# Belief Propagation with Deep Unfolding for High-dimensional Inference in Communication Systems

by

Mengke Lian

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

_____
Henry Pfister, Advisor

_____
Arthur Robert Calderbank

_____
Galen Reeves

_____
Alexander Volfovsky

_____
Scott Schmidler

Dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the Graduate School of
Duke University

2019

# ABSTRACT

## Belief Propagation with Deep Unfolding for High-dimensional Inference in Communication Systems

by

Mengke Lian

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

_____
Henry Pfister, Advisor

_____
Arthur Robert Calderbank

_____
Galen Reeves

_____
Alexander Volfovsky

_____
Scott Schmidler

An abstract of a dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the Graduate School of
Duke University

2019

# Abstract

High-dimensional probability distributions are important objects in a wide variety of applications for example, most prediction and inference applications focus on computing the posterior marginal of a subset of variables conditioned on observations of another subset of variables. In practice, this is untractable due to the curse of dimensionality. In some problems, high-dimensional joint probability distributions can be represented by factor graphs. For such problems, belief propagation (BP) is a polynomial-time algorithm that provides an efficient approximation of the posterior marginals, and it is exact if the factor graph does not contain cycles. With rapid improvements in machine learning over the past decade, using machine learning techniques to optimize system parameters is an emerging field in communication research.

This thesis considers applying BP for communication systems, and focuses on incorporating domain knowledge into machine learning models. For compressive sensing, two variants of relaxed belief propagation (RBP) algorithm are proposed. One improves the stability over a larger class of measurement matrices and the other reduces the computational complexity when measurement matrix is in the product of several sparse matrices. For optical communication, the non-linear Schrödinger equation is solved by modeling the signal in each step of split-step Fourier method as a multivariate complex Gaussian distribution. Then, the parameters of the Gaussian are tracked through in digital back-propagation. For recursive decoding for Reed–Muller codes, the algebraic structure of the code is utilized and a recursive BP approach for redundant factor graphs is developed for near-optimal decoding. Finally, we use deep unfolding to unroll BP decoding as a recursive neural network and introduce the idea of a the parameter adaptive network to learn the relation between channel SNR and optimal BP weight factors.

# Acknowledgements

I felt myself being so blessed for the past five years. The completion of this dissertation gives the opportunity to present my gratitude for many individuals who influenced me in numerous aspects during my PhD study.

First of all, I would like to express my deepest gratitude to my advisor, Professor Henry Pfister, for his invaluable mentoring and support. It's hard to evaluate how much his support, guidance and encouragement means to me throughout my PhD study. His way of supervision style and friendly communication shaped our research lab to a relatively relaxed atmosphere, allowing a large degree of freedom for my research direction, collaboration, and internship. Without his encourage and support I cannot build such a broad interdisciplinary background. These are beneficial not only for my study and career development, but also for my whole life.

I would like to show my appreciation to Galen Reeves, Robert Calderbank, Alexander Volfovsky, Scott Schmidler, for their time and effort to serve on my committee at various stages of my PhD study. I would like also to thank Patrick Charbonneau, Florent Krzakala and Lenka Zdeborova for their mentor and inspiration to guide me to the statistical physics world.

Fortunately, I also met a lot of incredible people, making my Ph.D. life an invaluable memory. I want to thank Christian Häger, who was a fantastic friend and a patient mentor for me. His collaboration is crucial to my research. I also want to thank Narayanan Rengawamy for his helpful discussions on quantum topics, we had a lot interesting brainstormings. I would like to thank Kai Fan, Yizhe Zhang and Chunyuan Li for the insightful discussions on deep learning topics. I want to thank Fabrizio Carpi, Sarah Brandsen, Elia Santi and Vaishaki Mayya for all the patient cooperation and valuable discussions. I also want to thank the many other

people I've interacted with at Duke, both at iiD and in the Electrical and Computer Engineering Department.

I would also like to thank my mentors/collaborators in the two internships of my PhD, including Wenjie Zhao at AT&T Lab, as well as Fan Zhang at Algorithm group of SK Hynix Memory Solutions, America. They directed my research from a more applicable way, broadening my industrial horizon.

Finally, I want to express my deepest thankfulness to my parents and family. I thank them for their support, and understanding. Everything they have done for me and fills my life with happiness.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

High-dimensional probability distributions are important objects in a wide variety of applications for example, most prediction and inference applications focus on computing the posterior marginal of a subset of variables conditioned on observations of another subset of variables. In practice, this is untractable due to the curse of dimensionality. In some problems, high-dimensional joint probability distributions can be represented by factor graphs. The complexity can be reduced dramatically when the underlying factor graph has some special structure. On trees, with bounded degree marginals can be computed in a number of operations that grows linearly with the graph size. This can be done through a 'dynamic programming' procedure that recursively sums over all variables, starting from the leaves and progressing towards the 'center' of the tree.

Belief propagation (BP) [MM09] is a polynomial-time algorithm for factor graphs that provides an efficient approximation of the posterior marginals, and it is exact if the factor graph does not contain cycles. It has a wide range of applications to various problems in statistical physics, signal processing, high-dimensional statistical inference and communication systems. However, for loopy factor graphs, the performance of BP is a complicated function of the structure of the factor graph, the message passing scheduling, and the precise update equation. In general, BP is a suboptimal algorithm with low complexity. But, there has been significant research on how to reformulate problems such that BP can approach optimal performance.

However, rapid improvements in machine learning over the past decade are beginning to have far-reaching effects. Prior to the current revolution in machine learning, the majority of communication engineers were quite aware that system parameters (such as filter coefficients) could be learned using stochastic gradient descent. It was not at all clear, however,

that more complicated parts of the system architecture could be learned as well. This leads to interesting problems such as how should one apply machine learning to communication systems, and how can one incorporate domain knowledge in these machine learning models [LHP19].

This thesis makes several contributions to the above topics. Chapter 2 focuses on applying BP on compressed sensing problems. Starting from the relaxed belief propagation (RBP) [GW06], two variants are proposed to improve stability and computational complexity. The RBP-SVD algorithm is inspired by the vector AMP [RSF16], good stability is achieved by reformulating the problem to avoid ill-conditioned measurement matrices. The RBP-SEQ algorithm borrows idea of FFT, and drastically reduces the computation complexity when the measurement matrix has a special form. Chapter 3 extends digital back-propagation in optical communication systems by modeling the posterior distribution as a multivariate complex Gaussian distribution. An explicit evolution equation for the covariance matrix is obtained by solving the nonlinear Schrödinger equation using the split-step Fourier method. For symbol detection tasks, the comprehensive information in this covariance matrix enables improved performance. In Chapter 4, we investigate the rich algebraic structure of Reed–Muller codes, and find relationships between BP and a new decoding algorithm [YA19]. Based on this, a novel decoding algorithm for high-order Reed–Muller codes is proposed. For $\mathrm{RM}(m-3, m)$, it achieves near-optimal performance for BP on a redundant factor graph. Finally, Chapter 5 builds up the connections between deep learning and channel coding by using deep unfolding to unroll iterations of BP into a multi-layered computation network. Then BP is extended to weighted BP and the optimal parameters weights are are trained with modern deep learning frameworks [AAB+15], [PGC+17]. We also propose parameter adaptive networks (PANs) that learn how the values of the optimal parameters as a function of signal-to-noise ratio. Simulation results show that the learned WBP decoder achieves optimal performance over a wide range of channel conditions.

# Chapter 2

# Relaxed Belief-Propagation for Compressed Sensing

## 2.1 Background

Compressed sensing (CS) is an area of signal processing and statistics that first emerged in the 1990's and then became very active in the mid-2000's [CDS98, BGV99, GKMS01, VMB02, CM05, Don06, CRT06]. The main idea is that many signals can be reconstructed and/or processed using many fewer measurements than predicted by a naïve application of Nyquist's theorem.

One widely used method for signal reconstruction in CS is based on the minimization of the $\ell_1$-norm of the signal under a linear equality constraint [Don06]. This approach can also be extended to probabilistic and Bayesian settings. In [BSB10], a straightforward application of belief propagation, called CSBP, is introduced for sparse measurement matrices. By using Gaussian approximation of messages to avoid tracking them exactly, the relaxed belief propagation (RBP) algorithm [GW06] can greatly reduce the computation complexity of CSBP. In [DMM09], the approximate message passing (AMP) algorithm is developed for the case where the measurement factors depend weakly on a large number of random variables. A rigorous analysis for AMP called state evolution is also derived to evaluate the average performance of the algorithm in the high-dimensional limit [BM11]. The drawback of AMP is that the algorithm is fragile when the measurement matrix is not i.i.d. sub-Gaussian. Recently, vector AMP (VAMP) [RSF16] was proposed and it also has a rigorous state evolution which holds for right-rotationally invariant matrices. This is a much broader class of random matrices.

In general, the compressive sensing problem is an linear inverse problem under sparsity constraints. Solving linear inverse problems is common in many fields, such as the decoding of linear codes in channel coding, variable selection problem in statistics, and recovering missing pixels from a linear model in machine learning. Other applications include compression, denoising, and super-resolution estimation.

In this chapter, several iterative reconstruction algorithms for compressive sensing are compared. Our main contribution is the introduction of two variants of the relaxed belief-propagation (RBP) algorithm. We focus on improving the convergence behavior of the RBP algorithm, so that its recursion is stable for a larger class of measurement matrices. Also, under specific measurement matrix structure (product of sparse matrices), a low-complexity variant of RBP algorithm for structured matrices is also derived based on the idea of the fast Fourier transform (FFT) algorithm.

## 2.2 Preliminaries

### 2.2.1 Compressed Sensing

The definition of the compressed sensing problem studied in this report is as follows:

$$\mathbf{y} = \mathbf{\Phi}\,\mathbf{x} + \mathbf{w},$$

where

- the $m$-by-$n$ measurement matrix $\mathbf{\Phi}$ is known,
- the observation is $y_a = \sum_{i=1}^{n} \Phi_{ai} x_i + w_a$ for $a = 1, \ldots, m$,
- the signal entries $\{x_i\}$ is mutually independent with $x_i \sim p_{X_i}$ for $i = 1, \ldots, n$, and
- the additive noise $\{w_a\}$ is mutually independent with $w_a \sim p_{W_a} = \mathcal{N}(0, \sigma_{W,a}^2)$.

Therefore, the joint probability of signal $\mathbf{x}$ and measurement $\mathbf{y}$ can be written as

$$p(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{n} p_{X_i}(x_i) \cdot \prod_{a=1}^{m} \frac{1}{\sqrt{2\pi\sigma_{W,a}^2}} \exp\left( -\frac{\left(y_a - \sum_{i=1}^{m} \Phi_{ai} x_i\right)^2}{2\sigma_{W,a}^2} \right).$$

A reasonable choice is to use the most probable signal $\hat{\mathbf{x}}$ given an measurement $\mathbf{y}$ as the estimate of the true signal $\mathbf{x}$. In that case, the solution is given by maximum a posteriori (MAP) estimation

$$\begin{aligned}\hat{\mathbf{x}} &= \arg\max_{\mathbf{x}\in\mathbb{R}^n} p\left(\mathbf{x}\,|\,\mathbf{y}\right) = \arg\max_{\mathbf{x}\in\mathbb{R}^n} p(\mathbf{x},\mathbf{y}) \\ &= \arg\max_{\mathbf{x}\in\mathbb{R}^n} \left\{\prod_{i=1}^{n} p_{X_i}(x_i) \prod_{a=1}^{m} \exp\left(-\frac{\left(y_a - \sum_{i=1}^{m}\Phi_{ai}x_i\right)^2}{2\sigma_{W,a}^2}\right)\right\} \\ &= \arg\min_{\mathbf{x}\in\mathbb{R}^n} \left\{\frac{1}{2}\left\|\boldsymbol{\Sigma}_W^{-1/2}\left(\mathbf{y}-\boldsymbol{\Phi}\mathbf{x}\right)\right\|_2^2 + f(\mathbf{x})\right\}, \end{aligned} \tag{2.1}$$

$$\text{where} \quad \boldsymbol{\Sigma}_W = \begin{bmatrix} \sigma_{W,1}^2 & & \\ & \ddots & \\ & & \sigma_{W,m}^2 \end{bmatrix} \quad \text{and} \quad f(\mathbf{x}) = -\sum_{i=1}^{n} \log\left(p_{X_i}(x_i)\right).$$

### 2.2.2 Factor Graphs

A factor graph is a bipartite graph that defines the conditional independence structure of a set of random variables. Such graphs are widely used to represent the setup of statistical inference problems. In practice, they are popular because the BP algorithm can be used to efficiently solve (or approximate) marginal inference problems associated with factor graphs [KFL01b, MM09].

