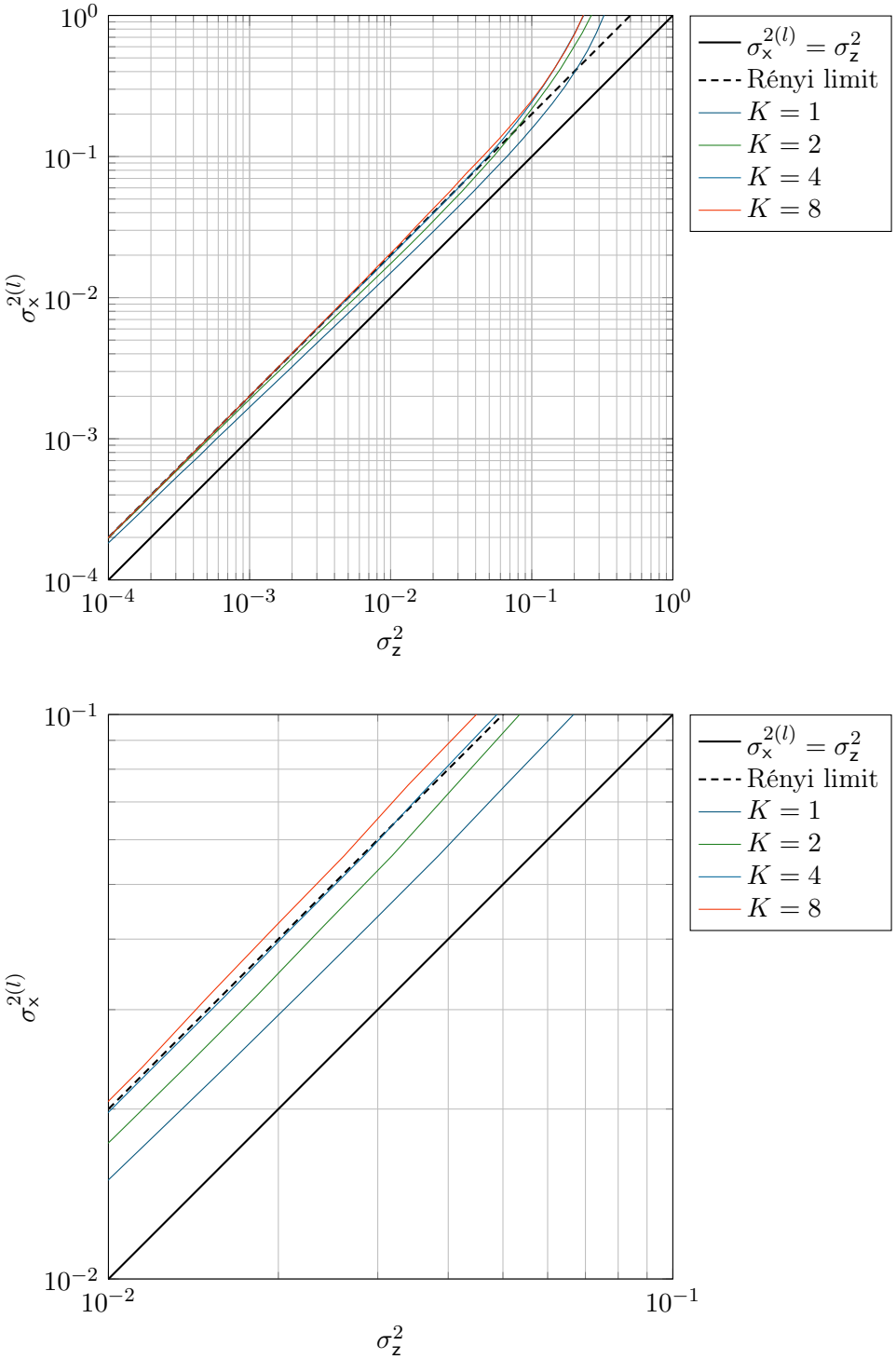


Figure 5.7: "Exit" plot of a Bernoulli-Gaussian signal with K-dimensional prior, sparsity $\gamma = 0.5$ and subsampling ratio $\rho = 1$.

5.2 Noisy Compressed Sensing

This section discusses the behavior of MD-BAMP and BAMP-VN in presence of non-uniformly distributed measurement noise. Signals in this section are drawn from a sparse multi-variate Bernoulli-Gaussian prior as defined in (5.4). The sparsity is fixed at $\gamma = 0.5$. The signal-to-noise ratio (SNR) in decibel is defined as

$$\mathrm{SNR}_{\mathrm{dB}} = 10 \log_{10} \left(\frac{\|\boldsymbol{A}\boldsymbol{x}\|_2^2}{\|\boldsymbol{w}\|_2^2} \right), \tag{5.8}$$

where $\boldsymbol{w}$ is the measurement noise which is assumed to be distributed according to the normal distribution. The behavior of MD-BAMP in presence of i.i.d. noise can be illustrated using state evolution and "Exit" plots. For both MD-BAMP and BAMP-VN the state evolution recursion consists of two equations. The behavior of the denoiser does not depend on the measurement noise $\boldsymbol{w}$. The second equation depends on the measurement noise variance σ_{w}^2. For MD-BAMP, the equation reads (cf. (2.122))

$$\sigma_{\mathsf{x}}^{2(l)} = \sigma_{\mathsf{w}}^2 + \sigma_{\mathsf{z}}^2 \,, \tag{5.9}$$

while for BAMP-VN it is (cf. (2.165))

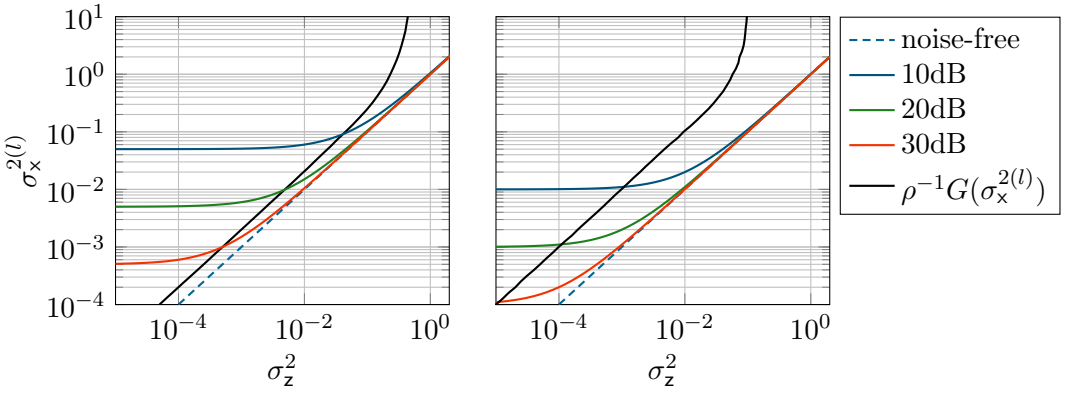
$$\sigma_{\mathsf{x}}^{2(l)} = L \left(\sum_{\mathcal{W}_k \in \mathcal{V}} \frac{|\mathcal{W}_k|}{\sigma_{\mathsf{w},k}^2 + \sigma_{\mathsf{z}}^2} \right)^{-1} . \tag{5.10}$$

Clearly, the variance $\sigma_{\mathsf{x}}^{2(l)}$ is lower bounded by the average noise variance σ_{w}^2 in (5.9). The behavior is plotted in Fig. 5.8. The expression (5.10) is identical to (5.9) if noise variances $\sigma_{\mathsf{w},k}^2$ are equal. In all other cases, the sum is dominated by the smallest $\sigma_{\mathsf{w},k}^2$. Indeed, as long as there is a set $\mathcal{W}_k$ whose entries w_a of $\mathbf{w}$ have zero variance, the variance $\sigma_{\mathsf{x}}^{2(l)}$ tends to zero as σ_{z}^2 goes to zero. This does not necessarily mean that having any number $|\mathcal{W}_k|$ of low-noise (or even noiseless) measurements y_a is sufficient to guarantee convergence towards the true $\boldsymbol{x}$. An upper bound for $\sigma_{\mathsf{x}}^{2(l)}$ is

$$\sigma_{\mathsf{x}}^{2(l)} = \left(\sum_{\mathcal{W}_k \in \mathcal{V}} \frac{|\mathcal{W}_k|}{L(\sigma_{\mathsf{w},k}^2 + \sigma_{\mathsf{z}}^2)} \right)^{-1} < \left(\frac{|\mathcal{W}_k|}{L\sigma_{\mathsf{w},\min}^2 + L\sigma_{\mathsf{z}}^2} \right)^{-1} , \tag{5.11}$$

where on the right-hand side the set $\mathcal{W}_k$ with the smallest variance was selected an all other terms in the sum were omitted. If the contrast of the noise variances $\sigma_{\mathsf{w},k}^2$ is large, e.g. if $\sigma_{\mathsf{w},\min}^2 = 0$, then the upper bound is a good approximation of $\sigma_{\mathsf{x}}^{2(l)}$ and

$$\sigma_{\mathsf{x}}^{2(l)} \approx \frac{L}{|\mathcal{W}_k|} \left(\sigma_{\mathsf{w},\min}^2 + \sigma_{\mathsf{z}}^2 \right) \tag{5.12}$$



(a) Bernoulli-Gaussian with sparsity $\gamma = 0.5$.

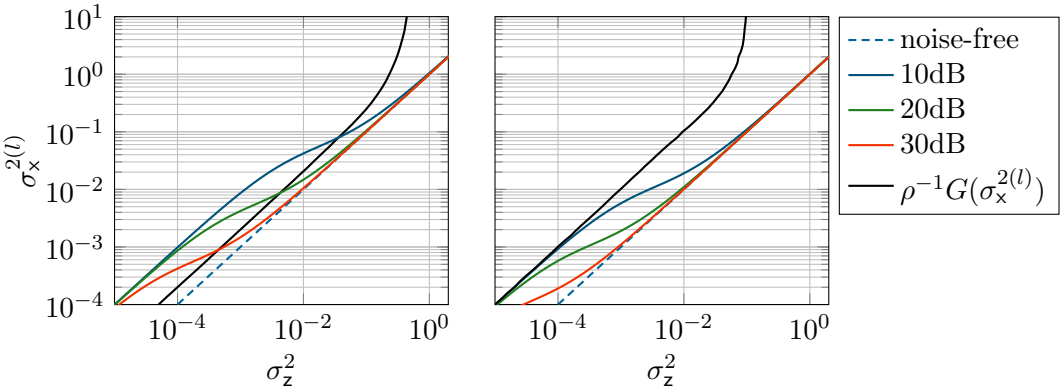
(b) Bernoulli-Gaussian with sparsity $\gamma = 0.1$.

Figure 5.8: "Exit" plot for MD-BAMP with i.i.d. Gaussian noise. The signal samples are drawn from an 8-dimensional Bernoulli-Gaussian prior, the subsampling ratio $\rho = 1$. Three different noise levels are plotted.

$$= \frac{L}{|\mathcal{W}_k|}\sigma_{\mathsf{w},\min}^2 + \frac{N}{|\mathcal{W}_k|}\sigma_{\mathsf{x}}^2 \,. \tag{5.13}$$

Since $|\mathcal{W}_k| \leq L$, the effective noise variance $\sigma_{\mathsf{w}}^2 \approx L\sigma_{\mathsf{w},\min}^2/|\mathcal{W}_k|$ is usually larger than the smallest noise variance. Furthermore, the variance σ_{x}^2 at the output of the denoiser is scaled with $N/|\mathcal{W}_k|$ instead of N/L. This result is intuitive: if all but a certain set of measurements y_a are extremely noisy, the algorithm's performance should be comparable to that of an algorithm which is supplied with the $L' = |\mathcal{W}_k|$ noise-free measurements only. This is equivalent to a subsampling ratio $\rho' = |\mathcal{W}_k|/N$.

This behavior can be observed in Fig. 5.9, where 10% of measurements are noise-free while the other 90% are affected by Gaussian noise with identical variance. In regions of large σ_{z}^2, the noise term in (5.9) is negligible. Conversely, for small σ_{z}^2, only noise-free measurements can further improve the estimation. Since only 10% of measurements are noise-free, the lines associated with the matrix projection equation are converging towards a parallel of the line showing evolution in the noise-free case. This parallel is shifted by 10dB to the left. This is roughly equivalent to using a subsampling ratio of $\rho = 0.1$, as predicted by approximation (5.13). For the signal with sparsity $\gamma = 0.5$, BAMP-VN does not perform significantly better than MD-BAMP. The behavior of BAMP-VN can be observed in Fig. 5.9a. The fixed points for BAMP-VN are almost identical to MD-BAMP, which assumes i.i.d. noise and thus behaves as shown in Fig. 5.8a. Since 10% of measurements are noiseless, the signal with sparsity $\gamma = 0.1$ is the densest recoverable signal. This can be observed in Fig. 5.9b. The Bayesian denoiser for the 8-dimensional Bernoulli-Gaussian prior does not quite achieve the Rényi bound, thus BAMP-VN



(a) Bernoulli-Gaussian with sparsity $\gamma = 0.5$.

(b) Bernoulli-Gaussian with sparsity $\gamma = 0.1$.

Figure 5.9: "Exit" plot for BAMP-VN with non-uniform Gaussian noise. The signal samples are drawn from an 8-dimensional Bernoulli-Gaussian prior, the subsampling ratio $\rho = 1$. A fraction of 10% of measurements are noise-free. Three different average noise levels are plotted.

does not converge towards $\boldsymbol{x}$ exactly for the $\gamma = 0.1$ sparse signal. Its fixed points promise much better recovery SDR compared to regular MD-BAMP (cf. Fig. 5.8b). Any sparser signal (i.e. $\gamma < 0.1$) will show even better performance for this noise pattern.

So far, the two extreme cases have been examined, namely i.i.d. Gaussian noise and "sparse" noise, where some measurements y_a are noisy while others are noiseless. Even for other noise-patterns, BAMP-VN shows improved SDR, however. To demonstrate this effect, assume an acquisition system in which two different types of sensors exist. One type of sensor has a superior noise figure such that the variance of the noise affecting the measurements is smaller by a certain factor $1/\beta$. Furthermore, assume that two thirds of the sensors are of the low-noise type. In such cases BAMP-VN improves upon MD-BAMP's recovery SDR due to the fact that $\sigma_x^{2(l)}$ converges to smaller values. An "Exit" plot comparing the behavior can be seen in Fig. 5.10. Numerical results are shown in Fig. 5.11. Given a noise variance contrast of $\beta = 10$, BAMP-VN (Fig. 5.11a) yields slightly better results than MD-BAMP (Fig. 5.11d). Larger values of β furthermore improve performance, enabling BAMP-VN to work in low-SNR conditions, as shown in Fig. 5.11b. In case of sparse noise, recovery is possible at any SNR at the cost of an increased subsampling ratio. This rate approaches $3\gamma/2$ as the SNR becomes smaller. This can be seen in Fig. 5.11c and agrees with (5.13).

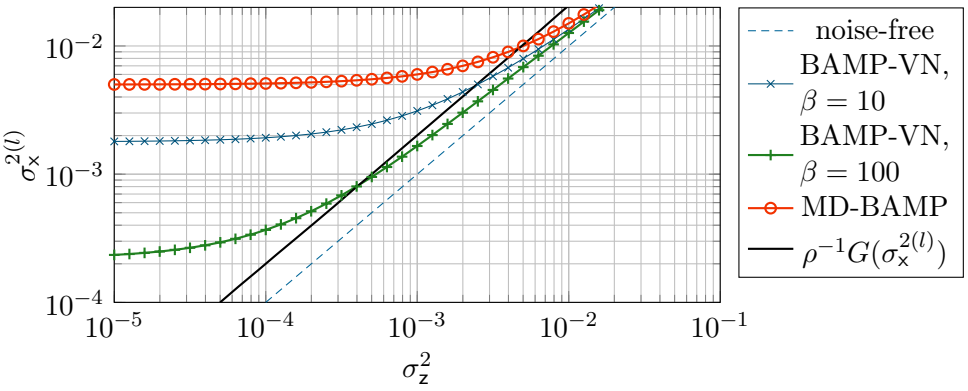
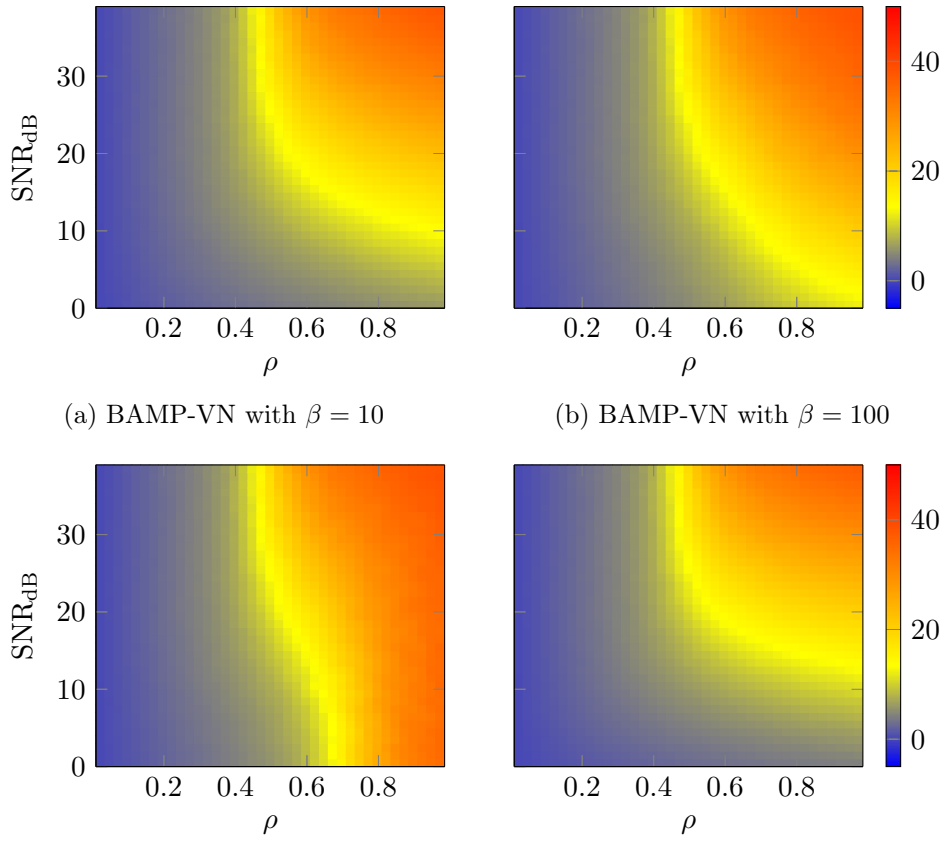


Figure 5.10: "Exit" plot comparing the performance of BAMP-VN and MD-BAMP for variable noise. The subsampling ratio $\rho = 1$, the average noise level is 20dB.



(a) BAMP-VN with $\beta = 10$

(b) BAMP-VN with $\beta = 100$

(c) BAMP-VN with sparse noise

(d) MD-BAMP

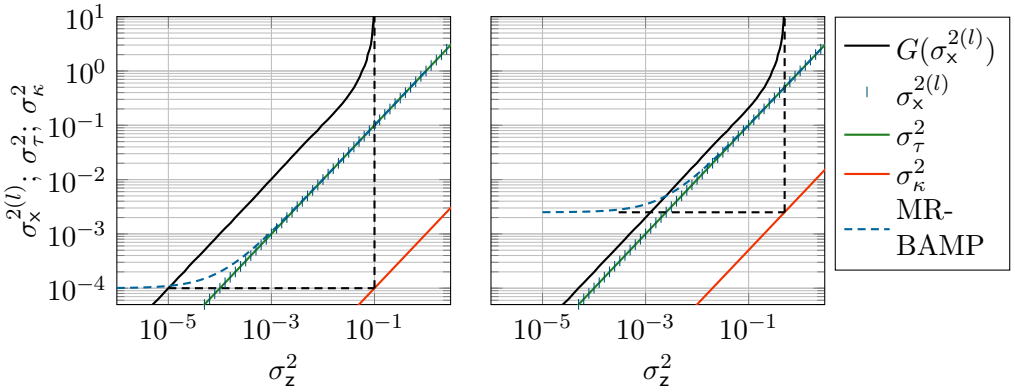
Figure 5.11: Recovery SDR for various noise patterns and SNR. Two thirds of measurements are affected by noise whose variance is smaller by $1/\beta$. In case of sparse noise, two thirds of measurements are noiseless. The signal samples are drawn from an 8-dimensional Bernoulli-Gaussian prior with sparsity $\gamma = 0.5$.

5.3 Non-Zero-Mean Sensing Matrix

This section discusses the behavior of MEAN-BAMP for systems with non-zero-mean measurement matrix $\boldsymbol{B}$. Furthermore, the performance of MD-BAMP with simple mean-removal is examined. Examples in this section use two different types of signals. A zero-mean signal is sampled from an 8-dimensional Bernoulli-Gaussian prior with sparsity $\gamma = 0.5$, while a nonzero-mean signal is obtained from a similar prior, shifted such that its mean is $\boldsymbol{\mu}_{\mathbf{x}} = \mathbf{1}_N$, i.e.

$$f_{\mathbf{x}}(\boldsymbol{x}) = (1-\gamma)\delta(\boldsymbol{x} - \mathbf{1}_8) + \gamma\mathcal{N}(\mathbf{1}_8, \boldsymbol{\Sigma}) \,, \tag{5.14}$$

with $\boldsymbol{\Sigma}$ as defined in Section 5.1. This nonzero-mean signal is used in numerical simulations showing that MEAN-BAMP behaves as predicted by state evolution even when both the sensing matrix $\boldsymbol{B}$ and the signal $\boldsymbol{x}$ are non-zero-mean.
Again, state evolution lends itself well to the analysis of MEAN-BAMP as well as MD-BAMP with mean-removal (MR-BAMP). A possible state evolution recursion for MEAN-BAMP is given by the expressions (2.237) to (2.240). Both algorithms are initialized with a sensing matrix $\boldsymbol{A}$ and a vector of measurements $\boldsymbol{y}$ which are obtained from the mean-affected matrix $\boldsymbol{B}$ and the original measurement vector $\boldsymbol{v}$ by removal of the arithmetic mean. As shown in Section 2.7, this incurs an additional error with a variance of $L^{-1}\sigma_{\mathsf{y}}^2$. While MEAN-BAMP iteratively eliminates this error by estimating the correct arithmetic mean of a particular measurement vector $\boldsymbol{y}$, MR-BAMP ignores it and is therefore limited by an additional noise term. The initial variance of $\boldsymbol{y}$'s entries is captured by σ_τ^2 in (2.238). The variance of the error is smaller by a factor L^{-1}. The variance σ_κ^2 is roughly equivalent to the error variance provided μ_B is large enough. The initial error variance can therefore be identified graphically by finding the initial value of σ_z^2 and determining the associated value of σ_κ^2. MR-BAMP is limited by a noise-floor at this level as shown in Fig. 5.12. Since $\sigma_{\mathsf{x}}^{2(l)}$ is the sum of σ_κ^2 and σ_τ^2, the latter is almost identical to $\sigma_{\mathsf{x}}^{2(l)}$ due to the former being orders of magnitude smaller. For any reasonably large L and matrix mean μ_B, it is therefore possible to use state evolution as defined for MD-BAMP ((2.121) - (2.122)) to estimate the performance of MEAN-BAMP. The artificial noise level is 30dB for $L = 1000$ and roughly 23dB for $L = 200$, which can be seen in Fig. 5.12a and Fig. 5.12b respectively.
Numerical simulations confirm the findings obtained via state evolution. As shown in Fig. 5.13, when MEAN-BAMP is used in settings with $\mu_B \neq 0$, it achieves performance comparable to MD-BAMP in the zero-mean-matrix regime. In fact, the recovered signal is almost identical. This agrees well with state evolution which is highly similar to MD-BAMP. Merely employing mean-removal is roughly identical to a noisy signal. In Fig. 5.13 it can be seen that MR-BAMP has similar performance as MD-BAMP operating at an equivalent noise level and zero-mean matrix. While the state evolution estimation (dotted) agrees well with the performance in the noisy case, MR-BAMP slightly outperforms the estimation. The precise reason for this deviation has not yet been thoroughly investigated.



(a) $L = 1000$ ($\rho = 1$) (b) $L = 200$ ($\rho = 0.2$)

Figure 5.12: "Exit" plot for MEAN-BAMP and MR-BAMP. The sparsity $\gamma = 0.1$, $N = 1000$. The sensing matrix mean $\mu_B = 1$.

The MEAN-BAMP algorithm is also applicable to signals with non-zero mean. Results for this case can be seen in Fig. 5.14. Values of the SDR at small subsampling ratio ρ start at a higher value due to (5.1) not accounting for the signal mean.

5.4 Other Sparse Priors

Some signals are not accurately modeled by Bernoulli-Gaussian priors. Other priors can be obtained using the methods outlined in Chapter 3. In this section, numerical results for a small number of sparse priors are presented. The main goal is comparison of phase transition curves with those of Bernoulli-Gaussian priors. Two different mixed discrete-continuous distributions are discussed. One is a Bernoulli-Exponential mixture while the second is a Bernoulli-Uniform mixture. Both are defined for prior dimensions $K \in \{1, 4\}$. Their pdfs are given by

$$f_{\mathsf{x},\mathrm{BE}}^{(K=1)}(x) = (1-\gamma)\delta(x-1) + \gamma \exp(-x) \tag{5.15}$$

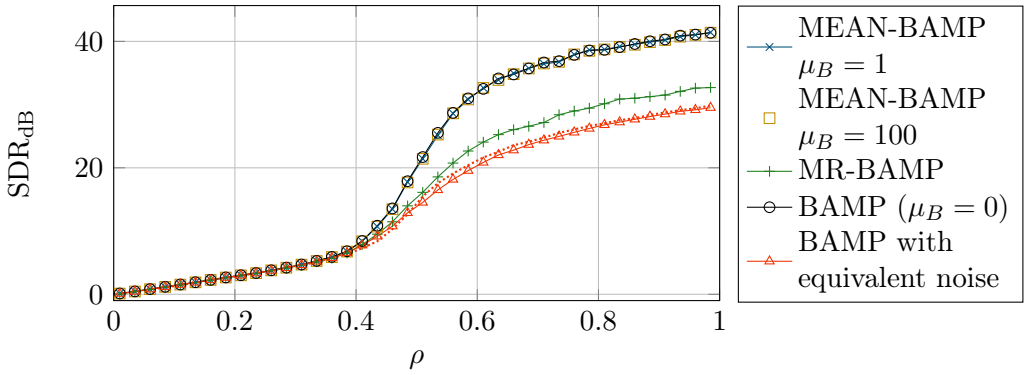
$$f_{\mathbf{x},\mathrm{BE}}^{(K=4)}(\boldsymbol{x}) = (1-\gamma)\delta(\boldsymbol{x}-\mathbf{1}_4) + \gamma \exp(-\mathbf{1}_4^T\boldsymbol{x}) \tag{5.16}$$

$$f_{\mathsf{x},\mathrm{BU}}^{(K=1)}(x) = (1-\gamma)\delta(x) + \gamma\mathcal{U}(-1,1) \tag{5.17}$$

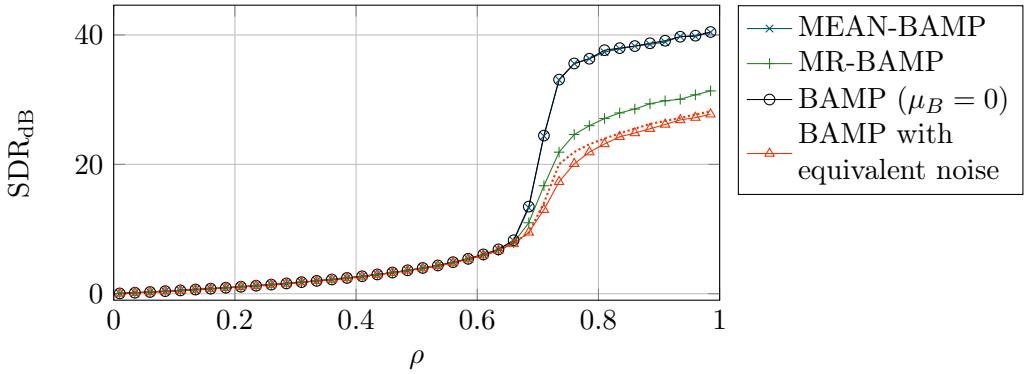
$$f_{\mathbf{x},\mathrm{BU}}^{(K=4)}(\boldsymbol{x}) = (1-\gamma)\delta(\boldsymbol{x}) + \frac{\gamma}{2}\left(\mathcal{U}(-\mathbf{1}_4, \mathbf{0}_4) + \mathcal{U}(\mathbf{0}_4, \mathbf{1}_4)\right), \tag{5.18}$$

where $\mathcal{U}(a,b)$ denotes the uniform distribution with the support defined by the minimum a and the maximum b. Furthermore, $\mathcal{U}(\boldsymbol{a},\boldsymbol{b})$ is the product of uniform distributions, i.e.

$$\mathcal{U}(\boldsymbol{a},\boldsymbol{b}) = \prod_k \mathcal{U}(a_k, b_k)\,. \tag{5.19}$$



(a) $K = 8$-dimensional prior



(b) $K = 1$-dimensional prior, $\mu_B = 1$

Figure 5.13: Performance of MEAN-BAMP and MR-BAMP in terms of SDR vs subsampling ratio ρ.