A factor graph $\mathcal{G} = (\mathcal{V}, \mathcal{F}, \mathcal{E})$ is defined by a set of variable nodes $\mathcal{V}$, a set of factor nodes $\mathcal{F}$, and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{F}$ that connect variables and factors. For a variable node $i \in \mathcal{V}$ or a factor node $a \in \mathcal{F}$, let $\partial$ denote the neighborhood operator defined by

$$\partial i \triangleq \left\{b \in \mathcal{F}\,|\,(i,b) \in \mathcal{E}\right\}, \qquad \partial a \triangleq \left\{j \in \mathcal{V}\,|\,(j,a) \in \mathcal{E}\right\}.$$

To construct the factor graph for a specific compressed sensing problem, one can simply set $\mathcal{V} = \{1,\ldots,n\}$, $\mathcal{F} = \{1,\ldots,m\}$ and $\mathcal{E} = \{(i,a) \in \mathcal{V}\times\mathcal{F}\,|\,\Phi_{ai} \neq 0\}$. Furthermore, the compatibility functions for nodes are defined by

$$\begin{aligned}\psi_a(\mathbf{x}_{\partial a}) &= \frac{1}{\sqrt{2\pi\sigma_{W,a}^2}} \exp\left(-\frac{\left(y_a - \sum_{i\in\partial a}\Phi_{ai}x_i\right)^2}{2\sigma_{W,a}^2}\right) \\ &= \frac{1}{\sqrt{2\pi\sigma_{W,a}^2}} \exp\left(-\frac{\left(y_a - \sum_{i=1}^{m}\Phi_{ai}x_i\right)^2}{2\sigma_{W,a}^2}\right).\end{aligned}$$

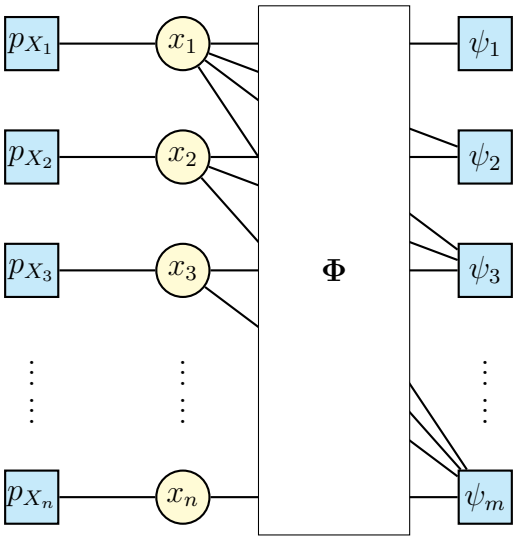



**Figure 2.1**: Factor graph representation of the compressed sensing problem.

## 2.3 Previous Works

In this section, we discuss various existing algorithms for solving compressed sensing problem. These algorithms are grouped by their space complexity (or number of messages) into $O(|\mathcal{E}|)$ edges-based algorithms and $O(|\mathcal{V}|)$ node-based algorithms.

Thus, when the factor graph is dense, node-based algorithms use fewer messages and are more time and space efficient. However, one has to pay the price of additional assumptions on measurement matrix for node-based algorithms to perform well.

### 2.3.1 Edge-based Algorithms

#### Belief Propagation (BP)

Assume that all the variables share same alphabet $\mathcal{X}$. BP operates by passing messages, $\mu_{i\to a}(x)$ and $\hat{\mu}_{a\to i}(x)$, between variable and factor nodes for every edge $(i,a) \in \mathcal{E}$. These messages are probability distributions over $\mathcal{X}$. Each iteration contains two phases. First

the variables nodes pass messages to all adjacent factor nodes, then the factor nodes pass messages to all adjacent variable nodes.

If the factor graph is a tree, then the messages will become constant after sufficiently many iterations and the algorithm computes the true marginals. If the factor graph has cycles, then the sequence of messages may or may not converge. Even if they converge, the final results will only give an approximation of the true marginals. In practice, a finite number of iterations are performed and damping is used to avoid oscillation and improve convergence. In the final step, the marginal of each variable is computed by combining all messages that enter the variable node.

The BP algorithm is described in pseudo code below. The functions $\mu_{i\to a}(x)$ and $\hat{\mu}_{a\to i}(x)$ represent messages from variable-node $i$ to factor-node $a$ and from factor-node $a$ to variable-node $i$, respectively. To simplify the notation, sets of variable-to-factor and factor-to-variable messages are denoted by $\boldsymbol{\mu}_{I\to a} = \{\mu_{i\to a} \,|\, i \in I\}$ for $I \subseteq \mathcal{V}$ and $\hat{\boldsymbol{\mu}}_{A\to i} = \{\hat{\mu}_{a\to i} \,|\, a \in A\}$ for $A \subseteq \mathcal{F}$.

**Algorithm 1** BP Algorithm for Compressed Sensing (CSBP)

Initialize: $\mu_{i\to a}^{(0)} = p_{X_i}, \quad (i,a) \in \mathcal{E}$
**for** $t = 1$ **to** $T$ **do**
  **for** $a = 1$ **to** $m$ **do**
    `/* Convolution step at factor nodes */`
    $\hat{\mu}_{a\to i}^{(t)} \leftarrow \mathsf{FNOP}(y_a, \boldsymbol{\mu}_{\partial a\setminus i\to a}^{(t-1)}, p_{W_a}), \quad \forall i \in \partial a$
  **end for**
  **for** $i = 1$ **to** $n$ **do**
    `/* Multiplication step at variable nodes */`
    $\mu_{i\to a}^{(t)} \leftarrow \mathsf{VNOP}(\hat{\boldsymbol{\mu}}_{\partial i\setminus a\to i}^{(t)}, p_{X_i}), \quad \forall a \in \partial i$
  **end for**
**end for**
**for** $i = 1$ **to** $n$ **do**
  $\mu_i \leftarrow \mathsf{VNOP}(\hat{\boldsymbol{\mu}}_{\partial i\to i}^{(T)}, p_{X_i}), \quad \hat{x}_i = \mathsf{EST}(\mu_i)$
**end for**

In the above pseudocode, FNOP represents the factor-node update operation and VNOP represents the variable-node update operation [BSB10]. Likewise, EST represents the operation that computes the signal estimate $\hat{x}_i$ from the approximate marginal $\mu_i$ respectively.

These operations are defined as follows:

- $\mathsf{FNOP}(\cdot,\cdot)$ returns a distribution that satisfies

$$\hat{\mu}_{a\to i}(x_i) \propto \int_{\mathbf{x}_{\partial a\setminus i}} p_{W_a}\left(y_a - \sum_{j\in\partial a} \Phi_{aj} x_j\right) \prod_{j\in\partial a\setminus i} \mu_{j\to a}(x_j)\,\mathrm{d}\mathbf{x}_{\partial a\setminus i}.$$

- $\mathsf{VNOP}(\cdot,\cdot)$ returns a distribution that satisfies

$$\mu_{i\to a}(x_i) \propto p_{X_i}(x_i) \times \prod_{b\in\partial i\setminus a} \hat{\mu}_{b\to i}(x_i), \qquad \mu_i(x_i) \propto p_{X_i}(x_i) \times \prod_{b\in\partial i} \hat{\mu}_{b\to i}(x_i).$$

- For signal estimate from a given marginal distribution, $\mathsf{EST}(\mu)$ is typically chosen to be the mean or the mode of distribution $\mu(x)$.

### Relaxed Belief Propagation (RBP)

The main drawback of BP algorithm is the expensive computation induced by keeping track of the exact message probability distributions. The relaxed belief-propagation algorithm mitigates this issue by approximating each probability distribution by a Gaussian distribution, which is parameterized by its mean and variance.

For the factor-node convolution step, a factor node $a$ has measurement $y_a$ and noise $w_a \sim \mathcal{N}(0, \sigma^2_{W,a})$. Consider the message passed along edge $(i,a)$ to variable node $i$. RBP approximates the $|\partial a|-1$ incoming messages as PDFs of Gaussian random variables $\mu_{j\to a} = \mathcal{N}(m_{j\to a}, v_{j\to a})$ for $j \in \partial a\setminus i$. Since in BP all incoming messages are assumed to be independent, therefore the outgoing message is also the PDF of a Gaussian random variable $\hat{\mu}_{a\to i} = \mathcal{N}(\hat{m}_{a\to i}, \hat{v}_{a\to i})$ whose distribution is the same as $\Phi^{-1}_{ai}\left(y_a - \sum_{j\in\partial a\setminus i} \Phi_{aj} x_{j\to a} - w_a\right)$. Using well-known linearity properties of Gaussian random variables, one can compute the update equation for the parameters at factor nodes:

$$\hat{v}_{a\to i} = \Phi^{-2}_{ai}(\sigma^2_{W,a} + \sum_{j\in\partial a\setminus i} \Phi^2_{aj} v_{j\to a}) \tag{2.2}$$

$$\hat{m}_{a\to i} = \Phi^{-1}_{ai}(y_a - \sum_{j\in\partial a\setminus i} \Phi_{aj} m_{j\to a}). \tag{2.3}$$

For the output message passed along edge $(i, a)$ to factor node $a$, there are $|\partial i| - 1$ incoming messages whose PDFs are associated with Gaussian random variables $\hat{\mu}_{b\to i} = \mathcal{N}(\hat{m}_{b\to i}, \hat{v}_{b\to i})$. During the variable-node update, the distributions of the signal prior and all incoming messages are multiplied pointwise and then renormalized. Recall that the distribution obtained by the normalized pointwise multiplication of Gaussian functions is also a Gaussian function. Thus, the normalized pointwise multiplication results in the Gaussian function denoted by $\tilde{\mu}_{i\to a} = \mathcal{N}(\tilde{m}_{i\to a}, \tilde{v}_{i\to a})$. Computing the mean and variance of this distribution leads to the update equations

$$\tilde{v}_{i\to a} = \frac{1}{\sum_{b\in\partial i\setminus a} 1/\hat{v}_{b\to i}} \tag{2.4}$$

$$\tilde{m}_{i\to a} = \tilde{v}_{i\to a} \sum_{b\in\partial i\setminus a} \frac{\hat{m}_{b\to i}}{\hat{v}_{b\to i}}. \tag{2.5}$$

Next, the $\tilde{\mu}_{i\to a}$ message distribution is multiplied pointwise by the signal prior $p_{X_i}$ and renormalized. If the signal prior is not Gaussian, then the resulting function is not Gaussian as well. Regardless, one computes the mean and variance of the resulting distribution using

$$\eta_i(m, v) = \frac{\int x\, p_{X_i}(x) \exp\left(-\frac{(x-m)^2}{2v}\right) \mathrm{d}x}{\int p_{X_i}(x) \exp\left(-\frac{(x-m)^2}{2v}\right) \mathrm{d}x} \tag{2.6}$$

$$\theta_i(m, v) = \frac{\int x^2 p_{X_i}(x) \exp\left(-\frac{(x-m)^2}{2v}\right) \mathrm{d}x}{\int p_{X_i}(x) \exp\left(-\frac{(x-m)^2}{2v}\right) \mathrm{d}x} - \left[\eta_i(m, v)\right]^2. \tag{2.7}$$

Using this, the message passed along edge $(i, a)$ to factor node $a$ is associated with the PDF $\mathcal{N}(m_{i\to a}, v_{i\to a})$, where

$$m_{i\to a} = \eta_i(\tilde{m}_{i\to a}, \tilde{v}_{i\to a}), \quad v_{i\to a} = \theta_i(\tilde{m}_{i\to a}, \tilde{v}_{i\to a}) \tag{2.8}$$

**Algorithm 2** Relaxed Belief Propagation (RBP)

Initialize: $m_{i\to a} = \eta_i(0,\infty), v_{i\to a} = \theta_i(0,\infty), \quad \forall\,(i,a) \in \mathcal{E}$
**for** $t = 1$ **to** $T$ **do**
  `/* Convolution step at factor nodes */`
  **for** $a = 1$ **to** $m$ **do**
    Update $(\hat{m}_{a\to i}, \hat{v}_{a\to i})$ using messages $\{(m_{j\to a}, v_{j\to a}) \,|\, j \in \partial a \setminus i\}$ by eqn (2.2), (2.3)
  **end for**
  `/* Multiplication step at variable nodes */`
  **for** $i = 1$ **to** $n$ **do**
    Update $(\tilde{m}_{i\to a}, \tilde{v}_{i\to a})$ using messages $\{(\hat{m}_{b\to i}, \hat{v}_{b\to i}) \,|\, b \in \partial i \setminus a\}$ by eqn (2.4), (2.5)
    Update $(m_{i\to a}, v_{i\to a})$ using equation (2.8)
  **end for**
**end for**
**for** $i = 1$ **to** $n$ **do**
  Compute $(\tilde{m}_i, \tilde{v}_i)$ using messages $\{(\hat{m}_{b\to i}, \hat{v}_{b\to i}) \,|\, b \in \partial i\}$ similarly as (2.4), (2.5)
  Compute estimation $\hat{x}_i = \eta_i(\tilde{m}_i, \tilde{v}_i)$
**end for**

### 2.3.2 Node-based Algorithms

For this section, we assume that the signal entries are i.i.d., i.e. $p_{X_i} \equiv p_X$, for all $i = 1, \ldots, n$, and $\sigma^2_{W,a} \equiv \sigma^2_W$ for all $a = 1, \ldots, m$. Thus, the problem in (2.1) simplifies into

$$\hat{\mathbf{x}} = \arg\min_{x\in\mathbb{R}^n} \left\{ \frac{1}{2} \|\mathbf{y} - \mathbf{\Phi}\mathbf{x}\|_2^2 + \sigma_W^2 f(\mathbf{x}) \right\}.$$

#### Iterative Shrinkage Thresholding Algorithm

The iterative shrinkage thresholding algorithm originates from the heuristic that alternates between thresholding insignificant contributions and reinforcing $\mathbf{\Phi}\mathbf{x} = \mathbf{y}$. To see this, we first define the proximity operation

$$\mathsf{prox}_{\lambda f}(\mathbf{r}) := \arg\min_{\mathbf{x}\in\mathbb{R}^n} \left\{ \frac{1}{2} \|\mathbf{r} - \mathbf{x}\|_2^2 + \lambda f(\mathbf{x}) \right\}.$$

Then, the general step of ISTA is in the form

$$\begin{aligned} \hat{\mathbf{x}}^{(t+1)} &= \mathsf{prox}_{\lambda f}\left( \hat{\mathbf{x}}^{(t)} - \frac{1}{2}\nabla \left\| \mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}}^{(t)} \right\|_2^2 \right) \\ &= \mathsf{prox}_{\lambda f}\left( \mathbf{\Phi}^{\mathrm{T}}\mathbf{y} + \left( \mathbf{I} - \mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} \right) \hat{\mathbf{x}}^{(t)} \right). \end{aligned}$$

**Algorithm 3** Iterative Shrinkage-Thresholding Algorithm (ISTA)

Initialize: $\hat{\mathbf{x}}^{(0)} = \mathbf{0}$
**for** $t = 0$ **to** $T$ **do**
  Compute residual $\mathbf{v}^{(t)} = \mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}}^{(t)}$
  Apply thresholding $\hat{\mathbf{x}}^{(t+1)} = \mathsf{prox}_{\lambda f}\left(\hat{\mathbf{x}}^{(t)} + \mathbf{\Phi}^{\mathrm{T}}\mathbf{v}^{(t)}\right)$
**end for**

For example, if the signal prior $X_i \overset{\text{i.i.d.}}{\sim} \exp(-|x|)$, i.e. $f(\mathbf{x}) = \|\mathbf{x}\|_1$, then $\mathsf{prox}_{\lambda f}$ is simply the soft-thresholding function with width $\lambda$. Besides, for our case if the measurement noise $\sigma_W^2$ is known, one can set all $\lambda = \sigma_W^2$. For convex $f(\cdot)$, $\|\mathbf{\Phi}\|_2^2 < 1$ ensures convergence. The main drawback of ISTA is its slow convergence which is induced by correlation between the residual and the signal estimate.