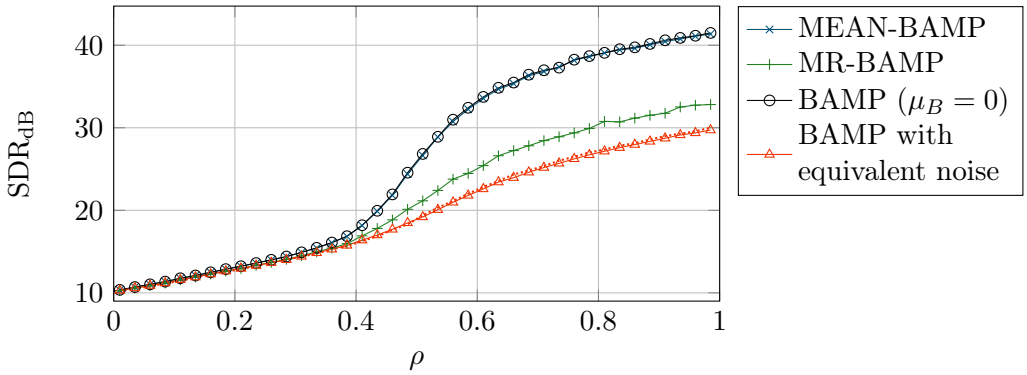


Figure 5.14: $K = 8$-dimensional prior, $\mu_B = 1$, $\mu_x = 1$

Note that while the individual components of the multi-dimensional priors can be factored, this is not true for their linear combination. Thus, the entries of $\mathbf{x}$ are dependent. This dependence is exploited by MD-BAMP to achieve successively better SDR for higher dimensions. The recovery SDR achieved by MD-BAMP can be seen in Fig. 5.16. The Bernoulli-Exponential distribution is one-sided and has a mean of $\mu_x = 1$, therefore the SDR towards the northwest (i.e. for small subsampling ratio ρ and dense signals) is relatively large. The support of the Bernoulli-Uniform distribution is bounded, contrary to the two other distributions, whose support is not bounded. Thus, while the denominator of (5.1) can become arbitrarily large for Bernoulli-Gaussian and Bernoulli-Exponential distributions, it is limited for Bernoulli-Uniform distributions. The SDR is thus not easily comparable. The phase transitions for the various priors are similar, however. They are shown in Fig. 5.15. The 20dB-boundary is not particularly well suited here since it creates the impression of MD-BAMP performing better than the theoretical optimum. It should be noted again that the Rényi limit specifies the minimum rate needed for *perfect* recovery, i.e. SDR $= \infty$.

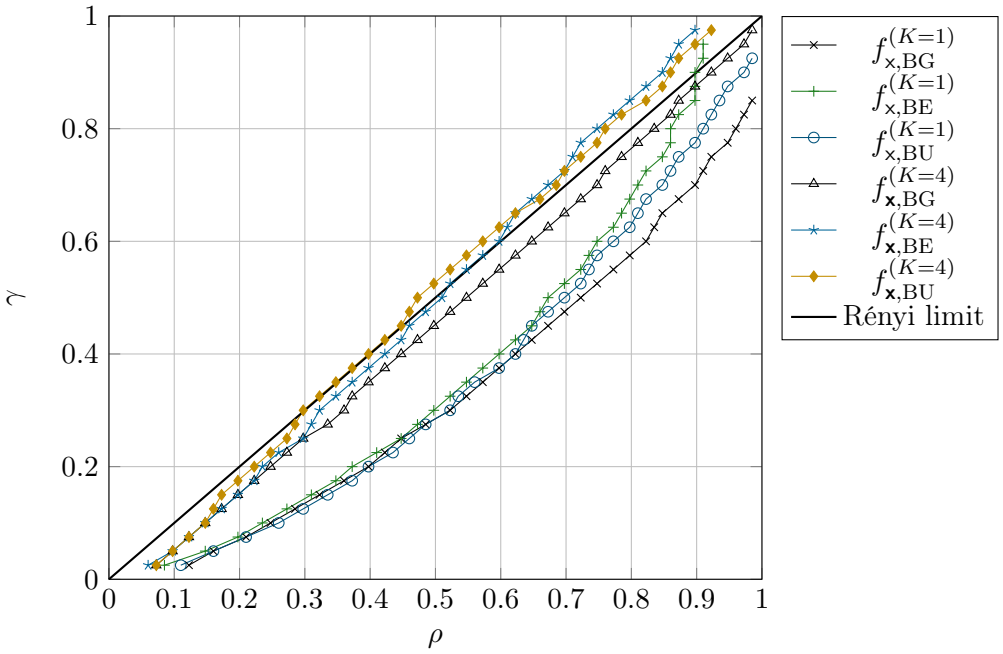
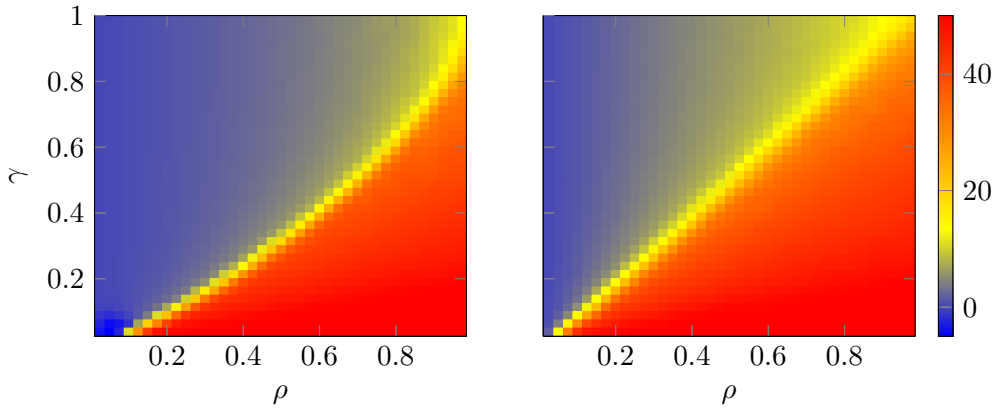
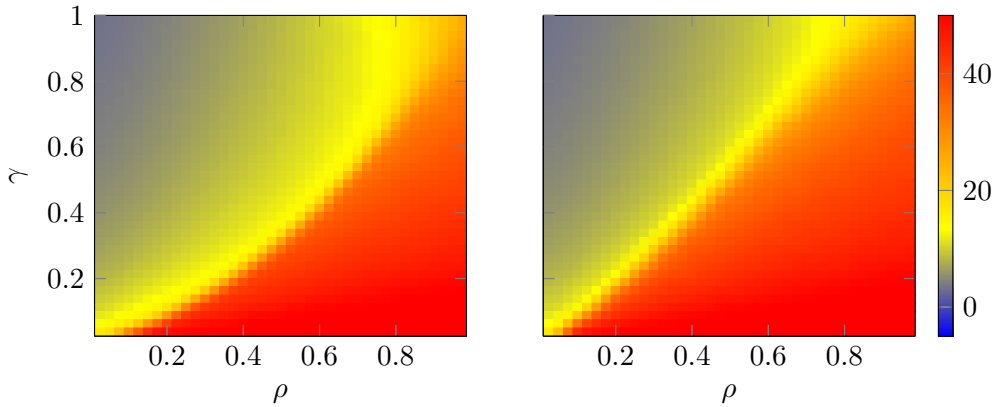


Figure 5.15: Phase transitions of MD-BAMP for various priors. The SDR of a particular algorithm is larger than 20dB southeast of the associated curve.



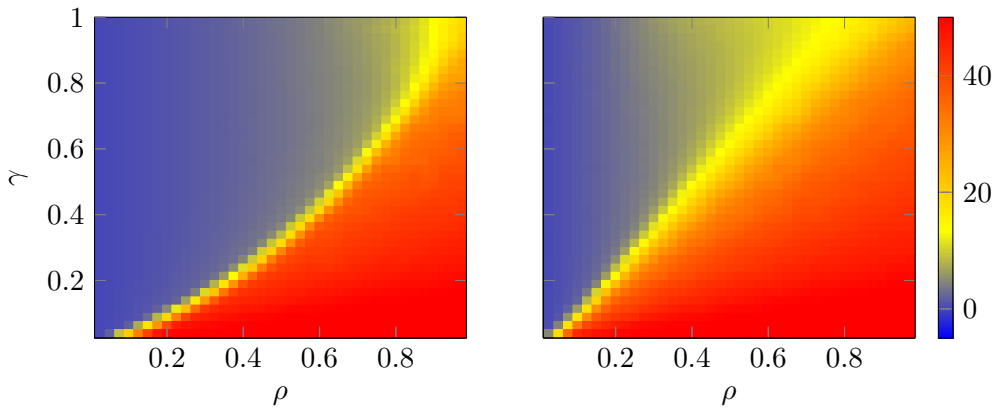
(a) Bernoulli-Gaussian, $K = 1$

(b) Bernoulli-Gaussian, $K = 4$



(c) Bernoulli-Exponential, $K = 1$

(d) Bernoulli-Exponential, $K = 4$



(e) Bernoulli-Uniform, $K = 1$

(f) Bernoulli-Uniform, $K = 4$

Figure 5.16: Performance of MD-BAMP in terms of SDR (color coded) vs. sparsity γ and subsampling ratio ρ for various priors and prior dimension K.

5.5 Discrete Priors

Purely discrete priors have applications in coding theory and data transmissions systems. An example of a practical problem with a purely discrete prior can be found in Chapter 4, with results in Section 5.6. Other examples are detection in multi-user systems [8], sparse regression codes [52] and OFDM crest factor reduction [30].
Numerical simulations for signals drawn from two discrete distributions are given in this section. One is a simple binary distribution while the other one is derived from a parity check code with parameters $(N_c, K_u) = (5, 2)$ and generator polynomial $g(x) = x^3 + x + 1$ on GF(2). Their pdfs are given by

$$f_{\mathsf{x},\mathrm{Bi}}(x) = \frac{1}{2}\left(\delta(x-1) + \delta(x+1)\right) \tag{5.20}$$

$$f_{\mathbf{x},\mathrm{P3}}(\boldsymbol{x}) = \frac{1}{4}(\delta(\boldsymbol{x} - (1,1,1,1,1)) + \tag{5.21}$$

$$\delta(\boldsymbol{x} - (-1,-1,1,-1,1)) + \tag{5.22}$$

$$\delta(\boldsymbol{x} - (1,-1,-1,1,-1)) + \tag{5.23}$$

$$\delta(\boldsymbol{x} - (-1,1,-1,-1,-1))) \,. \tag{5.24}$$

The achievable subsampling ratio ρ for purely discrete mixtures has been observed to be $\rho = 1/2$ for binary distributions [43] such as (5.20). This limit applies to BAMP as well as algorithms based on linear programming [67] and is highly sub-optimal: the Rényi information dimension $d(\mathsf{x}) = 0$ for purely discrete mixtures and thus, the minimum rate for which error-free recovery should be possible is $\rho = 0$ (i.e. $L = 1$ for $N \to \infty$). Exploiting the structure of high-dimensional distributions such as the parity check code described by $f_{\mathbf{x},\mathrm{P3}}(\boldsymbol{x})$ yields better results than the binary prior $f_{\mathsf{x},\mathrm{Bi}}(x)$, as can be seen in Fig. 5.17. The "Exit" plot associated with $f_{\mathsf{x},\mathrm{Bi}}(x)$, obtained via state evolution, is shown in Fig. 5.18. Contrary to distributions with a continuous component, where the variance at the output of the denoiser tends to become zero only as the variance at its input becomes vanishingly small, denoisers of purely discrete distributions exhibit output signal variance σ_{x}^2 which tends to zero even for finite input variance $\sigma_{\mathsf{x}}^{2(l)}$. Such distributions are therefore not sensitive to noise up to a certain level.
Using a Bayesian denoiser obtained as per Chapter 3 appears to result in an achievable subsampling region $\rho > 1/2 H_2(\mathsf{x})$. The position of the phase transition is not influenced by the size N of the unknown vector, although larger N realizes a slightly steeper "waterfall" region. Discrete-continuous mixtures with multiple Dirac components similarly result in a sub-optimal threshold, an example of which has been shown in [67], where a so-called "simple" signal is investigated.
Such "simple" signals occur in infrared absorption spectroscopy and can be modeled as the mixture of a continuous pdf $f_{\mathsf{x},c}(x)$ with support on the unit interval $[0, 1]$ and two Dirac deltas at zero and one respectively, such that

$$f_{\mathsf{x},\mathrm{simple}}(x) = \frac{1}{2}(1-\gamma)\left(\delta(x) + \delta(x-1)\right) + \gamma f_{\mathsf{x},c}(x) \,. \tag{5.25}$$

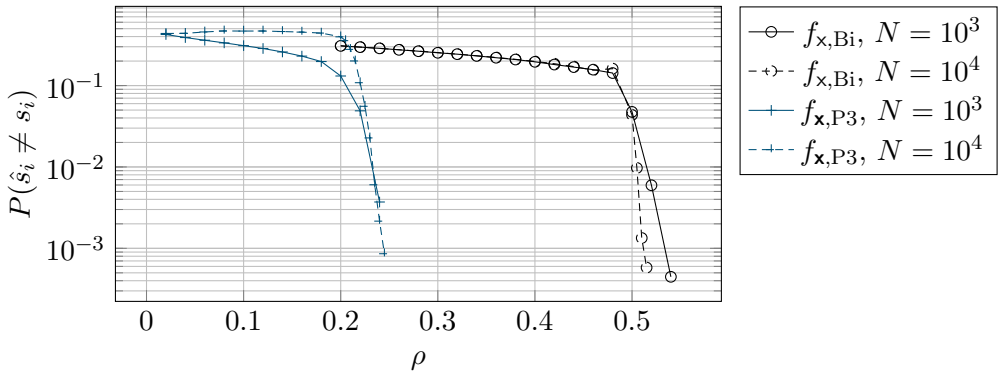


Figure 5.17: Phase transition in terms of symbol error probability vs. subsampling ratio ρ of MD-BAMP for discrete mixtures.

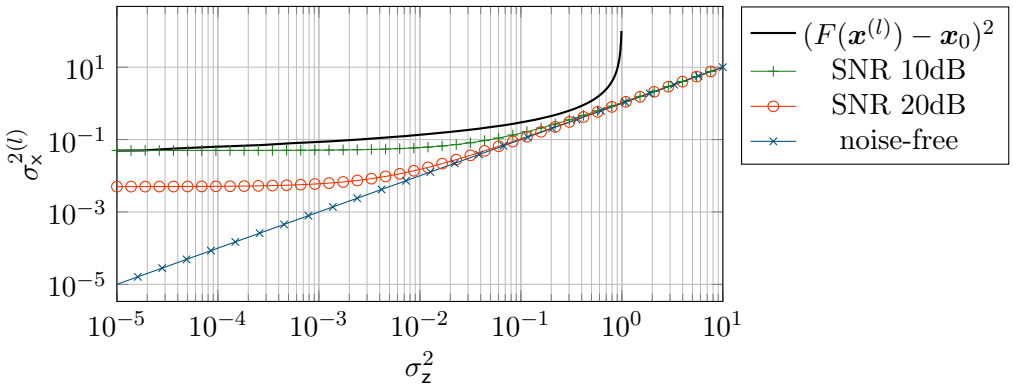


Figure 5.18: "Exit" plot of a binary prior at subsampling ratio $\rho = 1$. The noise level is computed in terms of E_b/N_0.

The phase transition for noiseless compressed sensing and recovery algorithms based on linear programming has been shown to be

$$\rho = \frac{\gamma + 1}{2} \tag{5.26}$$

for the simple signal prior [67, 26]. For $\gamma = 0$, the prior results in a binary signal and the phase transition becomes $\rho = 1/2$. There are several limitations which prevent any algorithm from achieving perfect recovery close to $L = 1$ in practice. In systems where the vector $\boldsymbol{y}$ is the result of measurements of physical quantities, the subsampling ratio is limited by the resolution of the analog-to-digital (A/D) converter. Furthermore, all recovery algorithms are limited by the choice of digital number representation. Both limitations arise from the fact that the number of bits used to represent the measurement vector $\boldsymbol{y}$ needs to be at least as large as the number of information bits one is trying to recover. The popular double precision floating point type occupies 8 bytes of memory and can therefore (due to reserved

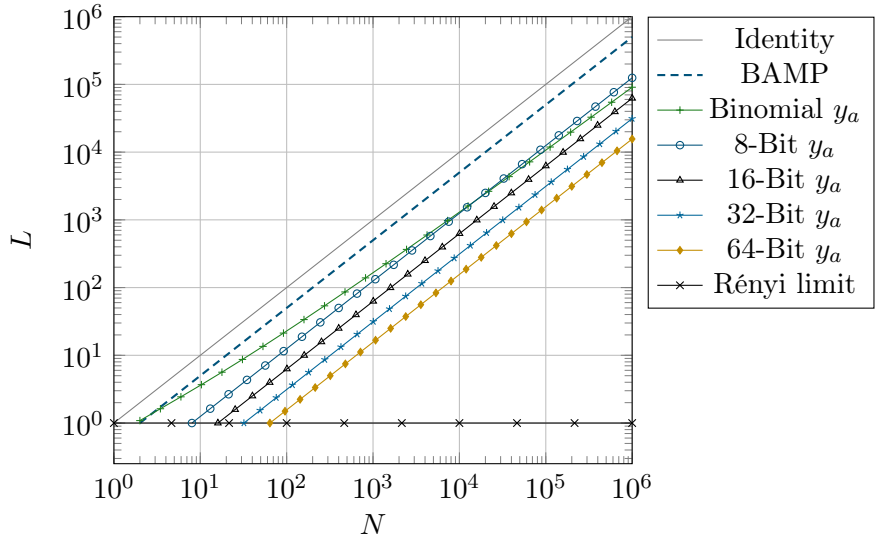


Figure 5.19: Minimum number of samples L necessary for binary signals of length N under various conditions.

special bit patterns) represent less than 2^{64} different numerical values. To give a concrete example, when using double precision floating point numbers, at least $L \geq 16$ samples ($= 1024$ bit) are required to be able to store a signal of length $N = 10^3$ sampled from an i.i.d. binary prior such as (5.20). Further restrictions can be due to the type of measurement matrix. Using Rademacher matrices (whose entries are i.i.d. and $A_{a,i} \in \{-1, 1\}$) for measurement of binary signals results in binomial distributed entries y_a with distribution parameters $n = N+1$ and $p = 1/2$. The Shannon entropy of the binomial distribution is given by

$$H_2(\mathrm{y}_{\mathrm{bin}}) = \frac{1}{2}(\log_2(\pi e(N+1)) - 1) + O(1/(N+1)) \,. \tag{5.27}$$

Assuming that the measurements y_a are independent, the minimum number of samples is

$$L \gtrapprox \frac{2N}{\log_2(\pi e(N+1)) - 1} \,. \tag{5.28}$$

Limits for various practical setups are plotted in Fig. 5.19. Perfect recovery is impossible below the given value for a selected (combination of) conditions. For example, any algorithm employing an 8-Bit representation of y_a for compressive sensing recovery of binary signals with Rademacher measurement matrices is limited by the choice of sensing matrix for $N \lessapprox 3 \cdot 10^4$, beyond this point it is limited by the selection of numerical representation (or possibly the resolution of the A/D converter). Note that approaching the Rényi limit of $L = 1$ requires either small

problem dimension ($N < 64$) or extremely precise digital number representation. These limits apply to noiseless compressive sensing. Noisy measurements are furthermore limited due to finite channel capacity. Note that the performance of BAMP (or, to my knowledge, any algorithm with polynomial complexity that is applicable to Gaussian measurement matrices) is not approaching these practical limits, however.

5.6 Compressed LDPC Codes

In this section, the behavior and performance of MD-BAMP as joint decompressor and detector for compressed LDPC codes is analyzed. Since state evolution is not usable due to the denoiser being stateful, numerical simulations were employed. The codes used for simulations are binary regular LDPC codes derived from Euclidean geometries [38]. These codes combine several advantages, one of them being that they are cyclic and it is therefore easy to realize an encoder. Their properties are listed in table 5.1, where N_c and K_u are the code word and the user data word length respectively, while N is the size of K concatenated code words and R is the rate of the (uncompressed) code. The last column lists the smallest ratio of energy per bit vs. noise power spectral density for the given rate on an additive, white Gaussian noise (AWGN) channel. The number of samples (i.e. simulated code word transmissions) was chosen such that at least 10^6 symbols are computed per parameter combination.

The code word lengths N_c of the codes examined in this section range from 15 to 4095 and are thus relatively short[1]. Even at these block lengths and when concatenating a small number of code words (cf. (4.3)), the size of the matrix $\boldsymbol{A}$ becomes large. For simulations, the Fast Hadamard Transform (FHT) was therefore used instead of a matrix with Gaussian entries. Compression is achieved by discarding all but a random subset of rows of the transform's output vector. This is equivalent to populating the sensing matrix $\boldsymbol{A}$ with randomly selected rows from the associated Walsh-Hadamard transform matrix[2]. This process was also used by Rush et al. in [52]. The complexity of a matrix-vector multiplication is then $O(N\log(N))$ instead of $O(NL)$. A limited subset of results was presented at the 19$^{\text{th}}$ Joint Workshop on Communications and Coding [11]. Simulation results in this section have been obtained by using an updated implementation and aim for better accuracy.

Using the denoiser for LDPC codes as derived in Chapter 4 together with MD-BAMP results in a "Turbo" structure[3]. Detection is performed iteratively. Each iteration can be split into two components: in a first step, the inner transmit symbols $\boldsymbol{x}$ are estimated using the sensing matrix $\boldsymbol{A}$ and the received symbols $\boldsymbol{y}$. The estimation is performed independently for each received symbol with the aim of minimizing the individual squared estimation error $\varepsilon^2 = \mathbb{E}\left\{(y_a - \boldsymbol{A}_a\boldsymbol{x})^2\right\}$ (cf. (2.31)) under the assumption of Gaussian noise. Due to several approximations, this operation is realized as a matrix-vector multiplication (cf. line 5 in Algorithm 8, p.114). This step neglects constraints such as the limited symbol alphabet of $\boldsymbol{x}$ and the structure of the code. In a second step, the denoiser (which is highly

[1] This is due to limited computational resources available for simulations.

[2] Note that normalization is different from the FHT, which assumes a square matrix. Furthermore, the first column (and row) of the Walsh-Hadamard matrix is $\mathbf{1}$ and should not be used in the sensing matrix.

[3] The structure is identical for all priors, however we feel that the term "Turbo" is more justified when the application is a detector.

short name	N_c	K_u	R	N	K	E_b/N_0 at Capacity
m2s2	15	7	0.467	10050	670	-0.111dB
m2s3	63	37	0.587	10017	159	0.296dB
m2s4	255	175	0.686	10200	40	0.637dB
m2s5	1023	781	0.763	10230	10	0.763dB
m2s6	4095	3367	0.822	40950	10	1.116dB
m3s2	63	13	0.206	10017	159	-0.956dB
m3s3	511	139	0.272	10220	20	-0.747dB
m3s4	4095	1377	0.336	40950	10	-0.540dB

Table 5.1: Properties of codes and simulation parameters.

similar to the LDPC sum-product detector) improves upon the estimation $\boldsymbol{\mu}_{\mathsf{x}}^{(l)}$ at its input. Its output, the vector $\boldsymbol{\mu}_{\mathsf{x}}$ of estimated inner transmit symbols, is then used to compute a new residual $\boldsymbol{\mu}_{\mathsf{z}}$, which is in turn passed back to compute a new estimate of $\boldsymbol{\mu}_{\mathsf{x}}^{(l)}$ aiming to minimize the estimation error ε. The algorithm iterates until it converges or aborts after a certain number of steps.
The performance of the joint system depends on the SNR at the input of the denoiser. For uncompressed LDPC codes using fixed-power binary signaling on an AWGN channel, the SNR E_b/N_0 is determined by the channel noise spectral density. To simplify the following analysis, let the symbol energy $E_x = 1$. In the case of uncompressed binary signaling, the energy per bit given by an (N_c, K_u) binary code with rate $R_c = K_u/N_c$ is $E_b = R_c^{-1}$. For the compressed system, the energy of a transmit symbol vector $\boldsymbol{y}^{(tx)}$ is obtained by compressing K code words, thus

$$E_{\mathbf{y}^{(tx)}} = \mathbb{E}\left\{\left\|\mathbf{y}^{(tx)}\right\|_2^2\right\} = L\frac{N}{L}\underbrace{\sigma_{\mathsf{x}}^2}_{=1} = N = KN_c\,. \tag{5.29}$$

Such a transmit symbol vector $\boldsymbol{y}$ contains $K \cdot K_u$ information (user) symbols (bits), and the average energy per information bit E_b is therefore

$$E_b = \frac{KN_c}{KK_u} = \frac{N_c}{K_u} = \frac{1}{R_c}\,, \tag{5.30}$$

which does not depend on the sampling ratio $\rho = {}^{L}/_{N}$ and is identical to the energy per bit in the uncompressed system. The noise at the input of the denoiser is $\sigma_{\mathsf{x}}^{2(l)}$, which is (cf. (2.122))

$$\sigma_{\mathsf{x}}^{2(l)} = \frac{N}{L}\sigma_{\mathsf{x}}^2 + \sigma_{\mathsf{w}}^2\,, \tag{5.31}$$