### Approximate Message Passing (AMP)

Recall that, if the signal entries are i.i.d., then $f(\mathbf{x}) = -\sum_{i=1}^n \log(p_X(x_i))$ is separable. With the iteration index $t$ as a superscript, define the scalar denoising function, which is parameterized by $\gamma^{(t)}$, for the $i$-th entry to be

$$g(r_i^{(t)}, \gamma^{(t)}) = \arg\min_{x_i \in \mathbb{R}} \left\{ \frac{\gamma^{(t)}}{2} \left| x_i - r_i^{(t)} \right|^2 - \log(p_X(x_i)) \right\}, \quad \forall\, i = 1, \ldots, n.$$

Also, let $\mathbf{g}(\cdot, \gamma^{(t)}) : \mathbb{R}^n \to \mathbb{R}^n$ denote the vector denoising function parameterized by $\gamma^{(t)}$ such that $[\mathbf{g}(\mathbf{r}^{(t)}, \gamma^{(t)})]_i = g(r_i^{(t)}, \gamma^{(t)})$. In addition, let $\mathbf{g}'(\mathbf{r}^{(t)}, \gamma^{(t)}) \in \mathbb{R}^n$ be the diagonal of Jacobian

$$\mathbf{g}'(\mathbf{r}^{(t)}, \gamma^{(t)}) := \mathsf{diag}\left( \frac{\partial \mathbf{g}(\mathbf{r}^{(t)}, \gamma^{(t)})}{\partial \mathbf{r}} \right).$$

Using $\langle \cdot \rangle$ to denote the empirical averaging operation $\langle \mathbf{u} \rangle := (1/n) \sum_{i=1}^n u_i$, the divergence at iteration $t$ can be written as $\left\langle \mathbf{g}'(\mathbf{r}^{(t)}, \gamma^{(t)}) \right\rangle$.

If the measurement matrix $\mathbf{\Phi}$ is i.i.d. sub-Gaussian and the entry denoising function $g(\cdot, \gamma^{(t)})$ is Lipschitz, then the state evolution of AMP algorithm shows that $\mathbf{r}^{(t)}$ behaves like a white-Gaussian-noise corrupted version of the true signal $\mathbf{x}^*$ [BM11]. More precisely, $\mathbf{r}^{(t)} = \mathbf{x}^* + \mathcal{N}(\mathbf{0}, \tau^{(t)}\mathbf{I})$ for some variance $\tau^{(t)} > 0$, where $\tau^{(t)}$ can be computed by state evolution of AMP.

**Algorithm 4** Approximate Message Passing Algorithm (AMP)

Set $\hat{\mathbf{x}}^{(0)} = \mathbf{0}$ and select initial $\mathbf{v}^{(0)}, \mathbf{r}^{(0)}, \gamma^{(0)}$
**for** $t = 1$ **to** $T$ **do**
  Compute divergence $\alpha^{(t)} = \left\langle \mathbf{g}'(\mathbf{r}^{(t-1)}, \gamma^{(t-1)}) \right\rangle$
  Compute Onsager corrected residual $\mathbf{v}^{(t)} = \mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}}^{(t)} + \frac{n}{m}\alpha^{(t)}\mathbf{v}^{(t-1)}$
  Compute "white-Gaussian corrupted" signal $\mathbf{r}^{(t)} = \hat{\mathbf{x}}^{(t)} + \Phi^{\mathrm{T}}\mathbf{v}^{(t)}$
  Select $\gamma^{(t)}$ and apply denoising $\hat{\mathbf{x}}^{(t+1)} = \mathbf{g}(\mathbf{r}^{(t)}, \gamma^{(t)})$
**end for**

Thanks to the Onsager correction term, AMP converges much faster than ISTA when $\mathbf{\Phi}$ is an i.i.d. Gaussian measurement matrix, because the residual and estimation uncorrelated and each iteration more productive. However, AMP is rather fragile and small deviations from the i.i.d. sub-Gaussian model may cause the algorithm to diverge.

### Vector AMP (VAMP)

Unlike the AMP algorithm, whose message-passing derivation uses a loopy factor graph with scalar-valued nodes, the VAMP algorithm uses a cycle-free graph with vector-valued nodes, hence the name "vector AMP".

The derivation of VAMP starts with $p(\mathbf{x}, \mathbf{y}) = p_{\mathbf{x}}(\mathbf{x})\mathcal{N}\left(\mathbf{y} \,\middle|\, \mathbf{\Phi}\mathbf{x}, \gamma_W^{-1}\mathbf{I}\right)$ and then splits $\mathbf{x}$ into two identical variables $\mathbf{x}_1 = \mathbf{x}_2$, yielding an equivalent formation

$$p(\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}) = p_{\mathbf{x}}(\mathbf{x}_1)\delta(\mathbf{x}_1 - \mathbf{x}_2)\mathcal{N}\left(\mathbf{y} \,\middle|\, \mathbf{\Phi}\mathbf{x}_2, \gamma_W^{-1}\mathbf{I}\right)$$

where $\delta(\cdot)$ is the Dirac delta distribution.

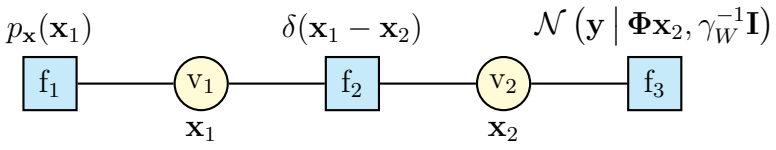



**Figure 2.2**: Factor graph to derive VAMP.

Compared with the tedious proof of its state evolution convergence, the underlying idea of VAMP is quite straightforward. The messages are passed according to the following rules:

1) Approximated beliefs: The approximated belief $b_{\text{app}}(\mathbf{x}_i)$ for a variable-node $\text{v}_i$ is the i.i.d. Gaussian approximation of its sum-product belief $b_{\text{sp}}(\mathbf{x}_i) \propto \prod_j \hat{\mu}_{\text{f}_j \to \text{v}_i}(\mathbf{x}_i)$, i.e.

$$b_{\text{app}}(\mathbf{x}_i) = \mathcal{N}\left(\mathbf{x}_i \,|\, \mathbb{E}\left[\mathbf{x} \,|\, b_{\text{sp}}\right], \langle \mathsf{diag}\left(\mathsf{Cov}\left(\mathbf{x} \,|\, b_{\text{sp}}\right)\right)\rangle \cdot \mathbf{I}\right).$$

2) Variable-to-factor messages: The message from variable-node $\text{v}_i$ to an adjacent factor-node $\text{f}_j$ is

$$\mu_{\text{v}_i \to \text{f}_j}(\mathbf{x}_i) \propto b_{\text{app}}(\mathbf{x}_i) / \hat{\mu}_{\text{f}_j \to \text{v}_i}(\mathbf{x}_i).$$

3) Factor-to-variable messages: The message from factor-node $\text{f}_j$ to a connected variable-node $\text{v}_i$ is

$$\hat{\mu}_{\text{f}_j \to \text{v}_i}(\mathbf{x}_i) \propto \int_{\{\mathbf{x}_{i'}\}_{i' \neq i}} \text{f}_j\left(\mathbf{x}_i, \{\mathbf{x}_{i'}\}_{i' \neq i}\right) \prod_{i' \neq i} \mu_{\text{v}_{i'} \to \text{f}_j}(\text{v}_{i'}) \, \text{d}\mathbf{x}_{i'}.$$

The pseudocode for VAMP is given below. Since the associated factor graph is a chain and there are only two vector variables, for each iteration the algorithm just alternates between updating $\mathbf{x}_1$ and $\mathbf{x}_2$. These steps, respectively, are called the denoising step and the LMMSE step.

**Algorithm 5** Vector AMP (VAMP)

Select $\mathbf{r}_1^{(0)}$ and $\gamma_1^{(0)} > 0$
**for** $t = 0$ **to** $T$ **do**
  `/* Denoising step */`
  Apply denoising $\hat{\mathbf{x}}_1^{(t)} = \mathbf{g}(\mathbf{r}_1^{(t)}, \gamma_1^{(t)})$
  Compute divergence $\alpha_1^{(t)} = \left\langle \mathbf{g}'(\mathbf{r}_1^{(t)}, \gamma_1^{(t)}) \right\rangle$
  Compute Onsager correction $\mathbf{r}_2^{(t)} = \left(\hat{\mathbf{x}}_1^{(t)} - \alpha_1^{(t)} \mathbf{r}_1^{(t)}\right) / (1 - \alpha_1^{(t)})$
  Update precision of LMMSE step $\gamma_2^{(t)} = \gamma_1^{(t)}(1 - \alpha_1^{(t)})/\alpha_1^{(t)}$
  `/* LMMSE step */`
  Compute LMMSE estimate $\hat{\mathbf{x}}_2^{(t)} = \left(\mathbf{\Phi}^{\text{T}}\mathbf{\Phi}/\sigma_W^2 + \gamma_2^{(t)}\mathbf{I}\right)^{-1}\left(\mathbf{\Phi}^{\text{T}}\mathbf{y}/\sigma_W^2 + \gamma_2^{(t)}\mathbf{r}_2^{(t)}\right)$
  Compute divergence $\alpha_2^{(t)} = (\gamma_2^{(t)}/n)\,\mathsf{Tr}\left(\left(\mathbf{\Phi}^{\text{T}}\mathbf{\Phi}/\sigma_W^2 + \gamma_2^{(t)}\mathbf{I}\right)^{-1}\right)$
  Compute Onsager correction $\mathbf{r}_1^{(t+1)} = \left(\hat{\mathbf{x}}_2^{(t)} - \alpha_2^{(t)}\mathbf{r}_2^{(t)}\right)/(1 - \alpha_2^{(t)})$
  Update precision of denoising step $\gamma_1^{(t+1)} = \gamma_2^{(t)}(1 - \alpha_2^{(t)})/\alpha_2^{(t)}$
**end for**

One advantage of VAMP is its theoretical robustness to right-rotational invariant matrices. In contrast, AMP requires i.i.d. sub-Gaussian matrices. For low-complexity implementation, VAMP does require the SVD of the measurement matrix though.

## 2.4 Measurement Matrices

The properties and structure of the measurement matrix can be crucial to the success of a compressive sensing reconstruction algorithm. In the sections below, three different classes of random measurement matrices are introduced. And they will be used later to compare different iterative reconstruction algorithms.

### 2.4.1 Right-Rotationally Invariant (RRI) Matrices

A rectangular random matrix $\mathbf{\Phi}$ is called right-rotationally invariant if the joint distributions of its entries equals that of $\mathbf{\Phi R}$ for any unitary matrix $\mathbf{R}$ independent of $\mathbf{\Phi}$.

In this work, we can construct RRI matrices by using SVD, to write $\mathbf{\Phi} = \mathbf{USV}^{\mathrm{H}}$, where $\mathbf{U}$ is any $m \times m$ (random) unitary matrix, $\mathbf{S}$ is an $m \times n$ diagonal matrix, and $\mathbf{V}$ is an $n \times n$ unitary matrix drawn from Haar measure.

A matrix with i.i.d. Gaussian entries is right-rotationally invariant. It is also easy to generate an RRI matrix with a given conditional number. In the numerical experiments, the entries $s_1, s_2, \ldots, s_m$ of $\mathbf{S}$ are chosen to be a decreasing geometric sequences such that $s_1/s_m$ equals the desired condition number.

### 2.4.2 Gallager-Gaussian (GG) Matrices

A $(w_c, w_r)$-Gallager ensemble [Gal60] is a class of binary random matrices where there are exactly $w_r$ ones on each row and $w_c$ ones on each column.

A $(w_c, w_r, \sigma_1^2, \sigma_0^2)$ GG matrix is defined as following: first draw a binary matrix $\mathbf{G}$ uniformly from the $(w_c, w_r)$-Gallager as the 'mask matrix'. Define the positions where

$G_{ij} = 1$ as the foreground, and the position where $G_{ij} = 0$ as the background. Each entry of the GG matrix $\mathbf{\Phi}$ obeys $\Phi_{ij} \sim \mathcal{N}(0, \sigma^2_{G_{ij}})$.

Since usually $\sigma_1^2 \gg \sigma_0^2$ by construction, if one views the amplitude of entries $\Phi_{ij}$ as a landscape, then it looks like the background noise with sparse peaks. Besides, it is worth noting that when $\sigma_1^2 = \sigma_0^2$, GG matrices degenerate to matrices with i.i.d. Gaussian entries.