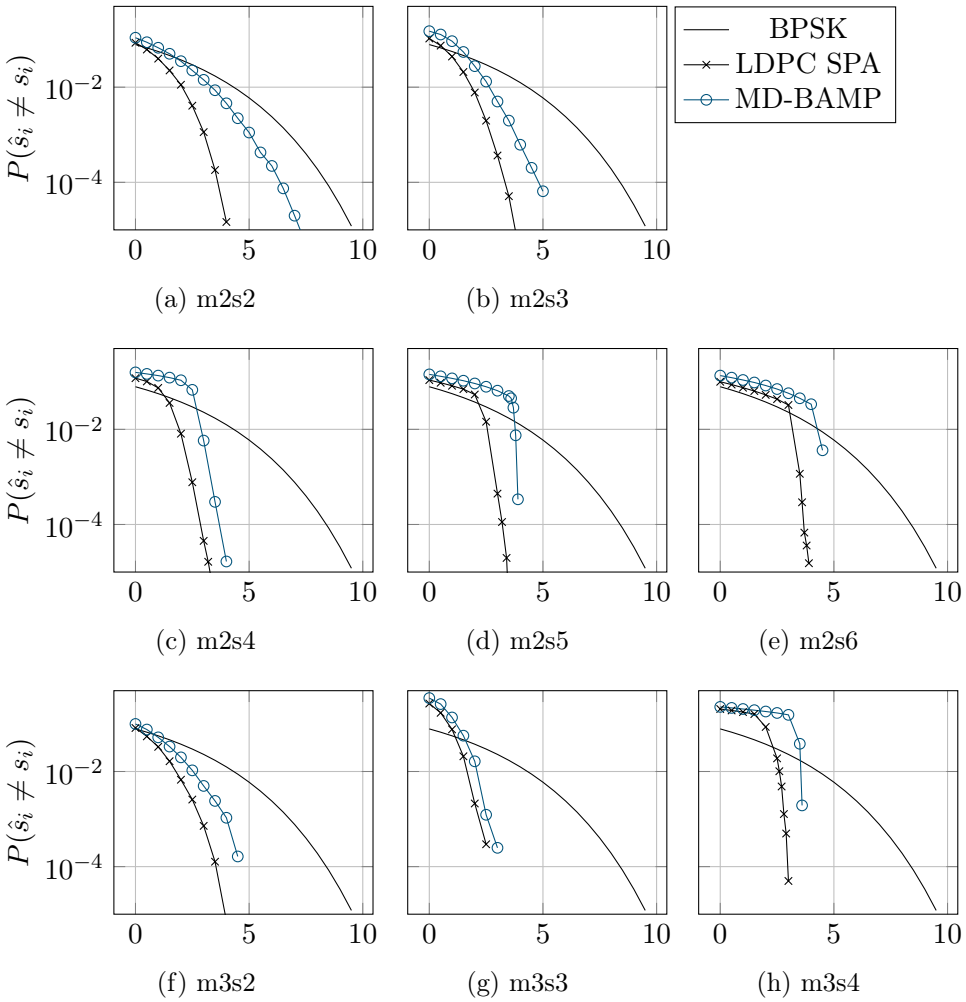
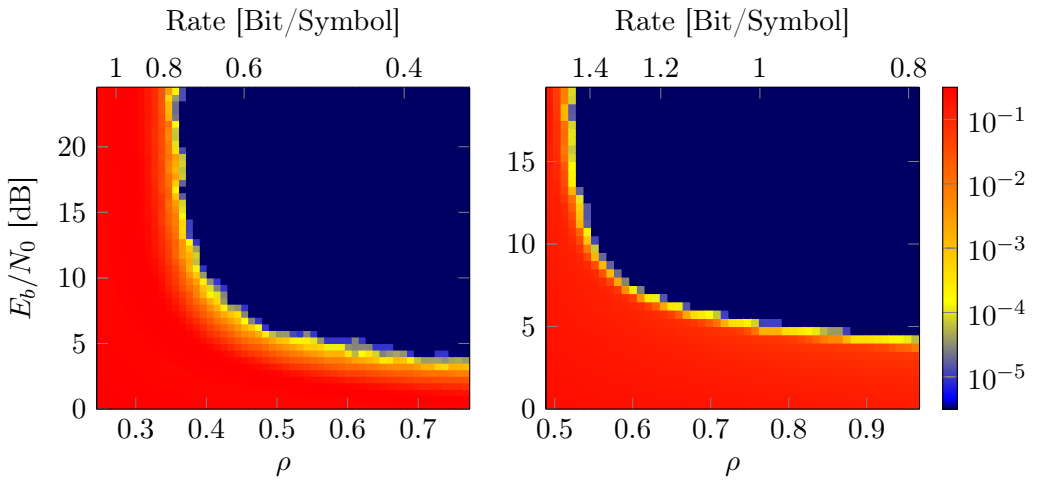


Figure 5.20: Comparison of the LDPC sum-product detector and MD-BAMP in the uncompressed case in terms of symbol error ratio vs. E_b/N_0.

where σ_{x}^2 is the average variance at the output of the denoiser and σ_{w}^2 the noise variance. At the input of the denoiser, the SNR is therefore strictly larger than for the uncompressed system, where the noise variance at the detector is σ_{w}^2. If MD-BAMP converges towards the true vector $\boldsymbol{x}$, the variance σ_{x}^2 at the output of the denoiser becomes zero and the SNR is identical to the uncompressed system. Initially, the SNR is larger however, thus the denoiser can be expected to perform inferior to the detector in an uncompressed system and thus, the joint decompressor/detector realized using MD-BAMP can be expected to have inferior performance compared to the uncompressed system. Note that this is true even for systems with $\rho \geq 1$. This is indicative of the system not achieving a coding gain.

The performance of the LDPC sum-product algorithm (LDPC SPA) is shown in



(a) m3s3 (b) m2s5

Figure 5.21: Symbol error ratio (color coded) vs. subsampling ratio ρ (bottom axis) and SNR using MD-BAMP as joint decompressor and detector. The rate in bit per transmit symbol is given in the (nonlinear) top axis. Note that the two plots cover different E_b/N_0 and subsampling regions. The color coding is identical for both plots. Dark blue regions indicate zero encountered symbol errors.

Fig. 5.20, where it is also compared with BAMP at subsampling ratio $\rho = 1$. For these simulations, the SNR E_b/N_0 was incremented in steps of 0.5dB in general, with closer intervals in the "waterfall" region of selected codes. All simulations range to at least 8dB E_b/N_0. For any parts not plotted, no symbol errors occurred. The symbol error probability of the LDPC SPA simulations replicate results in [38] (the associated graphs are labeled "EG-LDPC SPA bit"). MD-BAMP exhibits inferior performance by about 1-1.5dB for codes of length $N_c \geq 255$.

In Fig. 5.21, the performance in terms of symbol error ratio vs. subsampling ratio and SNR E_b/N_0 is shown. The m3s3 code in Fig. 5.21a is representative for the performance for low-rate codes. For these codes, it is possible to maintain a small symbol error probability even at a small subsampling ratio ρ. Due to the small code rate, the effective rate in bit per symbol is limited to ~ 0.85. For high-rate codes such as m2s5, which is shown in Fig. 5.21b, the smallest achievable subsampling ratio ρ is comparatively large. Due to the intrinsically high rate, even moderate compression results in a feasible effective rate in bit per symbol. Thus, the m2s5 code can operate at a rate of up to ~ 1.4 bit per symbol. Note that MD-BAMP operating with a denoiser for i.i.d. binary symbols maintains low symbol error probability (i.e. $P(\hat{s}_i \neq s_i) < 10^{-5}$) for rates $\rho > 0.5$ (equivalent to 2 bit per symbol) and $E_b/N_0 \gtrapprox 10.5$dB. Therefore, this region of operation can easily be realized.

It remains to find codes outperforming MD-BAMP in conjunction with an i.i.d.

binary prior. Such codes maintain a small symbol error probability for either $E_b/N_0 \lessapprox 10.5$dB or rates > 2 bit per symbol. In Fig. 5.22, the transitions from large to small symbol error probability are plotted. While the low-rate codes from the m3-family enable subsampling ratio $\rho < 0.5$, their effective rate remains small. Codes with large block length and high rate are limited to subsampling ratio $\rho > 0.5$ but achieve effective rates of up to ~ 1.6 bit per symbol. None of the codes achieve the rate of two bit per symbol attained by MD-BAMP with binary pior. Therefore, only regions with small symbol error probability at an SNR of $E_b/N_0 \lessapprox 10.5$dB provide superior performance compared to compressed i.i.d. binary signals. The highest rate in this region is realized using the m2s6 code, which outperforms binary signaling up to a rate of ~ 1.5 bit per symbol.

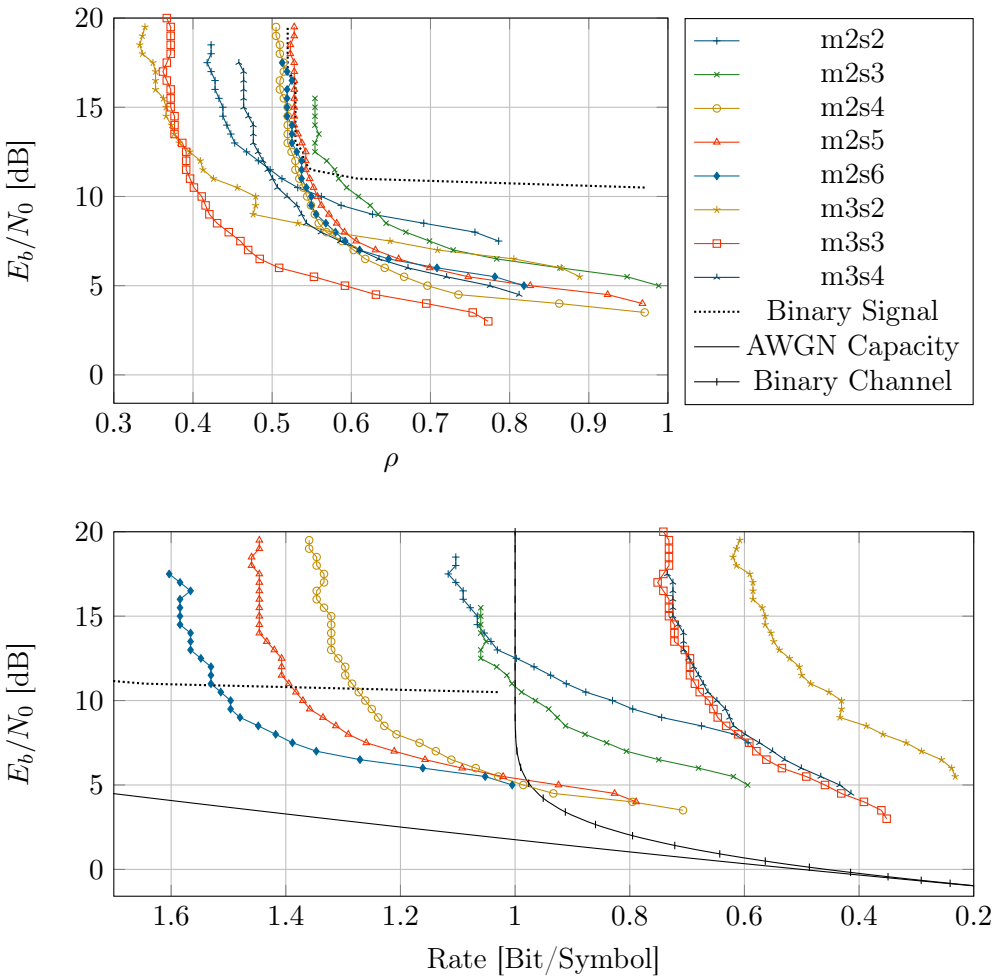


Figure 5.22: Achievable combinations of subsampling ratio (top), code rate (bottom) vs. SNR E_b/N_0. The symbol error rate of a particular code is smaller than 10^{-5} northeast of the associated curve.

Chapter 6

Conclusions and Future Work

Research into the topic of compressive sensing given multi-dimensional signal priors was initially motivated by the desire to obtain a BAMP algorithm for compressive sensing given complex valued measurements. It quickly became clear that pursuing this path would needlessly limit the scope of possibilities. On the one hand, papers covering approximate message passing for complex-valued signals had already been published [45]. On the other hand, the complex case stays simple for the case of independent real and imaginary parts of complex entries x_i only. Given dependencies between real and imaginary parts, a two-dimensional denoiser is required, even if samples of the complex signal $\boldsymbol{x}$ are otherwise independent.
This realization motivated the derivation of an algorithm that would allow dependencies between an arbitrary number of real-valued entries of $\boldsymbol{x}$. Even though this work does not offer a formulation for a recovery algorithm using complex-valued variables, the presented methods are nonetheless applicable, using suitable mappings to real numbers.
Detailed knowledge of the Bayesian approximate message passing algorithm enabled further work on other problems. While the option of in some form non-i.i.d. noise is usually not considered in compressive sensing, it becomes relevant given the extended range of applications that recovery algorithms are now applied to, which include coded transmission systems, user activity detection in MIMO scenarios and more. The case of nonzero-mean sensing matrices had already been the topic of several publications. These provided valuable clues leading towards an algorithm promising optimal recovery for (almost) any matrix mean.

Opportunities

While widening the applicability of BAMP-based algorithms towards more general inverse problems, this work leaves many questions unanswered. For example, numerical evidence as well as state evolution seem to suggest that making use of detailed knowledge of the signal, represented by high-dimensional priors, results in recovery performance approaching the Rényi limit. Efficient algorithms oper-

ating at or near this limit consistently for any type of signal are so far unknown, although it has been shown that special measurement matrices can improve performance [39]. Modifying the matrix statistics (apart from its mean) requires careful adaptation of the BAMP algorithm, which generally expects the matrix entries to be i.i.d. It is possible to fall back to GMP, whose computational complexity makes it unattractive for practical applications, however.

It would be desirable to find near-optimal algorithms that merely require scaling of the rows (or columns) of the sensing matrix. The scaling factors can then be hidden in the signal prior or by using appropriate transformations. Note that modifications of this kind can lead to diverging variances, i.e. it might be necessary to track more than one variance value. Not doing this might yield inferior results, comparable to using a single average noise variance when more detailed information is available.

Rush et al. achieve channel capacity with sparse regression codes (SPARCs) [52] using suitably adjusted symbol amplitudes. Unfortunately, the structure of SPARCs incurs computational complexity that is exponential in the number of bits per code word. It would be desirable to achieve a similar result using more powerful underlying codes, such as LDPC codes.

Other applications include compressed imaging systems. Compressive sensing techniques have already been successfully applied to single-pixel sensing systems at visible wavelengths [27] and in the terahertz region [18]. A variety of techniques has been developed, most of which recover the image in the wavelet domain. This can be done with simple schemes like soft thresholding or by making use of the wavelet coefficient's structure [55]. One of the best-performing algorithms uses a classical image denoising method ("non-local means denoising"), suitably adapted for use with AMP [46]. One of the difficulties of integrating a general denoiser with AMP is the requirement to know the derivative as well as the variance at the output of the denoiser. Note that these techniques all deal with monochromatic images. The field of multi-spectral compressive imaging offers interesting challenges, however. While it seems obvious to use multidimensional priors to make use of correlation between spectral components, preliminary experiments using methods presented in this thesis have shown that on its own, this is insufficient to approach the quality offered by recovery algorithms profiting from the (apparent) sparsity of an image in the wavelet domain, done independently per spectral component. A system combining the advantages of both techniques could allow for higher compression rates than currently achieved by CS recovery algorithms.