### 2.4.3 Sparse-Matrix-Product (SMP) Matrices

Inspired by the fact that fast Fourier transform (FFT) reduces the computation complexity from $O(n^2)$ to $O(n \log(n))$ and can be computed from the butterfly graph with $\log_2(n)$ stages, it can be seen that if a dense matrix can be decomposed into the product of several sparse matrices, then one can utilize the sparsity of the factors to achieve more efficient computation.

In this project, a $(K, \delta)$-SMP matrix $\mathbf{\Phi}$ is constructed by the product of $K$ sparse matrices $\mathbf{F}^{(k)} \in \mathbb{R}^{n_k \times n_{k-1}}$, where $n_K = m, n_0 = n$, and all the factor matrices are sparse with their fraction of nonzero entries to be $\delta$, and all nonzero entries are drawn from $\mathcal{N}(0, 1)$, i.e.

$$\mathbf{\Phi} = \mathbf{F}^{(K)} \mathbf{F}^{(K-1)} \cdots \mathbf{F}^{(2)} \mathbf{F}^{(1)}.$$

Define $\mathsf{nnz}(\cdot)$ to be the number of nonzero entries of a matrix. We will show when $\mathsf{nnz}(\mathbf{\Phi}) \gg \sum_{k=1}^{K} \mathsf{nnz}(\mathbf{F}^{(k)})$, the proposed new algorithm RBP-SEQ is more efficient than the standard RBP algorithm.

## 2.5 SVD-modified RBP (RBP-SVD)

Starting with the compressed sensing problem $\mathbf{y} = \mathbf{\Phi x} + \mathbf{w}$ with $w_a \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_W^2)$, we first apply SVD matrix decomposition to the measurement matrix $\mathbf{\Phi} = \mathbf{U S V}^{\mathrm{T}}$, where $\mathbf{U} \in \mathbb{R}^{m \times m}$, $\mathbf{V} \in \mathbb{R}^{n \times n}$ are unitary matrix, and $\mathbf{S} \in \mathbb{R}^{m \times n}$ is a diagonal matrix. If $\mathbf{\Phi}$ is not full rank, then let $r = \mathsf{rank}(\mathbf{\Phi})$ and define the "effective" SVD decomposition of $\mathbf{\Phi}$ as

$$\mathbf{\Phi} = \mathbf{U S V}^{\mathrm{T}} = \mathbf{U}_{\text{eff}} \mathbf{S}_{\text{eff}} \mathbf{V}_{\text{eff}}^{\mathrm{T}}, \qquad \mathbf{U}_{\text{eff}} \in \mathbb{R}^{m \times r}, \quad \mathbf{S}_{\text{eff}} \in \mathbb{R}^{r \times r}, \quad \mathbf{V}_{\text{eff}} \in \mathbb{R}^{n \times r},$$

where $\mathbf{U}_{\text{eff}}, \mathbf{V}_{\text{eff}}$ are formed by columns of $\mathbf{U}, \mathbf{V}$ whose associated singular values are non-zero.

Left multiplying by $\mathbf{S}_{\text{eff}}^{-1}\mathbf{U}_{\text{eff}}^{\mathrm{T}}$ on both sides of the equation gives

$$\begin{aligned}\mathbf{S}_{\text{eff}}^{-1}\mathbf{U}_{\text{eff}}^{\mathrm{T}}\mathbf{y} &= \mathbf{S}_{\text{eff}}^{-1}\mathbf{U}_{\text{eff}}^{\mathrm{T}}\left(\mathbf{\Phi}\mathbf{x}+\mathbf{w}\right)\\ &= \mathbf{S}_{\text{eff}}^{-1}\mathbf{U}_{\text{eff}}^{\mathrm{T}}\mathbf{U}_{\text{eff}}\mathbf{S}_{\text{eff}}\mathbf{V}_{\text{eff}}^{\mathrm{T}}\mathbf{x}+\mathbf{S}_{\text{eff}}^{-1}\mathbf{U}_{\text{eff}}^{\mathrm{T}}\mathbf{w}\\ &= \mathbf{V}_{\text{eff}}^{\mathrm{T}}\mathbf{x}+\mathbf{S}_{\text{eff}}^{-1}\mathbf{U}_{\text{eff}}^{\mathrm{T}}\mathbf{w}.\end{aligned}$$

Since the measurement noise is distributed as $\mathbf{w}\sim\mathcal{N}(\mathbf{0},\sigma_W^2\mathbf{I}_m)$ and $\mathbf{U}_{\text{eff}}$ has normalized orthogonal columns, the rotational invariant property of multivariate Gaussian distribution implies $\mathbf{S}_{\text{eff}}^{-1}\mathbf{U}_{\text{eff}}^{\mathrm{T}}\mathbf{w}\sim\mathcal{N}(\mathbf{0},\sigma_W^2\mathbf{S}_{\text{eff}}^{-2})$.

Finally, denote $\tilde{\mathbf{y}} = \mathbf{S}_{\text{eff}}^{-1}\mathbf{U}_{\text{eff}}^{\mathrm{T}}\mathbf{y} \in \mathbb{R}^r$, $\tilde{\mathbf{x}} = \mathbf{x}$, and $\tilde{\mathbf{w}} \sim \mathcal{N}(\mathbf{0},\sigma_W^2\mathbf{S}_{\text{eff}}^{-2})$. Then, the reformulated compressed sensing problem is

$$\tilde{\mathbf{y}} = \mathbf{V}_{\text{eff}}^{\mathrm{T}}\tilde{\mathbf{x}}+\tilde{\mathbf{w}}, \qquad \text{where} \quad p_{\tilde{X}_i} = p_{X_i} \quad \text{and} \quad \tilde{w}_a \sim \mathcal{N}(0,\sigma_W^2[\mathbf{S}_{\text{eff}}]_{aa}^{-2}).$$

The RBP-SVD algorithm is completed by running RBP algorithm on this reformulated problem. The advantage of the RBP-SVD algorithm is that, when the measurement matrix $\mathbf{\Phi}$ is ill-conditioned, after the SVD reformulation, the new measurement matrix $\mathbf{V}_{\text{eff}}^{\mathrm{T}}$ is well-conditioned, which greatly improves the convergence behavior of the iteration.

## 2.6 Sequential RBP (RBP-SEQ)

The idea of RBP-SEQ algorithm is originated from using Fast Fourier Transformation to compute Discrete Fourier Transformation. The question is, when computing a full matrix-vector multiplication, if we have side information of about the sparsity structure, can we utilize this side information to reduce the computation complexity?

Consider the compressed sensing problem $\mathbf{y} = \mathbf{\Phi}\mathbf{x}+\mathbf{w}$, where the measurement matrix $\mathbf{\Phi}$ is a SMP matrix with $K$ factors, i.e. $\mathbf{\Phi} = \mathbf{F}^{(K)}\mathbf{F}^{(K-1)}\cdots\mathbf{F}^{(2)}\mathbf{F}^{(1)}$ and for all $k \in \{1,\ldots,K\}$, the matrix $\mathbf{F}^{(k)} \in \mathbb{R}^{n_k\times n_{k-1}}$ is sparse.

For notational simplicity, we define $[n] := \{1, \dots, n\}$. Let $\tilde{\mathbf{x}}^{(1)} := \mathbf{x}$, and $\tilde{\mathbf{x}}^{(k+1)} := \mathbf{F}^{(k)}\tilde{\mathbf{x}}^{(k)}$ for $k \in [K-1]$, so that we have $K$ subproblems:

$$\tilde{\mathbf{y}}^{(k)} := \mathbf{0}_{n_{k+1}\times 1} = \begin{bmatrix}\mathbf{F}^{(k)} & -\mathbf{I}\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{x}}^{(k)} \\ \tilde{\mathbf{x}}^{(k+1)}\end{bmatrix}, \qquad \forall\, k \in [K-1]$$
$$\tilde{\mathbf{y}}^{(K)} := \mathbf{y} = \mathbf{F}^{(K)}\tilde{\mathbf{x}}^{(K)} + \mathbf{w},$$

where the prior information $\tilde{x}_i^{(1)} \sim p_{X_i}$ only affects the first stage, and the measurement noise $w_a \sim \mathcal{N}(0, \sigma_{W,a}^2)$ only affects the last stage. Therefore, in total we have $\sum_{k=0}^{K-1} n_k$ variable nodes $\left\{ \mathrm{v}_i^{(k)} \,\middle|\, k \in [K], i \in [n_k] \right\}$ and $\sum_{k=1}^{K} n_k$ function nodes $\left\{ \mathrm{f}_a^{(k)} \,\middle|\, k \in [K], a \in [n_{k+1}] \right\}$, and we call the resulted factor graph the "unfolded" factor graph.

For stage $k \in [K]$, there are $n_{k-1}$ variable nodes and $n_k$ function nodes, and the edges connecting them are determined by matrix $\mathbf{F}^{(k)}$ with

$$\mathcal{E}^{(k)} := \left\{ (a,i) \in [n_k] \times [n_{k-1}] \,\middle|\, \left[\mathbf{F}^{(k)}\right]_{ai} \neq 0 \right\}, \quad \forall\, k \in [K].$$

We define four types of messages:

- Backward VN-to-FN messages $\mu^{\mathrm{bkd}}_{i^{(k)}\to a^{(k-1)}}$: messages sent from stage $k$ variable nodes $\mathrm{v}_i^{(k)}$ backward to stage $k-1$ function nodes $\mathrm{f}_a^{(k-1)}$ with $i = a \in [n_{k-1}]$. Stage $k = 1$ does not have these types of messages.

- Forward VN-to-FN messages $\mu^{\mathrm{fwd}}_{i^{(k)}\to a^{(k)}}$: messages sent from stage $k$ variable nodes $\mathrm{v}_i^{(k)}$ forward to stage $k$ function nodes $\mathrm{f}_a^{(k)}$ with $(a,i) \in \mathcal{E}^{(k)}$

- Backward FN-to-VN messages $\hat{\mu}^{\mathrm{bkd}}_{a^{(k)}\to i^{(k)}}$: messages sent from stage $k$ function nodes $\mathrm{f}_a^{(k)}$ backward to stage $k$ variable nodes $\mathrm{v}_i^{(k)}$ with $(a,i) \in \mathcal{E}^{(k)}$.

- Forward FN-to-VN messages $\hat{\mu}^{\mathrm{fwd}}_{a^{(k)}\to i^{(k+1)}}$: messages sent from stage $k$ function nodes $\mathrm{f}_a^{(k)}$ forward to stage $k+1$ variable nodes $\mathrm{v}_i^{(k+1)}$ with $i = a \in [n_k]$. Stage $k = K$ does not have these types of messages.

A toy example of the factor graph with $m = 3$, $n = 5$, $K = 3$ and is shown below

$$\mathbf{F}^{(1)} = \begin{bmatrix} 1 & 0 & -2 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & 3 & 0 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 \end{bmatrix}, \quad \mathbf{F}^{(2)} = \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 0 & 1 & 0 \\ 3 & 0 & -1 & 0 \\ 0 & 1 & 4 & 0 \\ 0 & -2 & 0 & -1 \end{bmatrix}, \quad \mathbf{F}^{(3)} = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 \\ 0 & -1 & 0 & 3 & 1 \\ 1 & 0 & -1 & 0 & 2 \end{bmatrix}$$

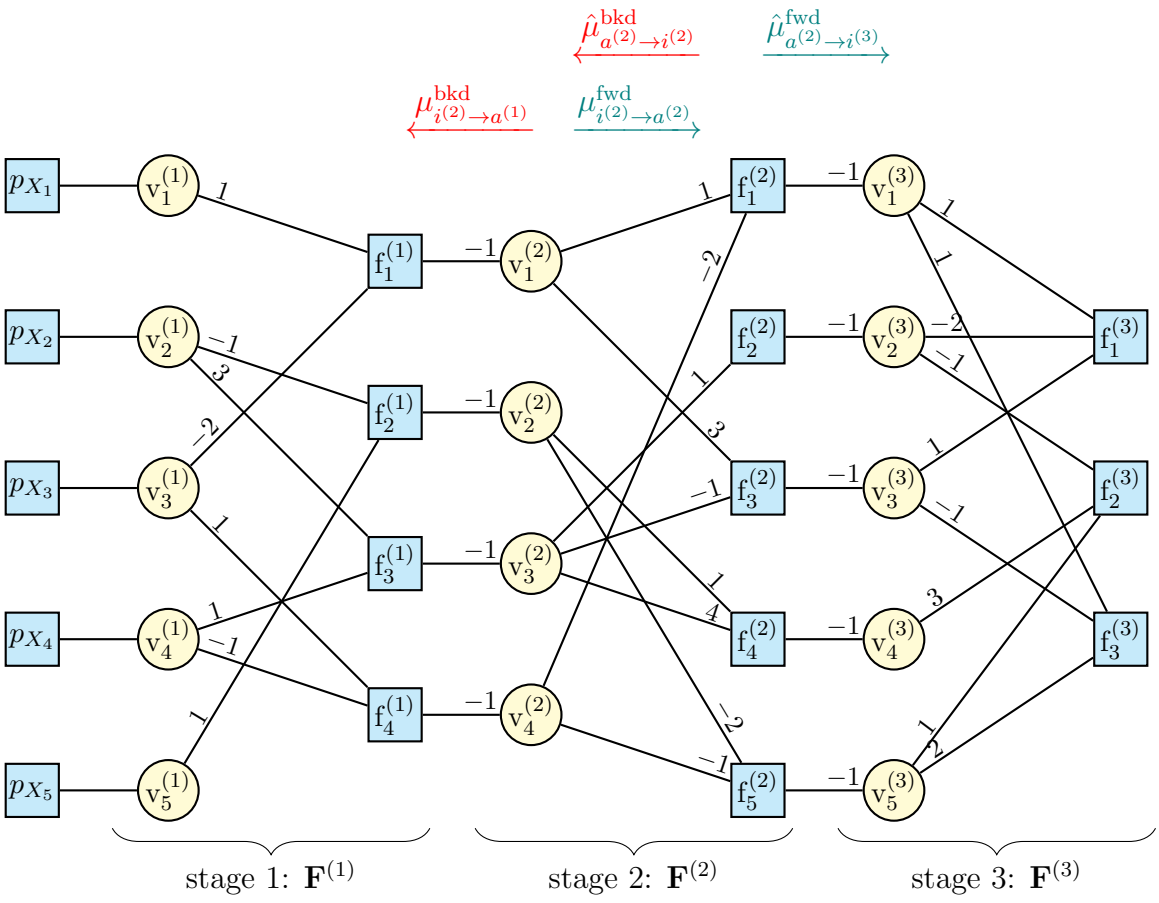



**Figure 2.3**: A toy example of SMP matrix and RBP-SEQ messages

Actually, if there is only one stage, then RBP-SEQ degenerates into standard RBP. The RBP-SEQ algorithm runs the RBP algorithm on this "unfolded" factor graph, but with a slightly different updating scheduling. Instead of updating all VN-to-FN (or FN-to-VN)

messages simultaneously, RBP-SEQ updates them stage by stage. One iteration of the RBP-SEQ algorithm is composed of a 'stage-wise round trip', i.e. a forward sweep and a backward sweep: all the forward messages are updated from stage 1 to stage $K$, and then all the backward messages are updated from stage $K$ to stage 1.