Many problems might profit from an approach less focused on exact statistical signal properties and more on integrating successful methods from other fields of research. While not yet popular, using on-line learning algorithms for "blind" recovery (i.e. without knowledge of the signal prior) have shown to yield excellent results [61] for i.i.d. unknown x_i. Applying the same strategy to high-dimensional priors is more challenging, due to the much larger number of variables that need to be estimated. For high-dimensional problems it might therefore be advantageous to

integrate machine-learning algorithms, which has been done successfully for OFDM channel estimation [68].
Of practical importance is also the type of sensing matrix occurring in a particular problem. It is desirable to employ matrices that allow for fast (i.e. $O(N \log(N))$) computation of matrix-vector multiplication. This is due to storage and computational complexity growing quadratically otherwise, which is impractical in many settings. Fortunately, linear measurements in physical systems are sometimes structured, e.g. in the case of non-destructive testing using ultrasound [53], where a Toeplitz structure was exploited. A Python library offering fast evaluation of many structured matrix operations can be found in [62]. While some of these structures might be suitable for use with BAMP right away (or with small modifications, such as Hadamard type matrices), others may exhibit correlated columns or non-i.i.d. entries. Extending BAMP in this direction is another open topic.

Acknowledgments

I would like to thank my parents, friends and colleagues for their encouragement, support and fruitful discussions. Most of all I would like to thank my advisor, Prof. Norbert Görtz, who actively supported my research.

Appendix A

Algorithms

A number of algorithms have been developed to recover a sparse signal from a small number of linear measurements. For each of these, several varieties exist. They improve certain aspects of their underlying algorithm, like the popular "FISTA" (fast iterative shrinkage-thresholding algorithm), which improves the convergence speed of ISTA [4]. Another example is orthogonal matching pursuit, which provides a more accurate recovery than its predecessor, matching pursuit. The particular variants of the algorithms used for comparison as well as those presented in this work are listed here.

A.1 Matching Pursuit

The matching pursuit [44] algorithm recovers a vector $\boldsymbol{x}$ from measurements $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$. In each iteration, the algorithm determines which of the non-selected columns of $\boldsymbol{A}$ best match the residual, adds this column to the selected columns and updates the estimate of $\boldsymbol{x}$ as well as the residual $\boldsymbol{z}$. The algorithm stops when the change in the estimated vector $\hat{\boldsymbol{x}}$ drops below a threshold ε or when a predefined number γ of nonzero entries of $\boldsymbol{x}$ has been estimated. A variant of MP exists, called orthogonal matching pursuit (OMP) [47], which computes the least-squares optimal estimate for the set of selected indices.

Algorithm 1 Matching Pursuit

$\boldsymbol{z} \leftarrow \boldsymbol{y}$, $\mathcal{S} \leftarrow \{\}$.
Set constants $t_{\max}$, $\gamma \leftarrow |\{x_i \neq 0\}|$, ε.
All other variables are initialized to zero.
repeat
  $\boldsymbol{x}^{(l)} \leftarrow \boldsymbol{A}^T \boldsymbol{z}$
  $k \leftarrow \arg\max_{i \in \{1 \dots N\} \setminus \mathcal{S}} |x_i^{(l)}|$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{k\}$
  $\hat{\boldsymbol{x}}^{[t-1]} \leftarrow \boldsymbol{x}$
  $\hat{x}_k \leftarrow x_k^{(l)}$
  $\boldsymbol{z} \leftarrow \boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{x}}$
until $t > t_{\max}$ **or** $|\mathcal{S}| \geq \gamma$ **or** $t > 1$ **and** $\left\|\hat{\boldsymbol{x}} - \hat{\boldsymbol{x}}^{[t-1]}\right\|^2 < \varepsilon \left\|\hat{\boldsymbol{x}}\right\|^2$

Algorithm 2 Orthogonal Matching Pursuit

$\boldsymbol{z} \leftarrow \boldsymbol{y}$, $\mathcal{S} = \{\}$.
Set constants $t_{\max}$, $\gamma = |\{x_i \neq 0\}|$, ε.
All other variables are initialized to zero.
repeat
  $\boldsymbol{x}^{(l)} \leftarrow \boldsymbol{A}^T \boldsymbol{z}$
  $k \leftarrow \arg\max_{i \in \{1 \dots N\} \setminus \mathcal{S}} |x_i^{(l)}|$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{k\}$
  $\boldsymbol{B}_{:,j} \leftarrow \boldsymbol{A}_{:,i}$, $i \in \mathcal{S}$, $j : 1 \dots |\mathcal{S}|$
  $\boldsymbol{v} \leftarrow (\boldsymbol{B}^T \boldsymbol{B})^{-1} \boldsymbol{B}^T \boldsymbol{y}$
  $\hat{\boldsymbol{x}}^{[t-1]} \leftarrow \boldsymbol{x}$
  $\hat{x}_k \leftarrow v_j$, $j : \boldsymbol{B}_j = \boldsymbol{A}_k$
  $\boldsymbol{z} \leftarrow \boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{x}}$
until $t > t_{\max}$ **or** $|\mathcal{S}| \geq \gamma$ **or** $t > 1$ **and** $\left\|\hat{\boldsymbol{x}} - \hat{\boldsymbol{x}}^{[t-1]}\right\|^2 < \varepsilon \left\|\hat{\boldsymbol{x}}\right\|^2$

A.2 Iterative Thresholding

Two variants of iterative thresholding exist. Iterative hard thresholding (IHT) [12, 13] uses a hard thresholding function $\eta(x_i^{(l)})$ to estimate $\boldsymbol{x}$ from the projection $\boldsymbol{x}^{(l)} = \boldsymbol{A}^T\boldsymbol{z}$. The thresholding function is defined as

$$\eta(x_i^{(l)}, \tau) = I(|x_i^{(l)}| > \tau) \cdot x_i^{(l)}, \tag{A.1}$$

with $I(x)$ an indicator function which evaluates to one if the expression x is true, otherwise it evaluates to zero. If the number γ of nonzero entries of $\boldsymbol{x}$ is known, the threshold s can be chosen such that the largest γ entries are selected. The entries of the matrix are assumed to be i.i.d. and distributed according to $\mathcal{N}(0, L^{-1})$. To maintain stability of the algorithm, it is necessary to compensate by $\boldsymbol{A}$'s largest singular value $s_{\max}$. This can either be computed or approximated using Bai-Yin's law [59]. The matrix is assumed to have size $L \times N$. Often, a tuning factor α is used. The iterative soft thresholding algorithm [21] is conceptually identical to iterative hard thresholding, but a soft thresholding function is used:

$$\eta(x_i^{(l)}, \tau) = I(|x_i^{(l)}| > \tau) \cdot (|x_i^{(l)}| - \tau) \cdot \operatorname{sign}(x_i^{(l)}), \tag{A.2}$$

with the indicator function $I(x)$ defined as before and $\operatorname{sign}(x)$ returns the sign of its argument, i.e. $+1$ if it is positive, 0 if it is zero and -1 otherwise.

Algorithm 3 Iterative Thresholding

$\boldsymbol{z} \leftarrow \boldsymbol{y}$
Set constants $t_{\max}$, τ, α, ε, $s_{\max} \leftarrow 1 + \sqrt{N/L}$.
All other variables are initialized to zero.
repeat
  $\boldsymbol{x}^{(l)} \leftarrow \alpha / s_{\max}^2 \boldsymbol{A}\boldsymbol{z} + \hat{\boldsymbol{x}}$
  $\hat{\boldsymbol{x}}^{[t-1]} \leftarrow \boldsymbol{x}$
  $\hat{x}_i \leftarrow \eta(x_i^{(l)}, \tau),\ i : 1 \dots N$
  $\boldsymbol{z} \leftarrow \boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{x}}$
until $t > t_{\max}$ **or** $t > 1$ **and** $\left\|\hat{\boldsymbol{x}} - \hat{\boldsymbol{x}}^{[t-1]}\right\|^2 < \varepsilon \left\|\hat{\boldsymbol{x}}\right\|^2$

A.3 Approximate Message Passing

The approximate message passing (AMP) algorithm is similar to soft thresholding. Approximate message passing adds the "Onsager-Term" $\frac{z}{L}\|\hat{\boldsymbol{x}}\|_0$. The term is originates from statistical physics [14] where it is used in a similar fashion, i.e. for a term that serves as first-order correction. Furthermore, AMP also employs a different method of setting the threshold of the soft thresholding function. Despite their similarity, the two algorithms have different origins. Iterative soft thresholding was derived by minimizing a particular cost function while AMP is an approximation of the sum-product algorithm applied to a graph representing the compressive sensing setting. In Algorithm 4, σ_w^2 is the noise variance and σ_x^2 the variance of any entry of $\boldsymbol{x}$. Bayesian approximate message passing uses a denoiser function called F instead of the soft thresholding function but is otherwise similar to AMP. An additional function G is used to estimate the variance at the output of the denoiser F.

Algorithm 4 Approximate Message Passing

$\boldsymbol{z} \leftarrow \boldsymbol{y}$
Set constants $t_{\max}$, ε, σ_w^2, $\tau^2 = \sigma_w^2 + \frac{N}{L}\sigma_x^2$.
All other variables are initialized to zero.
repeat
  $\boldsymbol{x}^{(l)} \leftarrow \boldsymbol{A}\boldsymbol{z} + \hat{\boldsymbol{x}}$
  $\hat{\boldsymbol{x}}^{[t-1]} \leftarrow \boldsymbol{x}$
  $\hat{x}_i \leftarrow \eta(x_i^{(l)}, \tau),\ i : 1 \dots N$
  $\boldsymbol{z} \leftarrow \boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{x}} + \frac{z}{L}\|\hat{\boldsymbol{x}}\|_0$
  $\tau^2 \leftarrow \sigma_w^2 + \frac{\tau^2}{L}\|\hat{\boldsymbol{x}}\|_0$
until $t > t_{\max}$ **or** $t > 1$ **and** $\left\|\hat{\boldsymbol{x}} - \hat{\boldsymbol{x}}^{[t-1]}\right\|^2 < \varepsilon \|\hat{\boldsymbol{x}}\|^2$

Algorithm 5 Bayesian Approximate Message Passing

$\boldsymbol{z} \leftarrow \boldsymbol{y}$
Set constants $t_{\max}$, ε, σ_w^2, $\sigma_x^{2(l)} = \sigma_w^2 + \frac{N}{L}\sigma_x^2$.
All other variables are initialized to zero.
repeat
  $\boldsymbol{x}^{(l)} \leftarrow \boldsymbol{A}\boldsymbol{z} + \hat{\boldsymbol{x}}$
  $\hat{\boldsymbol{x}}^{[t-1]} \leftarrow \boldsymbol{x}$
  $\hat{x}_i \leftarrow F(x_i^{(l)}, \sigma_x^{2(l)}),\ i : 1 \dots N$
  $\boldsymbol{z} \leftarrow \boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{x}} + \frac{z}{L}\sum_i \frac{\partial F(x_i^{(l)}, \sigma_x^{2(l)})}{\partial x_i^{(l)}}$
  $\sigma_x^{2(l)} \leftarrow \sigma_w^2 + \frac{N}{L}\sum_i G(x_i^{(l)}, \sigma_x^{2(l)})$
until $t > t_{\max}$ **or** $t > 1$ **and** $\left\|\hat{\boldsymbol{x}} - \hat{\boldsymbol{x}}^{[t-1]}\right\|^2 < \varepsilon \|\hat{\boldsymbol{x}}\|^2$

A.4 Message Passing for Multi-Dimensional Priors

Two variants of message passing for multi-dimensional priors exist, which are presented in Section 2.5. Gaussian message passing, listed as Algorithm 6, uses Gaussian approximations of messages between factor and variable nodes while Bayesian approximate message passing (Algorithm 7) computes updates of local states, using a correction term to compensate for non-targeted messages. A simplified version of MD-BAMP, which only requires the diagonal of the Jacobian of F, is listed as Algorithm 8.