With a little abuse of notation, let $\mu$ denote the mean/variance pair instead of the exact distribution. Let all the "update" in the following pseudocode follow the RBP update equations. Then we have the following algorithm.

**Algorithm 6** RBP-SEQ Algorithm

Initialize all four types of messages randomly.
**for** $t = 1$ **to** $T$ **do**
  `/* Forward Sweep */`
  Update $\mu^{\text{fwd}}_{i^{(1)}\to a^{(1)}}$ using $\hat{\mu}^{\text{bkd}}_{a^{(1)}\to i^{(1)}}$ and $p_{X_i}$, $\forall\,(i,a)\in\mathcal{E}^{(1)}$
  **for** $k = 2$ **to** $K$ **do**
    Update $\hat{\mu}^{\text{fwd}}_{a^{(k-1)}\to i^{(k)}}$ using $\mu^{\text{fwd}}_{i^{(k-1)}\to a^{(k-1)}}$, $\forall\, a = i \in [n_{k-1}]$.
    Update $\mu^{\text{fwd}}_{i^{(k)}\to a^{(k)}}$ using $\hat{\mu}^{\text{bkd}}_{a^{(k)}\to i^{(k)}}$ and $\hat{\mu}^{\text{fwd}}_{a^{(k-1)}\to i^{(k)}}$, $\forall\,(i,a)\in\mathcal{E}^{(k)}$.
  **end for**
  `/* Backward Sweep */`
  Update $\hat{\mu}^{\text{bkd}}_{a^{(K)}\to i^{(K)}}$ using $\mu^{\text{fwd}}_{i^{(K)}\to a^{(K)}}$ and $p_{W_a}$, $\forall\,(i,a)\in\mathcal{E}^{(K)}$.
  **for** $k = K-1$ **to** $1$ **do**
    Update $\mu^{\text{bkd}}_{i^{(k+1)}\to a^{(k)}}$ using $\hat{\mu}^{\text{bkd}}_{a^{(k+1)}\to i^{(k+1)}}$, $\forall\, a = i \in [n_k]$.
    Update $\hat{\mu}^{\text{bkd}}_{a^{(k)}\to i^{(k)}}$ using $\mu^{\text{fwd}}_{i^{(k)}\to a^{(k)}}$ and $\mu^{\text{bkd}}_{i^{(k+1)}\to a^{(k)}}$, $\forall\,(i,a)\in\mathcal{E}^{(k)}$.
  **end for**
**end for**
**for** $i = 1$ **to** $n$ **do**
  Estimate $\hat{x}_i$ using $\left\{\hat{\mu}^{\text{bkd}}_{a^{(1)}\to i^{(1)}}\,\middle|\,(a,i)\in\mathcal{E}^{(1)}\right\}$ and $p_{X_i}$.
**end for**

The complexity of RBP algorithm is proportional to the number of messages. And, for SMP matrices, we have the assumption that each stage $\mathbf{F}^{(k)}$ is sparse with $\mathsf{nnz}(\mathbf{\Phi}) \gg \sum_{k=1}^{K}\mathsf{nnz}(\mathbf{F}^{(k)})$. Thus, the RBP-SEQ algorithm has much fewer messages than standard RBP and much lower computational complexity. If damping is needed, it is added only to the first and last stages where the algorithm has a 'U-turn'.

## 2.7 Experimental Results

Now, we compare the performance of RBP-SVD and RBP-SEQ with RBP, AMP, VAMP and SpaRSA [WNF09a]. For this comparison, we choose the signal to be Bernoulli-Gaussian with sparsity $\epsilon = 0.1$ and $\sigma_1^2 = 10$ such that $\mathsf{Var}(X) = 1$, i.e.

$$p_{X_i}(x) \equiv p_X(x) = (1-\epsilon)\,\delta_0(x) + \epsilon\,\mathcal{N}\left(x \,\middle|\, 0, \sigma_1^2\right).$$

The dimensions of the measurement and signal vectors are $m = 2^9$ and $n = 2^{10}$. Also, the measurement matrices have size $m \times n$, and are normalized so that their Frobenius norm equals $n$. Simulations are run with ten different measurement noise levels so that the SNR ranges geometrically from 5dB to 40dB. All plot points are averaged over 500 random matrices, with 10 random signals per matrix. All algorithms terminated when the relative error meets a stopping criterion, or the iteration number reaches 30. To improve stability, damping with $\alpha = 0.95$ is applied to all iterative reconstruction algorithms. In other words, for all messages at iteration $t$, the message updating rule is $\mu^{(t)} = \alpha\,\mu_{\text{new}} + (1-\alpha)\,\mu^{(t-1)}$, where $\mu_{\text{new}}$ is computed using the standard update equations.

### 2.7.1 Stability of RBP-SVD

To demonstrate that RBP-SVD has good performance on a wide variety of measurement matrices, the median NMSE is chosen as the metric. For RRI matrices, $\mathbf{U}, \mathbf{V}$ are Haar distributed, and we choose $\kappa(\mathbf{\Phi}) = 100$, so that the measurement matrix is not well-conditioned.

It can be seen that the NMSE curve for standard RBP disappears in the plot since standard RBP diverges in this case and its NMSE curve is far above. Similarly, the NMSE of AMP does not decrease with the SNR. But, the curve of RBP-SVD overlaps with VAMP, which is specifically designed for RRI matrices. Therefore, RBP-SVD has much better convergence behavior than standard RBP when the measurement matrix is ill-conditioned.

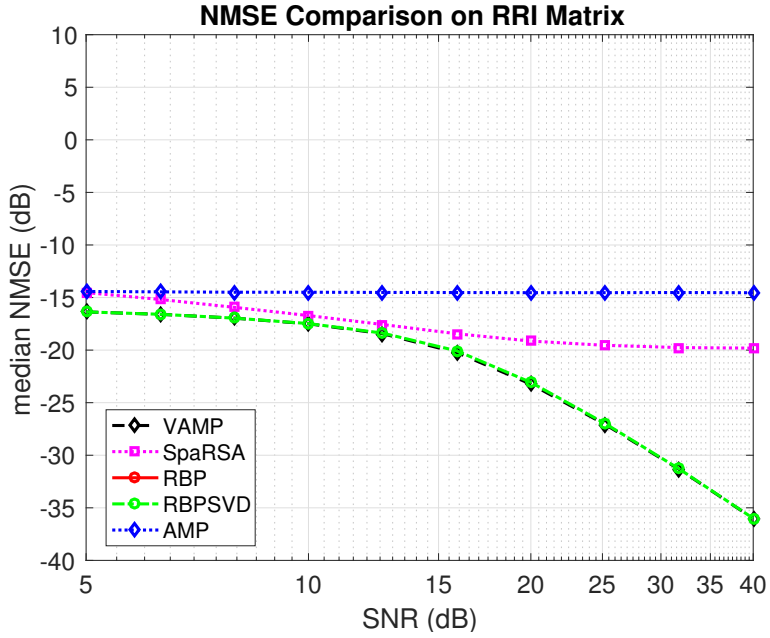



**Figure 2.4**: Simulation results for RRI matrices

For GG matrices, the measurement matrix has size $510 \times 1020$, the parameters are chosen to be $(6, 12, 0.25, 2.5 \times 10^{-3})$, which means the 'mask matrix' is drawn from regular $(6, 12)$-Gallager ensemble, entries are $\mathcal{N}(0, 0.25)$ at positions where the mask matrix has a one, and are $\mathcal{N}(0, 0.25 \times 10^{-3})$ at positions where the mask matrix has a zero.

For GG matrices, AMP does not converge at all, VAMP has a higher median NMSE than standard RBP and RBP-SVD. This is because the amplitudes of entries in GG matrices vary widely between foreground and background and therefore the random matrix is far from right-rotationally invariant. Additionally, it is worth noting that the median NMSE curve of RBP-SVD matches standard RBP, but the mean NMSE for standard RBP is highly above and not shown in the plot, which is because the standard RBP diverges at some case. Experiments on other Gallager 'mask' matrix ensembles from $(3, 6)$ to $(8, 16)$ also shows similar behavior. These results indicate that RBP-SVD is the best algorithm in this setting.

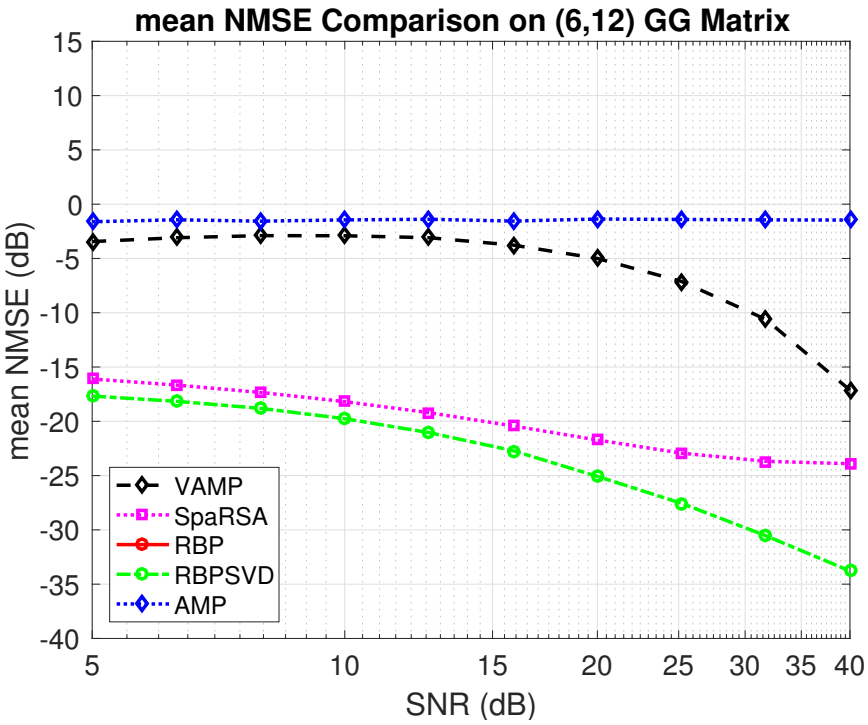



**Figure 2.5**: Simulation results for GG matrices in the metric mean NMSE

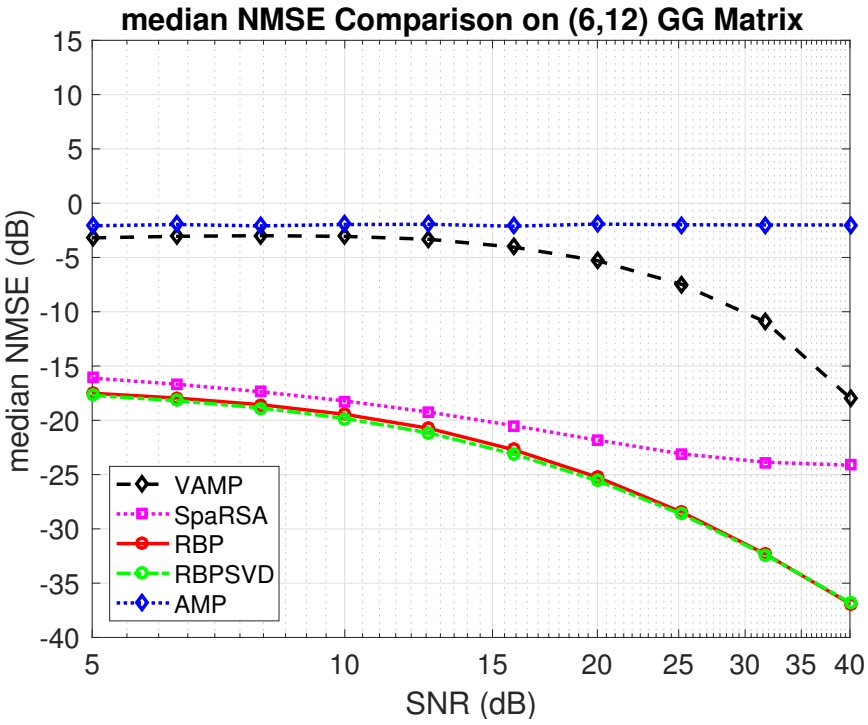



**Figure 2.6**: Simulation results for GG matrices in the metric median NMSE

### 2.7.2 Complexity of RBP-SEQ

To apply RBP-SEQ algorithm, the measurement matrix must have the SMP structure.