Algorithm 6 MD-GMP

$\sigma^2_{\mathsf{x},a\to i} \leftarrow \frac{\|\boldsymbol{y}\|^2}{LA^2_{a,i}}$, $\mu_{\mathsf{x},a\to i} \leftarrow y_a$
Set constants $\sigma^2_{\mathsf{w}_a}$, $t_{\max}$, ε, $A_{a,i}$.
All other variables are initialized to zero.
repeat
  $\sigma^{2(l)}_{\mathsf{x},i} \leftarrow \left(\sum_a \frac{1}{\sigma^2_{\mathsf{x},a\to i}}\right)^{-1}$
  $\mu^{(l)}_{\mathsf{x},i} \leftarrow \sigma^2_{\mathsf{x},i} \sum_a \frac{\mu_{\mathsf{x},a\to i}}{\sigma^2_{\mathsf{x},a\to i}}$
  $\boldsymbol{\mu}^{(l)}_{\mathbf{x},k} \leftarrow \left(\dots, \mu^{(l)}_{\mathsf{x},i}, \dots\right)^T \quad i \in \mathcal{K}$
  $\boldsymbol{\Sigma}^{(l)\mathrm{diag}}_{\mathbf{x},k} \leftarrow \left(\dots, \sigma^{2(l)}_{\mathsf{x},i}, \dots\right)^T \quad i \in \mathcal{K}$
  $\mu^{[t-1]}_{\mathsf{x},i} \leftarrow \mu_{\mathsf{x},i}$
  $\mu_{\mathsf{x},i} \leftarrow F_i(\boldsymbol{\mu}^{(l)}_{\mathbf{x},k}, \boldsymbol{\Sigma}^{(l)}_{\mathbf{x},k})$
  $\sigma^{2(l)}_{\mathsf{x},i\to a} \leftarrow \left(\sum_{b\neq a} \frac{1}{\sigma^2_{\mathsf{x},b\to i}}\right)^{-1}$
  $\mu^{(l)}_{\mathsf{x},i\to a} \leftarrow \sigma^{2(l)}_{\mathsf{x},i\to a} \sum_{b\neq a} \frac{\mu_{\mathsf{x},b\to i}}{\sigma^2_{\mathsf{x},b\to i}}$
  $\boldsymbol{\mu}^{(l)}_{\mathbf{x},k\to a} \leftarrow \left(\dots, \mu^{(l)}_{\mathsf{x},i\to a}, \dots\right)^T \quad i \in \mathcal{K}$
  $\boldsymbol{\Sigma}^{(l)\mathrm{diag}}_{\mathbf{x},k\to a} \leftarrow \left(\dots, \sigma^{2(l)}_{\mathsf{x},i\to a}, \dots\right)^T \quad i \in \mathcal{K}$
  $\mu_{\mathsf{x},i\to a} \leftarrow F_i(\boldsymbol{\mu}^{(l)}_{\mathbf{x},k\to a}, \boldsymbol{\Sigma}^{(l)}_{\mathbf{x},k\to a})$
  $\sigma^2_{\mathsf{x},i\to a} \leftarrow G_i(\boldsymbol{\mu}^{(l)}_{\mathbf{x},k\to a}, \boldsymbol{\Sigma}^{(l)}_{\mathbf{x},k\to a})$
  $\mu_{\mathsf{x},a\to i} \leftarrow \left(y_a - \sum_{j\neq i} A_{a,j}\mu_{\mathsf{x},j\to a}\right)\frac{1}{A_{a,i}}$
  $\sigma^2_{\mathsf{x},a\to i} \leftarrow \left(\sigma^2_{\mathsf{w}_a} + \sum_{j\neq i} A_{a,j}\sigma^2_{\mathsf{x},j\to a}\right)\frac{1}{A^2_{a,i}}$
until $t > t_{\max}$ **or** $t > 1$ **and** $\sum_i \left|\mu_{\mathsf{x},i} - \mu^{[t-1]}_{\mathsf{x},i}\right|^2 < \varepsilon \left\|\boldsymbol{\mu}_{\mathbf{x}}\right\|^2$
$\hat{\boldsymbol{x}} = (\dots, \mu_{\mathsf{x},i}, \dots)^T$

Algorithm 7 MD-BAMP

$\boldsymbol{\mu}_{\mathbf{z}} \leftarrow \boldsymbol{y}$, $\sigma_{\mathrm{x}}^{2(l)} \leftarrow \frac{1}{L}\|\boldsymbol{y}\|^2$
Set constants σ_{w}^2, $t_{\max}$, ε.
All other variables are initialized to zero.
repeat
  $\boldsymbol{\mu}_{\mathbf{x}}^{(l)} \leftarrow \boldsymbol{A}^T\boldsymbol{\mu}_{\mathbf{z}} + \boldsymbol{\mu}_{\mathbf{x}}$
  $\boldsymbol{\mu}_{\mathbf{x}}^{[t-1]} \leftarrow (\boldsymbol{\mu}_{\mathbf{x}_0}^T, \boldsymbol{\mu}_{\mathbf{x}_1}^T, \dots, \boldsymbol{\mu}_{\mathbf{x}_K}^T)^T$
  $\boldsymbol{\mu}_{\mathbf{x}_k} \leftarrow \boldsymbol{F}_k(\boldsymbol{\mu}_{\mathbf{x}_k}^{(l)}, \sigma_{\mathrm{x}}^{2(l)}\boldsymbol{I})$
  $\boldsymbol{\sigma}_{\mathbf{x}_k}^2 \leftarrow \boldsymbol{G}_k(\boldsymbol{\mu}_{\mathbf{x}_k}^{(l)}, \sigma_{\mathrm{x}}^{2(l)}\boldsymbol{I})$
  $v_a \leftarrow \mu_{\mathrm{z},a} \sum_i A_{a,i} \sum_{j\in\mathcal{K}} A_{a,j} \frac{\partial F_i(\boldsymbol{\mu}_{\mathbf{x}_k}^{(l)}, \sigma_{\mathrm{x}}^{2(l)}\boldsymbol{I})}{\partial \mu_{\mathrm{x}_j}^{(l)}}$
  $\boldsymbol{\mu}_{\mathbf{z}} \leftarrow \boldsymbol{y} - \boldsymbol{A}\boldsymbol{\mu}_{\mathbf{x}} + \boldsymbol{v}$
  $\sigma_{\mathrm{x}}^{2(l)} \leftarrow \sigma_{\mathrm{w}}^2 + \frac{1}{L}\sum_i \sigma_{\mathrm{x},i}^2$
until $t > t_{\max}$ **or** $t > 1$ **and** $\left\|\hat{\boldsymbol{\mu}}_{\mathbf{x}} - \hat{\boldsymbol{\mu}}_{\mathbf{x}}^{[t-1]}\right\|^2 < \varepsilon\|\boldsymbol{\mu}_{\mathbf{x}}\|^2$
$\hat{\boldsymbol{x}} = (\dots, \mu_{\mathbf{x}_k}^T, \dots)^T$

Algorithm 8 MD-BAMP with simplified Onsager term

$\boldsymbol{\mu}_{\mathbf{z}} \leftarrow \boldsymbol{y}$, $\sigma_{\mathrm{x}}^{2(l)} \leftarrow \frac{1}{L}\|\boldsymbol{y}\|^2$
Set constants σ_{w}^2, $t_{\max}$, ε.
All other variables are initialized to zero.
repeat
  $\boldsymbol{\mu}_{\mathbf{x}}^{(l)} \leftarrow \boldsymbol{A}^T\boldsymbol{\mu}_{\mathbf{z}} + \boldsymbol{\mu}_{\mathbf{x}}$
  $\boldsymbol{\mu}_{\mathbf{x}}^{[t-1]} \leftarrow (\boldsymbol{\mu}_{\mathbf{x}_0}^T, \boldsymbol{\mu}_{\mathbf{x}_1}^T, \dots, \boldsymbol{\mu}_{\mathbf{x}_K}^T)^T$
  $\boldsymbol{\mu}_{\mathbf{x}_k} \leftarrow \boldsymbol{F}_k(\boldsymbol{\mu}_{\mathbf{x}_k}^{(l)}, \sigma_{\mathrm{x}}^{2(l)}\boldsymbol{I})$
  $\boldsymbol{\sigma}_{\mathbf{x}_k}^2 \leftarrow \boldsymbol{G}_k(\boldsymbol{\mu}_{\mathbf{x}_k}^{(l)}, \sigma_{\mathrm{x}}^{2(l)}\boldsymbol{I})$
  $v_a \leftarrow \frac{\mu_{\mathrm{z},a}}{L} \sum_i \frac{\partial F_i(\boldsymbol{\mu}_{\mathbf{x}_k}^{(l)}, \sigma_{\mathrm{x}}^{2(l)}\boldsymbol{I})}{\partial \mu_{\mathrm{x}_j}^{(l)}}$
  $\boldsymbol{\mu}_{\mathbf{z}} \leftarrow \boldsymbol{y} - \boldsymbol{A}\boldsymbol{\mu}_{\mathbf{x}} + \boldsymbol{v}$
  $\sigma_{\mathrm{x}}^{2(l)} \leftarrow \sigma_{\mathrm{w}}^2 + \frac{1}{L}\sum_i \sigma_{\mathrm{x},i}^2$
until $t > t_{\max}$ **or** $t > 1$ **and** $\left\|\hat{\boldsymbol{\mu}}_{\mathbf{x}} - \hat{\boldsymbol{\mu}}_{\mathbf{x}}^{[t-1]}\right\|^2 < \varepsilon\|\boldsymbol{\mu}_{\mathbf{x}}\|^2$
$\hat{\boldsymbol{x}} = (\dots, \mu_{\mathbf{x}_k}^T, \dots)^T$

A.5 MD-BAMP for Variable Noise

This algorithm, dubbed BAMP-VN, extends MD-BAMP towards variable measurement noise. It is derived in Section 2.6.

Algorithm 9 BAMP-VN

$\boldsymbol{\mu}_{\mathsf{z}} \leftarrow \boldsymbol{y}$, $\sigma_{\mathsf{z}}^2 \leftarrow \|\boldsymbol{y}\|_2^2 L^{-1}$
Set constants $\sigma_{\mathsf{w}_a}^2$, $t_{\max}$, ε, $\boldsymbol{A}$.
All other variables are initialized to zero.
repeat
  $\sigma_{\mathsf{x}}^{2(l)} \leftarrow L \left(\sum_{\mathcal{W}_k \in \mathcal{V}} \frac{|\mathcal{W}_k|}{\sigma_{\mathsf{w},k}^2 + \sigma_{\mathsf{z}}^2} \right)^{-1}$
  $\mu'_{\mathsf{z},a} \leftarrow \frac{\sigma_{\mathsf{x}}^{2(l)}}{\sigma_{\mathsf{w},a}^2 + \sigma_{\mathsf{z}}^2} \mu_{\mathsf{z},a}$
  $\boldsymbol{\mu}_{\mathsf{x}}^{(l)} \leftarrow \boldsymbol{A}^T \boldsymbol{\mu}'_{\mathsf{z}} + \boldsymbol{\mu}_{\mathsf{x}}$
  $\boldsymbol{\mu}_{\mathsf{x}}^{[t-1]} \leftarrow \boldsymbol{\mu}_{\mathsf{x}}$
  $\boldsymbol{\mu}_{\mathsf{x}} \leftarrow \boldsymbol{F}(\boldsymbol{\mu}_{\mathsf{x}}^{(l)}, \sigma_{\mathsf{x}}^{2(l)})$
  $\boldsymbol{\sigma}_{\mathsf{x}}^2 \leftarrow \boldsymbol{G}(\boldsymbol{\mu}_{\mathsf{x}}^{(l)}, \sigma_{\mathsf{x}}^{2(l)})$
  $v_a \leftarrow \frac{\mu'_{\mathsf{z},a}}{L} \sum_i \frac{\partial F_i(\mu_{\mathsf{x},i}^{(l)}, \sigma_{\mathsf{x}}^{2(l)})}{\partial \mu_{\mathsf{x},i}^{(l)}}$
  $\boldsymbol{\mu}_{\mathsf{z}} \leftarrow \boldsymbol{y} - \boldsymbol{A}\boldsymbol{\mu}_{\mathsf{x}} + \boldsymbol{v}$
  $\sigma_z^2 \leftarrow \frac{N-1}{NL} \sum_i \sigma_{\mathsf{x},i}^2$
until $t > t_{\max}$ **or** $t > 1$ **and** $\left\| \boldsymbol{\mu}_{\mathsf{x}} - \boldsymbol{\mu}_{\mathsf{x}}^{[t-1]} \right\|_2^2 < \varepsilon \|\boldsymbol{\mu}_{\mathsf{x}}\|_2^2$
$\hat{\boldsymbol{x}} = \boldsymbol{\mu}_{\mathsf{x}}$

A.6 MD-BAMP for Nonzero-Mean Sensing Matrices

The MEAN-BAMP algorithm extends MD-BAMP towards compressed sensing problems with a nonzero-mean sensing matrix $\boldsymbol{B}$. Its derivation is presented in Section 2.7.

Algorithm 10 MEAN-BAMP

Set constants σ_{w}^2, $t_{\max}$, ε, $\boldsymbol{B}$.
$\boldsymbol{A} = \boldsymbol{B} - \mu_B \mathbf{1}\mathbf{1}^T$
$\boldsymbol{\mu_z} = \boldsymbol{v} - \frac{1}{L}\sum_a v_a$
$\sigma_{\mathsf{x}}^{2(l)} = \sigma_{\mathsf{w}}^2 + L^{-1}\left\|\boldsymbol{\mu_z}\right\|_2^2 + \frac{1}{L^2}\left(\sum_i \sigma_{\mathsf{x},i}^2 + L\sigma_{\mathsf{w}}^2\right)$
All other variables are initialized to zero.
repeat
  $\boldsymbol{\mu}_{\mathbf{x}}^{(l)} = \boldsymbol{A}^T\boldsymbol{\mu_z} + \boldsymbol{\mu_x}$
  $\boldsymbol{\mu_x} = \boldsymbol{F}(\boldsymbol{\mu}_{\mathbf{x}}^{(l)}, \boldsymbol{\sigma}_{\mathbf{x}}^{2(l)})$
  $\boldsymbol{\sigma}_{\mathbf{x}}^2 = \boldsymbol{G}(\boldsymbol{\mu}_{\mathbf{x}}^{(l)}, \boldsymbol{\sigma}_{\mathbf{x}}^{2(l)})$
  $\sigma_{vy\to\bar{\mathsf{x}}}^2 = \frac{1}{(N\mu_B)^2}\left(\frac{1}{L}\sum_i \sigma_{\mathsf{x},i}^2 + \sigma_{\mathsf{w}}^2\right)$
  $\sigma_{\bar{\mathsf{x}}\to vy}^2 = \left(\frac{N^2}{\sum_i \sigma_{\mathsf{x},i}^2} + \frac{L-1}{\sigma_{vy\to\bar{\mathsf{x}}}^2}\right)^{-1}$
  $\sigma_{vy\to y}^2 = \sigma_{\mathsf{w}}^2 + (N\mu_B)^2\sigma_{\bar{\mathsf{x}}\to vy}^2$
  $\sigma_{\mathsf{x}}^{2(l)} = \sigma_{vy\to y}^2 + \frac{1}{L}\sum_i \sigma_{\mathsf{x},i}^2$
  $\mu_{f,\bar{\mathsf{x}}\to\bar{\mathsf{x}}} = \frac{1}{N}\sum_i \mu_{\mathsf{x},i}$
  $\mu_{vy,a\to\bar{\mathsf{x}}} = \frac{1}{N\mu_B}\left(v_a - \sum_i A_{a,i}\mu_{\mathsf{x},i} + \frac{\mu_{\mathsf{z},a}}{L}\sum_i \frac{\partial F}{\partial \mu_{\mathsf{x},i}^{(l)}}\right)$
  $\mu_{\bar{\mathsf{x}}\to vy} = \sigma_{\bar{\mathsf{x}}\to vy}^2\left(\frac{\mu_{f,\bar{\mathsf{x}}\to\bar{\mathsf{x}}}}{\sigma_{f,\bar{\mathsf{x}}\to\bar{\mathsf{x}}}^2} + \frac{L-1}{L\sigma_{vy\to\bar{\mathsf{x}}}^2}\sum_b \mu_{vy,b\to\bar{\mathsf{x}}}\right)$
  $\boldsymbol{\mu}_{vy\to\boldsymbol{y}} = \boldsymbol{v} - N\mu_B\mu_{\bar{\mathsf{x}}\to vy}\mathbf{1}$
  $\boldsymbol{\mu_z} = \boldsymbol{\mu}_{vy\to\boldsymbol{y}} - \boldsymbol{A}\boldsymbol{\mu_x} + \boldsymbol{\mu_z}\frac{1}{L}\sum_i \frac{\partial F}{\partial \mu_{\mathsf{x},i}^{(l)}}$
until $t > t_{\max}$ **or** $t > 1$ **and** $\left\|\boldsymbol{\mu_x} - \boldsymbol{\mu}_{\mathbf{x}}^{[t-1]}\right\|_2^2 < \varepsilon\left\|\boldsymbol{\mu_x}\right\|_2^2$