For SMP matrices, we set the number of factor matrices to be $K = 2$, $\mathbf{F}^{(2)} \in \mathbb{R}^{m \times m}$, $\mathbf{F}^{(1)} \in \mathbb{R}^{m \times n}$, and for each factor matrix we set the fraction of nonzero entries to be $\delta = 0.08$.

Again, the curve for standard RBP is slightly above while AMP fails to converge. And VAMP does not perform well. SparRSA performs stable for all cases in NMSE but its performance is not outstanding, except for its fast speed. Both RBP-SVD and RBP-SEQ work well, RBP-SEQ is more than three times faster. Additional simulations show that when size of the measurement matrix increases to $2^{11} \times 2^{12}$, RBP-SEQ is roughly 30 times faster than RBP-SVD.

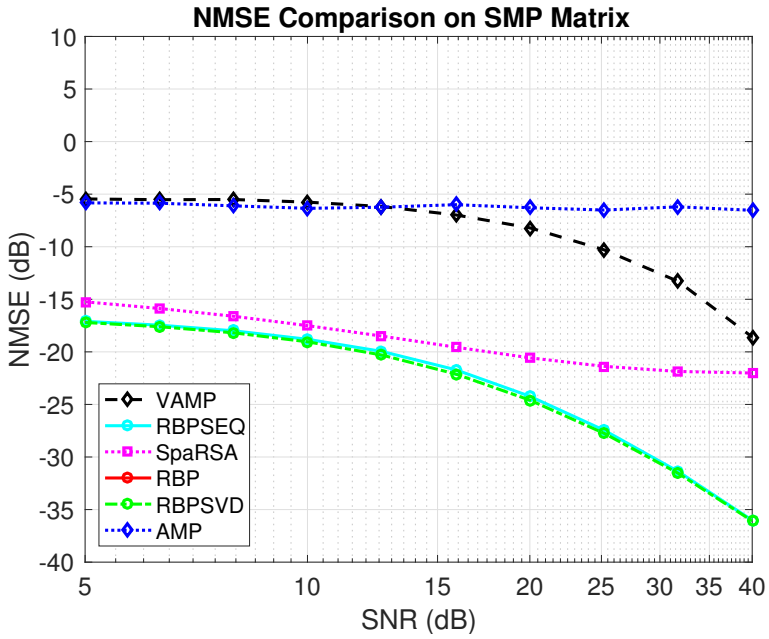



**Figure 2.7**: Simulation result on SMP matrices

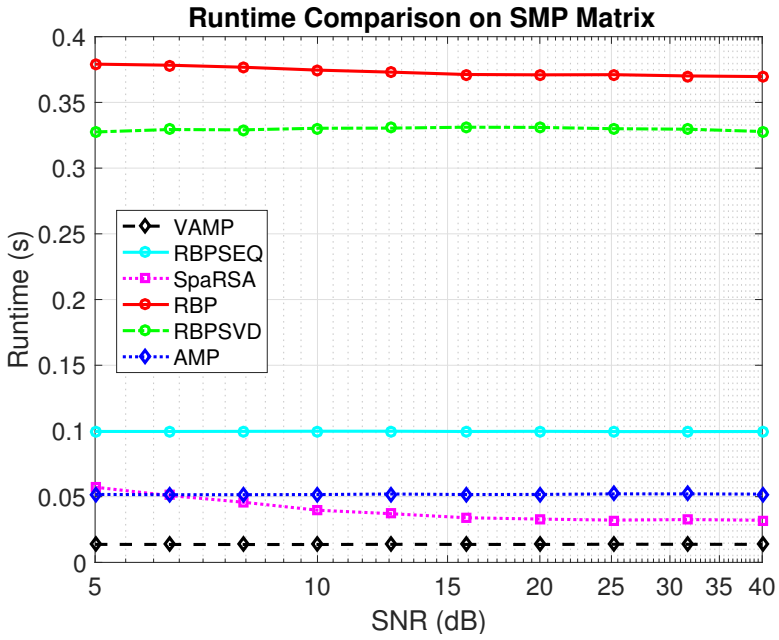



**Figure 2.8**: Simulation runtime on SMP matrices

## 2.8 Conclusion

In this project, two new variants of the RBP algorithm: RBP-SVD and RBP-SEQ are introduced. In comparison with RBP, these new algorithms either improve the stability or to reduce the computation complexity. They are compared with various existing iterative reconstruction algorithms over three classes of random matrices.

RBP-SVD has convergence behavior similar to VAMP on RRI matrices, whereas RBP and AMP do not converge when the matrix is ill-conditioned.

AMP does not converge on GG matrices and both RBP and RBP-SVD outperform VAMP. RBP-SVD and standard RBP both work well in terms of metric median NMSE, but RBP-SVD has much lower mean NMSE than standard RBP because of its improved convergence behavior.

RBP-SEQ, can take advantage of the sparse structure and reduce the computation complexity. For SMP matrices, RBP diverges, while RBP-SEQ behaves similar to RBP-SVD but is three times faster and outperforms VAMP in NMSE.

In summary, the RBP-SVD algorithm greatly improves the convergence behavior of the standard RBP algorithm and has better performance than other compared algorithms. When the measurement matrix is an SMP matrix, the RBP-SEQ algorithm leverages the sparsity of its factor matrices to reduce the computation complexity. However, RBP-SEQ cannot be applied to general measurement matrices.

# Chapter 3

# Covariance Backpropagation for Optical Communication

## 3.1 Introduction

In recent years, the amount of data carried by optical fiber has increased rapidly because of growing demands from bandwidth-intense multimedia applications. For this reason, more powerful signal detection methods are needed to overcome linear and nonlinear fiber impairments that affect transmitted signals.

Digital back propagation (DBP) [IK08], [RMF+11], which ignores noise from optical amplifiers and back solves the nonlinear Schrödinger equation, has been used as a benchmark of many papers. An extension of this, called stochastic digital back propagation (SDBP) [IWJA14], [WIS+15], uses a particle filter to approximate the posterior distribution. With the information about covariance between symbols, the Viterbi algorithm and decision-directed approaches are proposed to improve symbol detection [IMJ+16].

In this chapter, we model the posterior distribution in DBP as a multivariate complex Gaussian distribution and investigate how it evolves during DBP. Using the split-step Fourier method (SSFM) for DBP, the main difficulty is the derivation of the update rules for the covariance matrix. After this, we compare the symbol detection performance between SDBP and covariance back-propagation.

## 3.2 Preliminary

In this chapter, we use $\overline{\mathbf{A}}$, $\mathbf{A}^{\mathsf{T}}$, $\mathbf{A}^{\mathsf{H}}$ to denote the element-wise conjugation, matrix transpose, and matrix conjugate transpose, respectively. Also, both $\mathsf{Re}\,(x)$ and $x_{\mathrm{r}}$ are used as real part

of $x$. Similarly, $\mathsf{Im}\,(x)$ and $x_{\mathrm{i}}$ are used to denote the imaginary part of $x$.

### 3.2.1 The Uparrow Notation

When dealing with complex-valued random variables, it can be more convenient to view a complex scalar as a length-2 real-valued vector. This is especially true when there is correlation between the real and imaginary pars. We define an uparrow notation in order to extend this idea into vectors and matrices.

#### Uparrow Notation for Vectors

For vectors, consider $\mathbf{x} \in \mathbb{C}^n$, one can split into real part and imaginary part as

$$\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}^{\mathsf{T}} = \begin{bmatrix} x_{1,\mathrm{r}} & \cdots & x_{n,\mathrm{r}} \end{bmatrix}^{\mathsf{T}} + \mathbb{i} \begin{bmatrix} x_{1,\mathrm{i}} & \cdots & x_{n,\mathrm{i}} \end{bmatrix}^{\mathsf{T}}.$$

Then the uparrow notation of the complex vector is defined as

$$\mathbf{x}^{\uparrow} \triangleq \begin{bmatrix} x_1^{\uparrow} \\ \vdots \\ x_n^{\uparrow} \end{bmatrix} = \begin{bmatrix} x_{1,\mathrm{r}} \\ x_{1,\mathrm{i}} \\ \vdots \\ x_{n,\mathrm{r}} \\ x_{n,\mathrm{i}} \end{bmatrix} \in \mathbb{R}^{2n}. \tag{3.1}$$

#### Uparrow Notation for Matrices

For matrices, consider $\mathbf{M} \in \mathbb{C}^{m \times n}$, define

$$\mathbf{M}^{\Uparrow} \triangleq \begin{bmatrix} M_{1,1}^{\Uparrow} & \cdots & M_{1,n}^{\Uparrow} \\ \vdots & \ddots & \vdots \\ M_{m,1}^{\Uparrow} & \cdots & M_{m,n}^{\Uparrow} \end{bmatrix} \tag{3.2}$$

$$= \begin{bmatrix} \mathsf{Re}(M_{1,1}) & -\mathsf{Im}(M_{1,1}) & \cdots & \mathsf{Re}(M_{1,n}) & -\mathsf{Im}(M_{1,n}) \\ \mathsf{Im}(M_{1,1}) & \mathsf{Re}(M_{1,1}) & & \mathsf{Im}(M_{1,n}) & \mathsf{Re}(M_{1,n}) \\ \vdots & & \ddots & \vdots & \\ \mathsf{Re}(M_{m,1}) & -\mathsf{Im}(M_{m,1}) & \cdots & \mathsf{Re}(M_{m,n}) & -\mathsf{Im}(M_{m,n}) \\ \mathsf{Im}(M_{m,1}) & \mathsf{Re}(M_{m,1}) & & \mathsf{Im}(M_{m,n}) & \mathsf{Re}(M_{m,n}) \end{bmatrix} \in \mathbb{R}^{2m \times 2n}. \tag{3.3}$$

**Properties**

(i) real and imaginary parts can be extracted as sub-vector or sub-matrix:

$$\begin{aligned}\mathsf{Re}\left(\mathbf{x}\right) &= \left[\mathbf{x}^{\uparrow}\right]_{1:2:n}, \qquad \mathsf{Im}\left(\mathbf{x}\right) = \left[\mathbf{x}^{\uparrow}\right]_{2:2:n} \\ \mathsf{Re}\left(\mathbf{M}\right) &= \left[\mathbf{M}^{\hat{\uparrow}}\right]_{1:2:m,1:2:n} = \left[\mathbf{M}^{\hat{\uparrow}}\right]_{2:2:m,2:2:n} \\ \mathsf{Im}\left(\mathbf{M}\right) &= \left[\mathbf{M}^{\hat{\uparrow}}\right]_{2:2:m,1:2:n} = -\left[\mathbf{M}^{\hat{\uparrow}}\right]_{1:2:m,2:2:n},\end{aligned}$$

where the subscripts are in MATLAB convention.

(ii) uparrow of Hermitian conjugate equals to transpose of uparrow

$$\left[\mathbf{M}^{\mathsf{H}}\right]^{\hat{\uparrow}} = \left[\mathbf{M}^{\hat{\uparrow}}\right]^{\mathsf{T}}.$$

(iii) matrix-vector multiplication and matrix-matrix multiplication are preserved:

$$\mathbf{M}^{\hat{\uparrow}}\mathbf{x}^{\uparrow} = \left[\mathbf{M}\mathbf{x}\right]^{\uparrow}, \qquad \mathbf{M}_1^{\hat{\uparrow}}\mathbf{M}_2^{\hat{\uparrow}} = \left[\mathbf{M}_1\mathbf{M}_2\right]^{\hat{\uparrow}}.$$

(iv) it is easy to map between separate and interleaved orders:

$$\mathbf{\Pi}\mathbf{x}^{\uparrow} = \begin{bmatrix}\mathsf{Re}\left(\mathbf{x}\right)\\ \mathsf{Im}\left(\mathbf{x}\right)\end{bmatrix} = \begin{bmatrix}\mathbf{I}_n \otimes \begin{bmatrix}1 & 0\end{bmatrix}\\ \mathbf{I}_n \otimes \begin{bmatrix}0 & 1\end{bmatrix}\end{bmatrix} \begin{bmatrix}\mathsf{Re}(x_1)\\ \mathsf{Im}(x_1)\\ \vdots\\ \mathsf{Re}(x_n)\\ \mathsf{Im}(x_n)\end{bmatrix} \quad \Rightarrow \quad \mathbf{\Pi} = \begin{bmatrix}\mathbf{I}_n \otimes \begin{bmatrix}1 & 0\end{bmatrix}\\ \mathbf{I}_n \otimes \begin{bmatrix}0 & 1\end{bmatrix}\end{bmatrix}. \tag{3.4}$$

### 3.2.2 Complex Gaussian Distribution

A complex-valued length-$n$ random vector can be viewed as a length-$2n$ real-valued random vector. However, when defining the random variables as complex-valued, dependencies exist not only between different symbols, but also between real and imaginary parts of the same symbol. As a consequence, the multivariate complex Gaussian distribution [van95] is far from a trivial extension of the well-known real-valued multivariate Gaussian distribution.