Appendix B

Acronyms and Abbreviations

AMP Approximate Message Passing

AWGN Additive, White Gaussian Noise

BAMP Bayesian Approximate Message Passing

BAMP-VN Bayesian Approximate Message Passing for Variable Noise variance

CS Compressed Sensing

DLP Digital Light Processing

EUSIPCO European Signal Processing Conference

FHT Fast Hadamard Transform

GAMP Generalized Approximate Message Passing

GMP Gaussian Message Passing

ICASSP International Conference on Acoustics, Speech and Signal Processing

IHT Iterative Hard Thresholding

IST Iterative Soft Thresholding

KLT Karhunen-Loève Transformation

LASSO Leas Absolute Shrinkage and Selection Operator

LDPC Low Density Parity Check

MEAN-BAMP Bayesian Approximate Message Passing for nonzero-MEAN sensing matrices

MD-BAMP Bayesian Approximate Message Passing for Multi-Dimensional priors

MD-GMP Gaussian Message Passing for Multi-Dimensional priors

MIMO Multiple-Input and Multiple-Output

MP Matching Pursuit

MRI Magnetic Resonance Imaging

MSE Mean Squared Error

NMSE Normalized Mean Squared Error

OFDM Orthogonal Frequency Division Multiplexing

OMP Orthogonal Matching Pursuit

pdf Probability Density Function

SDR Signal-to-Distortion Ratio

SNR Signal-to-Noise Ratio

SPA Sum-Product Algorithm

SPARC Sparse Regression Code

SPCC Single-Parity Check Code

VAMP Vector Approximate Message Passing

i.i.d. independent and identically distributed

w.r.t. with respect to

s.t. subject to

Bibliography

[1] IEEE standard for binary floating-point arithmetic. *ANSI/IEEE Std 754-1985*, pages 1–20, October 1985.

[2] George E. Andrews, Richard Askey, and Ranjan Roy. *Special Functions*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1999.

[3] M. Bayati and Andrea Montanari. The dynamics of message passing on dense graphs, with applications to compressed sensing. *IEEE Transactions on Information Theory*, 57(2):764–785, February 2011.

[4] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[5] Raphael Berthier, Andrea Montanari, and Phan-Minh Nguyen. State evolution for approximate message passing with non-separable functions. *arXiv preprint arXiv:1708.03950*, 2017.

[6] Stefan C. Birgmeier. Utilization of correlation between signal components for compressed-sensing recovery. Master's thesis, TU Wien, 2015.

[7] Stefan C. Birgmeier and Norbert Goertz. Optimizing approximate message passing for variable measurement noise. *2018 Proceedings of the 26th European Signal Processing Conference (EUSIPCO)*, pages 489–493, September 2018.

[8] Stefan C. Birgmeier and Norbert Goertz. Approximate message passing for joint activity detection and decoding in non-orthogonal CDMA. In *23rd International ITG Workshop on Smart Antennas (WSA 2019)*, pages 1–5, April 2019.

[9] Stefan C. Birgmeier and Norbert Goertz. Exploiting general multi-dimensional priors in compressed-sensing reconstruction. In *12th International ITG Conference on Systems, Communications and Coding (SCC 2019)*, pages 113–118. VDE Verlag, February 2019.

[10] Stefan C. Birgmeier and Norbert Goertz. Robust approximate message passing for nonzero-mean sensing matrices. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4898–4902, May 2019.

[11] Stefan C. Birgmeier and Norbert Goertz. Variable-rate channel-coding using compressive sensing. 19th Joint Workshop on Communications and Coding (JWCC), March 2019.

[12] Thomas Blumensath and Mike E. Davies. Iterative thresholding for sparse approximations. *Journal of Fourier analysis and Applications*, 14(5-6):629–654, 2008.

[13] Thomas Blumensath and Mike E. Davies. Iterative hard thresholding for compressed sensing. *Applied and computational harmonic analysis*, 27(3):265–274, 2009.

[14] Erwin Bolthausen. An iterative construction of solutions of the TAP equations for the Sherrington–Kirkpatrick model. *Communications in Mathematical Physics*, 325(1):333–366, 2014.

[15] P. T. Boufounos and R. G. Baraniuk. 1-bit compressive sensing. In *2008 42nd Annual Conference on Information Sciences and Systems*, pages 16–21, March 2008.

[16] F. Caltagirone, L. Zdeborová, and F. Krzakala. On convergence of approximate message passing. In *2014 IEEE International Symposium on Information Theory*, pages 1812–1816, June 2014.

[17] Emmanuel J. Candes, Justin Romberg, and Terence Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, February 2006.

[18] Wai Lam Chan, Matthew L. Moravec, Richard G. Baraniuk, and Daniel M. Mittleman. Terahertz imaging with compressed sensing and phase retrieval. *Optics Letters*, 33(9):974–976, May 2008.

[19] Scott Shaobing Chen, David L. Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[20] John P. Cunningham, Philipp Hennig, and Simon Lacoste-Julien. Gaussian probabilities and expectation propagation. *arXiv preprint arXiv:1111.6832*, 2011.

[21] 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*, 57(11):1413–1457, November 2004.

[22] David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, April 2006.

[23] David L. Donoho, Arian Maleki, and Andrea Montanari. Message passing algorithms for compressed sensing: I. motivation and construction. In *2010 IEEE Information Theory Workshop on Information Theory (ITW 2010, Cairo)*, pages 1–5, January 2010.

[24] David L. Donoho, Arian Maleki, and Andrea Montanari. Message passing algorithms for compressed sensing: II. analysis and validation. In *Proceedings IEEE Information Theory Workshop (ITW)*, pages 1–5, 2010.

[25] David L. Donoho, Arian Maleki, and Andrea Montanari. The noise-sensitivity phase transition in compressed sensing. *IEEE Transactions on Information Theory*, 57(10):6920–6941, October 2011.

[26] David L. Donoho and Jared Tanner. Counting the faces of randomly-projected hypercubes and orthants, with applications. *Discrete & computational geometry*, 43(3):522–541, 2010.

[27] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 25(2):83–91, March 2008.

[28] Digital video broadcasting (DVB); second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications; part 2: DVB-S2 extensions (DVB-S2X). Standard, ETSI, 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex - FRANCE, February 2015.

[29] 5G NR; multiplexing and channel coding (3GPP TS 38.212 version 15.3.0 release 15). Standard, ETSI, 650 Route des Lucioles F-06921 Sophia Antipolis Cedex - FRANCE, October 2018.

[30] R. F. H. Fischer and F. Waeckerle. Peak-to-average power ratio reduction in OFDM via sparse signals: Transmitter-side tone reservation vs. receiver-side compressed sensing. In *OFDM 2012; 17th International OFDM Workshop 2012 (InOWo'12)*, pages 1–8, August 2012.

[31] Robert G. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, January 1962.

[32] Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. Standard, IEEE, 3 Park Avenue, New York, NY 10016-5997, USA, December 2016.

[33] IEEE standard for Ethernet. Standard, IEEE, 3 Park Avenue, New York, NY 10016-5997, USA, August 2018.

[34] L. Isserlis. On A Formula For The Product-Moment Coefficient Of Any Order Of A Normal Frequency Distribution In Any Number Of Variables. *Biometrika*, 12(1-2):134–139, 11 1918.

[35] G. J. Hurford, E. Schmahl, R.A. Schwartz, A.J. Conway, Markus Aschwanden, André Csillaghy, Brian Dennis, Christopher Johns-Krull, S. Krucker, R. P. Lin, et al. The RHESSI imaging concept. In *The Reuven Ramaty High-Energy Solar Spectroscopic Imager (RHESSI)*, pages 61–86. Springer, 2003.

[36] C. Jeon, R. Ghods, Arian Maleki, and C. Studer. Optimality of large MIMO detection via approximate message passing. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 1227–1231, June 2015.

[37] Yalei Ji, Carsten Bockelmann, and Armin Dekorsy. Numerical analysis for joint PHY and MAC perspective of compressive sensing multi-user detection with coded random access. In *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1018–1023. IEEE, 2017.

[38] Yu Kou, Shu Lin, and Marc PC Fossorier. Low-density parity-check codes based on finite geometries: a rediscovery and new results. *IEEE Transactions on Information Theory*, 47(7):2711–2736, November 2001.

[39] F. Krzakala, M. Mézard, F. Sausset, Y. F. Sun, and L. Zdeborová. Statistical-physics-based reconstruction in compressed sensing. *Phys. Rev. X*, 2:021005, May 2012.

[40] F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, February 2001.

[41] David J. C. MacKay. Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 45(2):399–431, March 1999.

[42] David J. C. MacKay and Radford M. Neal. Near Shannon limit performance of low density parity check codes. *Electronics Letters*, 33(6):457–458, March 1997.

[43] Arian Maleki. *Approximate Message Passing Algorithms for Compressed Sensing*. PhD thesis, Stanford University, September 2011.

[44] Stéphane G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993.

[45] X. Meng, S. Wu, L. Kuang, and J. Lu. Concise Derivation of Complex Bayesian Approximate Message Passing via Expectation Propagation. *ArXiv e-prints*, September 2015.

[46] Christopher A. Metzler, Arian Maleki, and Richard G. Baraniuk. From denoising to compressed sensing. *IEEE Transactions on Information Theory*, 62(9):5117–5144, September 2016.

[47] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, pages 40–44 vol.1, November 1993.

[48] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. *The matrix cookbook*. Technical University of Denmark, November 2008.

[49] Sundeep Rangan. Generalized approximate message passing for estimation with random linear mixing. In *2011 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 2168–2172. IEEE, July 2011.

[50] Sundeep Rangan, Philip Schniter, and Alyson K. Fletcher. Vector approximate message passing. *CoRR*, abs/1610.03082, 2016.

[51] Alfréd Rényi. On the dimension and entropy of probability distributions. *Acta Mathematica Hungarica*, 10(1-2):193–215, 1959.

[52] C. Rush, A. Greig, and R. Venkataramanan. Capacity-achieving sparse regression codes via approximate message passing decoding. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 2016–2020, June 2015.

[53] S. Semper, J. Kirchhof, C. Wagner, F. Krieg, F. Römer, A. Osman, and G. Del Galdo. Defect detection from 3D ultrasonic measurements using matrix-free sparse recovery algorithms. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 1700–1704, September 2018.

[54] M. Sipser and D. A. Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42(6):1710–1722, November 1996.

[55] S. Som and P. Schniter. Compressive imaging using approximate message passing and a Markov-tree prior. *IEEE Transactions on Signal Processing*, 60(7):3439–3448, July 2012.

[56] S. ten Brink. Convergence of iterative decoding. *Electronics Letters*, 35(10):806–808, May 1999.

[57] S. ten Brink. Convergence behavior of iteratively decoded parallel concatenated codes. *IEEE Transactions on Communications*, 49(10):1727–1737, October 2001.

[58] Robert Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[59] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge University Press, 2018.

[60] J. Vila, P. Schniter, S. Rangan, F. Krzakala, and L. Zdeborová. Adaptive damping and mean removal for the generalized approximate message passing algorithm. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2021–2025, April 2015.

[61] J. P. Vila and P. Schniter. Expectation-maximization Gaussian-mixture approximate message passing. *IEEE Transactions on Signal Processing*, 61(19):4658–4672, October 2013.

[62] Christoph Wagner and Sebastian Semper. Fast linear transformations in python. *CoRR*, abs/1710.09578, 2017.

[63] S. Wu, L. Kuang, Z. Ni, J. Lu, D. Huang, and Q. Guo. Low-complexity iterative detection for large-scale multiuser MIMO-OFDM systems using approximate message passing. *IEEE Journal of Selected Topics in Signal Processing*, 8(5):902–915, October 2014.

[64] Y. Wu and S. Verdú. Rényi information dimension: Fundamental limits of almost lossless analog compression. *IEEE Transactions on Information Theory*, 56(8):3721–3748, August 2010.

[65] Y. Wu and S. Verdú. Optimal phase transitions in compressed sensing. *IEEE Transactions on Information Theory*, 58(10):6241–6263, October 2012.

[66] Y. Wu and S. Verdú. Optimal phase transitions in compressed sensing with noisy measurements. In *2012 IEEE International Symposium on Information Theory Proceedings*, pages 1638–1642, July 2012.

[67] Yihong Wu. *Shannon Theory for Compressed Sensing*. PhD thesis, Princeton University, 2011.

[68] J. Zhang, H. He, C. Wen, S. Jin, and G. Y. Li. Deep learning based on orthogonal approximate message passing for CP-free OFDM. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8414–8418, May 2019.

[69] Hao Zhong, Yan LI, Radoslav Danilak, and Earl T. Cohen. LDPC erasure decoding for flash memories, June 2017.

[70] A. Zymnis, S. Boyd, and E. Candes. Compressed sensing with quantized measurements. *IEEE Signal Processing Letters*, 17(2):149–152, February 2010.