**Definition 1** (Complex Gaussian Distribution)**.** *Consider* $\mathbf{X} = \begin{bmatrix}X_1 & \cdots & X_n\end{bmatrix}^{\mathsf{T}}$ *and* $\mathbf{Y} = \begin{bmatrix}Y_1 & \cdots & Y_n\end{bmatrix}^{\mathsf{T}}$ *are random vectors in* $\mathbb{R}^n$ *such that* $\left[\mathbf{X}^{\mathsf{T}}, \mathbf{Y}^{\mathsf{T}}\right]^{\mathsf{T}}$ *is a length-*$2n$ *Gaussian*

*random vector. Then we say the complex vector* $\mathbf{Z} = \mathbf{X} + \mathbb{i}\mathbf{Y}$ *is a complex Gaussian random vector. The complex Gaussian distribution can be described with 3 parameters:*

- *location parameter,* $\boldsymbol{\mu} = \mathbb{E}[\mathbf{Z}]$.
- *covariance matrix,* $\boldsymbol{\Gamma} = \mathsf{Cov}(\mathbf{Z}) = \mathbb{E}\left[(\mathbf{Z} - \boldsymbol{\mu})(\mathbf{Z} - \boldsymbol{\mu})^{\mathsf{H}}\right]$.
- *pseudo-covariance matrix,* $\mathbf{C} = \mathsf{Cov}(\mathbf{Z}, \overline{\mathbf{Z}}) = \mathbb{E}\left[(\mathbf{Z} - \boldsymbol{\mu})(\mathbf{Z} - \boldsymbol{\mu})^{\mathsf{T}}\right]$.

*In this case, we say that* $\mathbf{Z}$ *is a the complex normal random vector and this is denoted by*

$$\mathbf{Z} \sim \mathcal{CN}(\boldsymbol{\mu}, \boldsymbol{\Gamma}, \mathbf{C}).$$

*Moreover, matrices* $\boldsymbol{\Gamma}$ *and* $\mathbf{C}$ *will satisfy condition that the matrix* $\mathbf{P} \triangleq \overline{\boldsymbol{\Gamma}} - \mathbf{C}^{\mathsf{H}}\boldsymbol{\Gamma}^{-1}\mathbf{C}$ *is positive-semi definite.*

### Relationship between Covariance Matrices

Let $\mathbf{V_{XX}}, \mathbf{V_{XY}}, \mathbf{V_{YX}}, \mathbf{V_{YY}}$ denote the covariance matrices between real part and imaginary part of $Z$, we obtain

$$\mathbf{V_{XX}} \triangleq \mathbb{E}\left[(\mathsf{Re}(\mathbf{Z}) - \mathsf{Re}(\boldsymbol{\mu}))(\mathsf{Re}(\mathbf{Z}) - \mathsf{Re}(\boldsymbol{\mu}))^{\mathsf{T}}\right] = \frac{1}{2}\mathsf{Re}(\boldsymbol{\Gamma} + \mathbf{C}), \tag{3.5}$$

$$\mathbf{V_{XY}} \triangleq \mathbb{E}\left[(\mathsf{Re}(\mathbf{Z}) - \mathsf{Re}(\boldsymbol{\mu}))(\mathsf{Im}(\mathbf{Z}) - \mathsf{Im}(\boldsymbol{\mu}))^{\mathsf{T}}\right] = \frac{1}{2}\mathsf{Im}(-\boldsymbol{\Gamma} + \mathbf{C}), \tag{3.6}$$

$$\mathbf{V_{YX}} \triangleq \mathbb{E}\left[(\mathsf{Im}(\mathbf{Z}) - \mathsf{Im}(\boldsymbol{\mu}))(\mathsf{Re}(\mathbf{Z}) - \mathsf{Re}(\boldsymbol{\mu}))^{\mathsf{T}}\right] = \frac{1}{2}\mathsf{Im}(\boldsymbol{\Gamma} + \mathbf{C}), \tag{3.7}$$

$$\mathbf{V_{YY}} \triangleq \mathbb{E}\left[(\mathsf{Im}(\mathbf{Z}) - \mathsf{Im}(\boldsymbol{\mu}))(\mathsf{Im}(\mathbf{Z}) - \mathsf{Im}(\boldsymbol{\mu}))^{\mathsf{T}}\right] = \frac{1}{2}\mathsf{Re}(\boldsymbol{\Gamma} - \mathbf{C}). \tag{3.8}$$

Define matrix $\boldsymbol{\Omega}$ such that

$$\begin{bmatrix}\mathbf{Z} \\ \overline{\mathbf{Z}}\end{bmatrix} = \begin{bmatrix}\mathsf{Re}(\mathbf{Z}) + \mathbb{i}\,\mathsf{Im}(\mathbf{Z}) \\ \mathsf{Re}(\mathbf{Z}) - \mathbb{i}\,\mathsf{Im}(\mathbf{Z})\end{bmatrix} = \boldsymbol{\Omega}\begin{bmatrix}\mathsf{Re}(\mathbf{Z}) \\ \mathsf{Im}(\mathbf{Z})\end{bmatrix}. \tag{3.9}$$

Then, we get

$$\boldsymbol{\Omega} = \begin{bmatrix} \mathbf{I}_n & \mathbb{i}\,\mathbf{I}_n \\ \mathbf{I}_n & -\mathbb{i}\,\mathbf{I}_n \end{bmatrix} = \begin{bmatrix} 1 & \mathbb{i} \\ 1 & -\mathbb{i} \end{bmatrix} \otimes \mathbf{I}_n, \tag{3.10}$$

$$\boldsymbol{\Omega}^{-1} = \begin{bmatrix} 1 & \mathbb{i} \\ 1 & -\mathbb{i} \end{bmatrix}^{-1} \otimes \mathbf{I}_n^{-1} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -\mathbb{i} & \mathbb{i} \end{bmatrix} \otimes \mathbf{I}_n = \frac{1}{2}\boldsymbol{\Omega}^{\mathsf{H}}, \tag{3.11}$$

$$\det(\boldsymbol{\Omega}) = \left[\det\left(\begin{bmatrix} 1 & \mathbb{i} \\ 1 & -\mathbb{i} \end{bmatrix}\right)\right]^n [\det(\mathbf{I}_n)]^2 = 2^n \mathbb{i}^{-n}. \tag{3.12}$$

Using (3.4) and (3.9), the covariance matrix of of $\mathbf{Z}^{\uparrow}$ is given by

$$\begin{aligned} \mathsf{Cov}\left(\mathbf{Z}^{\uparrow}\right) &= \mathsf{Cov}\left(\begin{bmatrix} \mathsf{Re}(Z_1) \\ \mathsf{Im}(Z_1) \\ \vdots \\ \mathsf{Re}(Z_n) \\ \mathsf{Im}(Z_n) \end{bmatrix}\right) = \mathsf{Cov}\left(\boldsymbol{\Pi}^{\mathsf{T}} \begin{bmatrix} \mathsf{Re}\,(\mathbf{Z}) \\ \mathsf{Im}\,(\mathbf{Z}) \end{bmatrix}\right) \\ &= \mathsf{Cov}\left(\frac{1}{2}\boldsymbol{\Pi}^{\mathsf{T}}\boldsymbol{\Omega}^{\mathsf{H}} \begin{bmatrix} Z \\ \overline{Z} \end{bmatrix}\right) = \frac{1}{4}\boldsymbol{\Pi}^{\mathsf{T}}\boldsymbol{\Omega}^{\mathsf{H}} \begin{bmatrix} \mathsf{Cov}(\mathbf{Z},\mathbf{Z}) & \mathsf{Cov}(\mathbf{Z},\overline{\mathbf{Z}}) \\ \mathsf{Cov}(\overline{\mathbf{Z}},\mathbf{Z}) & \mathsf{Cov}(\overline{\mathbf{Z}},\overline{\mathbf{Z}}) \end{bmatrix} \boldsymbol{\Omega}\boldsymbol{\Pi} \\ &= \frac{1}{4}\boldsymbol{\Pi}^{\mathsf{T}}\boldsymbol{\Omega}^{\mathsf{H}} \begin{bmatrix} \boldsymbol{\Gamma} & \mathbf{C} \\ \overline{\mathbf{C}} & \overline{\boldsymbol{\Gamma}} \end{bmatrix} \boldsymbol{\Omega}\boldsymbol{\Pi}. \end{aligned} \tag{3.13}$$

### Probability Density Function

For $\mathbf{z} \in \mathbb{C}^n$, the probability density function of a complex Gaussian distribution is given by

$$f(\mathbf{z}) = \frac{1}{\pi^n \sqrt{\det(\boldsymbol{\Gamma})\det(\mathbf{P})}} \exp\left(-\frac{1}{2}\begin{bmatrix} \mathbf{z}-\boldsymbol{\mu} \\ \overline{\mathbf{z}}-\overline{\boldsymbol{\mu}} \end{bmatrix}^{\mathsf{H}} \begin{bmatrix} \boldsymbol{\Gamma} & \mathbf{C} \\ \overline{\mathbf{C}} & \overline{\boldsymbol{\Gamma}} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{z}-\boldsymbol{\mu} \\ \overline{\mathbf{z}}-\overline{\boldsymbol{\mu}} \end{bmatrix}\right) \tag{3.14}$$

Since inverting (3.13) yields

$$\left[\mathsf{Cov}(\mathbf{Z}^{\uparrow})\right]^{-1} = 4\boldsymbol{\Pi}^{\mathsf{T}}\frac{1}{2}\boldsymbol{\Omega}^{\mathsf{H}} \begin{bmatrix} \boldsymbol{\Gamma} & \mathbf{C} \\ \overline{\mathbf{C}} & \overline{\boldsymbol{\Gamma}} \end{bmatrix}^{-1} \frac{1}{2}\boldsymbol{\Omega}\boldsymbol{\Pi} = \boldsymbol{\Pi}^{\mathsf{T}}\boldsymbol{\Omega}^{\mathsf{H}} \begin{bmatrix} \boldsymbol{\Gamma} & \mathbf{C} \\ \overline{\mathbf{C}} & \overline{\boldsymbol{\Gamma}} \end{bmatrix}^{-1} \boldsymbol{\Omega}\boldsymbol{\Pi}, \tag{3.15}$$

One can also think of the $n$-dimensional complex Gaussian distribution as a $2n$-dimensional Gaussian distribution with density, i.e. for $\mathbf{z}^{\uparrow} \in \mathbb{R}^{2n}$,

$$f(\mathbf{z}^{\uparrow}) = \frac{1}{(2\pi)^n \sqrt{\det(\boldsymbol{\Gamma})\det(\mathbf{P})}} \exp\left(-\frac{1}{2}\left[\mathbf{z}^{\uparrow}\right]^{\mathsf{T}} \boldsymbol{\Pi}^{\mathsf{T}}\boldsymbol{\Omega}^{\mathsf{H}} \begin{bmatrix} \boldsymbol{\Gamma} & \mathbf{C} \\ \overline{\mathbf{C}} & \overline{\boldsymbol{\Gamma}} \end{bmatrix}^{-1} \boldsymbol{\Omega}\boldsymbol{\Pi}\,\mathbf{z}^{\uparrow}\right). \tag{3.16}$$

For a univariate complex Gaussian distribution $\mathcal{CN}(\mu, \Gamma, C)$, we have $\Gamma \in \mathbb{R}$ and thus

$$f(z) = \frac{1}{\pi\sqrt{|\Gamma|^2 - |C|^2}} \exp\left(-\frac{\Gamma|z-\mu|^2 - \overline{C}\mathsf{Re}\left((z-\mu)^2\right)}{|\Gamma|^2 - |C|^2}\right).$$

**Properties**

(i) Linear transformation: if $\mathbf{Z} \in \mathbb{C}^n, \mathbf{A} \in \mathbb{C}^{m\times n}, \mathbf{b} \in \mathbb{C}^m$ and $\mathbf{Z} \sim \mathcal{CN}(\boldsymbol{\mu}, \boldsymbol{\Gamma}, \mathbf{C})$, then the linear transformation $\mathbf{AZ} + \mathbf{b}$ has the distribution $\mathcal{CN}\left(\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{A}\boldsymbol{\Gamma}\mathbf{A}^{\mathsf{H}}, \mathbf{A}\mathbf{C}\mathbf{A}^{\mathsf{T}}\right)$.

(ii) Independent sum: if $\mathbf{Z}_1 \sim \mathcal{CN}(\boldsymbol{\mu}_1, \boldsymbol{\Gamma}_1, \mathbf{C}_1)$ and $\mathbf{Z}_2 \sim \mathcal{CN}(\boldsymbol{\mu}_2, \boldsymbol{\Gamma}_2, \mathbf{C}_2)$ are independent, then $\mathbf{Z}_1 + \mathbf{Z}_2$ has the distribution $\mathcal{CN}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2, \boldsymbol{\Gamma}_1 + \boldsymbol{\Gamma}_2, \mathbf{C}_1 + \mathbf{C}_2)$.

(iii) The matrix

$$\begin{bmatrix} \boldsymbol{\Gamma} & \mathbf{C} \\ \overline{\mathbf{C}} & \overline{\boldsymbol{\Gamma}} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{R} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Gamma} & \mathbf{0} \\ \mathbf{0} & \mathbf{P} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{R}^{\mathsf{H}} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{3.17}$$

with

$$\mathbf{R} \triangleq \mathbf{C}^{\mathsf{H}}\boldsymbol{\Gamma}^{-1}, \quad \mathbf{P} \triangleq \overline{\boldsymbol{\Gamma}} - \mathbf{C}^{\mathsf{H}}\boldsymbol{\Gamma}^{-1}\mathbf{C}.$$

It is worth noting that if we denote $\tilde{\mathbf{Z}} \triangleq \overline{\mathbf{Z}} - \mathbf{R}\mathbf{Z}$, then

$$\mathsf{Cov}(\tilde{\mathbf{Z}}, \mathbf{Z}) = \mathsf{Cov}(\overline{\mathbf{Z}}, \mathbf{Z}) - \mathbf{R}\mathsf{Cov}(\mathbf{Z}, \mathbf{Z}) = \mathbf{C}^{\mathsf{H}} - \mathbf{C}^{\mathsf{H}}\boldsymbol{\Gamma}^{-1}\boldsymbol{\Gamma} = \mathbf{0} \tag{3.18}$$

$$\mathsf{Cov}(\tilde{\mathbf{Z}}, \overline{\mathbf{Z}}) = \mathsf{Cov}(\overline{\mathbf{Z}}, \overline{\mathbf{Z}}) - \mathbf{R}\mathsf{Cov}(\mathbf{Z}, \overline{\mathbf{Z}}) = \boldsymbol{\Gamma}^{\mathsf{H}} - \mathbf{C}^{\mathsf{H}}\boldsymbol{\Gamma}^{-1}\mathbf{C} = \mathbf{P} \tag{3.19}$$

(iv) The two equivalent inverses of covariance matrix are

$$\begin{bmatrix} \boldsymbol{\Gamma} & \mathbf{C} \\ \overline{\mathbf{C}} & \overline{\boldsymbol{\Gamma}} \end{bmatrix}^{-1} = \begin{bmatrix} \overline{\mathbf{P}}^{-1} & -\mathbf{R}^{\mathsf{H}}\mathbf{P}^{-1} \\ -\mathbf{P}^{-1}\mathbf{R} & \mathbf{P}^{-1} \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{P}}^{-1} & -\mathbf{R}^{\mathsf{H}}\mathbf{P}^{-1} \\ -\mathbf{R}^{\mathsf{T}}\overline{\mathbf{P}}^{-1} & \mathbf{P}^{-1} \end{bmatrix}. \tag{3.20}$$

Also, by Woodbury matrix identity

$$\overline{\mathbf{P}}^{-1} = \boldsymbol{\Gamma}^{-1} + \boldsymbol{\Gamma}^{-1}\mathbf{C}\mathbf{P}^{-1}\mathbf{C}^{\mathsf{H}}\boldsymbol{\Gamma}^{-1} \tag{3.21}$$

### Conditional Distribution

**Theorem 2.** *Assume the multivariate complex Gaussian distribution*

$$\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_a \\ \mathbf{Z}_b \end{bmatrix} \sim \mathcal{CN}\left( \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Gamma}_a & \boldsymbol{\Gamma}_{ab} \\ \boldsymbol{\Gamma}_{ab}^{\mathsf{H}} & \boldsymbol{\Gamma}_b \end{bmatrix}, \begin{bmatrix} \mathbf{C}_a & \mathbf{C}_{ab} \\ \mathbf{C}_{ab}^{\mathsf{T}} & \mathbf{C}_b \end{bmatrix} \right)$$

*Then the conditional complex Gaussian distribution of* $\mathbf{Z}_a$ *given* $\mathbf{Z}_b = \mathbf{z}_b$ *is*

$$\mathcal{CN}\left(\boldsymbol{\mu}_{\mathbf{Z}_a|\mathbf{Z}_b=\mathbf{z}_b}, \boldsymbol{\Gamma}_{\mathbf{Z}_a|\mathbf{Z}_b=\mathbf{z}_b}, \mathbf{C}_{\mathbf{Z}_a|\mathbf{Z}_b=\mathbf{z}_b}\right) \tag{3.22}$$

*where*

$$\boldsymbol{\mu}_{\mathbf{Z}_a|\mathbf{Z}_b=\mathbf{z}_b} = \boldsymbol{\mu}_{\mathbf{Z}_a} + \mathbf{A}_{ab}(\mathbf{z}_b - \boldsymbol{\mu}_b) + \mathbf{B}_{ab}(\overline{\mathbf{z}}_b - \overline{\boldsymbol{\mu}}_b), \tag{3.23}$$

$$\boldsymbol{\Gamma}_{\mathbf{Z}_a|\mathbf{Z}_b=\mathbf{z}_b} = \boldsymbol{\Gamma}_a - \mathbf{A}_{ab}\boldsymbol{\Gamma}_{ab}^{\mathsf{H}} - \mathbf{B}_{ab}\mathbf{C}_{ab}^{\mathsf{H}}, \tag{3.24}$$

$$\mathbf{C}_{\mathbf{Z}_a|\mathbf{Z}_b=\mathbf{z}_b} = \mathbf{C}_a - \mathbf{A}_{ab}\mathbf{C}_{ab}^{\mathsf{T}} - \mathbf{B}_{ab}\boldsymbol{\Gamma}_{ab}^{\mathsf{T}} \tag{3.25}$$

*and*

$$\begin{aligned} \mathbf{P}_b &\triangleq \overline{\boldsymbol{\Gamma}}_b - \mathbf{C}_b^{\mathsf{H}}\boldsymbol{\Gamma}_b^{-1}\mathbf{C}_b, \\ \mathbf{A}_{ab} &\triangleq \left(\boldsymbol{\Gamma}_{ab} - \mathbf{C}_{ab}\overline{\boldsymbol{\Gamma}}_b^{-1}\overline{\mathbf{C}}_b\right)\overline{\mathbf{P}}_b^{-1}, \\ \mathbf{B}_{ab} &\triangleq \left(\mathbf{C}_{ab} - \boldsymbol{\Gamma}_{ab}\boldsymbol{\Gamma}_b^{-1}\mathbf{C}_b\right)\mathbf{P}_b^{-1}. \end{aligned}$$

## 3.2.3 Joint Moment Generating Function

**Definition 3** (Joint Moment Generating Function)**.** *Let* $\mathbf{X}$ *be a* $K$*-dimensional random vector. If there exists* $\{h_i\}_{i=1}^K$ *such that* $h_i > 0$ *and the expected value*

$$\mathbb{E}\left[\exp\left(\mathbf{t}^{\mathsf{T}}\mathbf{X}\right)\right] = \mathbb{E}\left[\exp\left(t_1X_1 + \cdots + t_KX_K\right)\right] \tag{3.26}$$

*exists and is finite for all* $\mathbf{t} \in \mathcal{H} \triangleq \otimes_{i=1}^K [-h_i, h_i] \subseteq \mathbb{R}^K$*, then we say that* $\mathbf{X}$ *possesses a joint moment generating function and the function* $M_X : \mathcal{H} \to \mathbb{R}$ *defined by*

$$M_{\mathbf{X}}(\mathbf{t}) = \mathbb{E}\left[\exp\left(\mathbf{t}^{\mathsf{T}}\mathbf{X}\right)\right] \tag{3.27}$$

*is called the joint moment generating function of* $\mathbf{X}$*.*

The joint moment generating function is useful to compute the cross-moments of a multivariate distribution. If a $K$-dimensional random vector $\mathbf{X}$ possesses a joint moment generating function $M_{\mathbf{X}}(\mathbf{t})$, then define the cross-moment of order $n$ as

$$\mu_{\mathbf{X}}(n_1, \ldots, n_K) = \mathbb{E}\left[X_1^{n_1} \cdot X_2^{n_2} \cdots X_K^{n_K}\right], \tag{3.28}$$

where $n_1, \ldots, n_K \in \mathbb{N}$ and $n = \sum_{i=1}^{K} n_i$. Then, this gives

$$\mu_{\mathbf{X}}(n_1, \ldots, n_K) = \left. \frac{\partial^{n_1+n_2+\cdots+n_K} M_{\mathbf{X}}(t_1, t_2, \ldots, t_K)}{\partial t_1^{n_1} \partial t_2^{n_2} \cdots \partial t_K^{n_K}} \right|_{t_1=0, t_2=0, \ldots, t_K=0} \tag{3.29}$$

For example, consider $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ with $\Sigma_{ij} = \rho_{ij}\sigma_i\sigma_j$ and $\rho_{ii} = 1$, $\sigma_i > 0$ for $i = 1, \ldots, n$, and $\rho_{ij} \in [-1, 1]$ for $i \neq j$.

The corresponding joint MGF is

$$M_{\mathbf{X}}(\mathbf{t}) = \exp\left(\mathbf{t}^\top \boldsymbol{\mu} + \frac{1}{2}\mathbf{t}^\top \mathbf{\Sigma} \mathbf{t}\right) = \exp\left(\frac{1}{2}\mathbf{t}^\top \mathbf{\Sigma} \mathbf{t}\right)$$

For distinct $i, j, k, l$, the following cross-moments are computed by (3.29) for later use

$$\begin{aligned}
\mathbb{E}[X_i] = \mathbb{E}[X_i X_j^2] = \mathbb{E}[X_i X_j X_k] = 0, \quad \mathbb{E}[X_i^2] = \sigma_i^2, \quad \mathbb{E}[X_i X_j] = \rho_{ij}\sigma_i\sigma_j \\
\mathbb{E}[X_i^2 X_j^2] = (1 + 2\rho_{ij}^2)\sigma_i^2\sigma_j^2, \quad \mathbb{E}[X_i X_j X_k^2] = (\rho_{ij} + 2\rho_{ik}\rho_{jk})\sigma_i\sigma_j\sigma_k^2 \\
\mathbb{E}[X_i X_j X_k X_l] = (\rho_{ij}\rho_{kl} + \rho_{ik}\rho_{jl} + \rho_{il}\rho_{jk})\,\sigma_i\sigma_j\sigma_k\sigma_l
\end{aligned} \tag{3.30}$$

### Properties

(i) Let $\mathbf{X}$ be a $K$-dimensional random vector possessing joint moment generating function $M_{\mathbf{X}}(\mathbf{t})$. Define $\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{b}$ where $\mathbf{A} \in \mathbb{R}^{L \times K}$, $\mathbf{b} \in \mathbb{R}^{L \times 1}$ are constant, then the joint moment generating function of $\mathbf{Y}$ is given by

$$M_{\mathbf{Y}}(\mathbf{t}) = \exp\left(\mathbf{t}^\top \mathbf{b}\right) M_{\mathbf{X}}\left(\mathbf{A}^\top \mathbf{t}\right). \tag{3.31}$$

(ii) Let $\mathbf{X}$ be a $K$-dimensional random vector with mutually independent entries $X_i$'s, each one has MGF $M_{X_i}(t_i)$, then the MGF of $\mathbf{X}$ is

$$M_{\mathbf{X}}(t_1, \ldots, t_K) = \prod_{i=1}^{K} M_{X_i}(t_i) \tag{3.32}$$

(iii) Let $\mathbf{X}$ be a $K$-dimensional random vector with mutually independent entries $X_i$'s, each one has MGF $M_{X_i}(t_i)$, then the MGF of their sum $\mathbf{Z} = \sum_{i=1}^{K} X_i$ is

$$M_{\mathbf{Z}}(t) = \prod_{i=1}^{K} M_{X_i}(t) \tag{3.33}$$

### 3.2.4 Non-linear Schrödinger Equation

The propagation of light in optical fibers is governed by the nonlinear Schrödinger equation (NLSE) [Agr06]. Ignoring the polarization effects, and considering chromatic dispersion, nonlinear Kerr effect, and energy loss, in a time frame moving with the signal group velocity, the NLSE for a fiber span can be written as

$$\frac{\partial v(z,t)}{\partial z} = -\mathbb{i}\frac{\beta_2}{2}\frac{\partial^2 v(z,t)}{\partial t^2} + \mathbb{i}\gamma\left|v(z,t)\right|^2 v(z,t) - \frac{\alpha}{2}v(z,t) \tag{3.34}$$

where

- $v(z,t)$ is the optical field complex envelope, where $z$ is location and $t$ is time,
- $\alpha$ is the attenuation coefficient describing the exponential attenuation of the signal power,
- $\beta_2$ is the chromatic dispersion coefficient accounting for group velocity dispersion that causes pulse broadening during propagation, and
- $\gamma$ is the nonlinear Kerr effect coefficient capturing the nonlinear refraction.

In the following context, we use $\mathsf{NLSE}(v(z,t), L; \alpha, \beta_2, \gamma)$ to denote the signal $v(z+L,t)$ obtained by solving the NLSE with initial condition $v(z,t)$ and parameters $\alpha, \beta_2, \gamma$.

To cancel of the last term, we use the change of variables $u(z,t) \triangleq \mathbb{e}^{\alpha z/2} v(z,t)$ to write

$$\frac{\partial u(z,t)}{\partial z} = -\mathbb{i}\frac{\beta_2}{2}\frac{\partial^2 u(z,t)}{\partial t^2} + \mathbb{i}\gamma\mathbb{e}^{-\alpha z}\left|u(z,t)\right|^2 u(z,t), \tag{3.35}$$

To solve NLSE analytically, let $f(z,t) \triangleq \mathbb{e}^{-\alpha z}\left|u(z,t)\right|^2 u(z,t)$ and take the Fourier transform on both side

$$\frac{\partial U(z,\omega)}{\partial z} = \mathbb{i}\frac{\beta_2}{2}\omega^2 U(z,\omega) + \mathbb{i}\gamma F(z,\omega) \tag{3.36}$$

By letting $U(z,\omega) \triangleq \mathrm{e}^{\mathrm{i}\beta_2\omega^2 z/2}Y(z,\omega)$, we have

$$\frac{\partial Y(z,\omega)}{\partial z} = \mathrm{i}\gamma \mathrm{e}^{-\mathrm{i}\beta_2\omega^2 z/2}F(z,\omega) \tag{3.37}$$

Using (3.37) and the relation between $U(z,w)$ and $Y(z,w)$, by integrating from 0 to $z$, and let $H(z,\omega) \triangleq \exp(\mathrm{i}\beta_2\omega^2 z/2)$, it yields that

$$\begin{aligned} Y(z,\omega) &= Y(0,\omega) + \int_0^z \left.\frac{\partial Y(z,\omega)}{\partial z}\right|_{z=\zeta} \mathrm{d}\zeta \\ &= Y(0,\omega) + \mathrm{i}\gamma \int_0^z \mathrm{e}^{-\mathrm{i}\beta_2\omega^2\zeta/2}F(\zeta,\omega)\,\mathrm{d}\zeta \qquad (3.38) \\ U(z,\omega) &= \mathrm{e}^{\mathrm{i}\beta_2\omega^2 z/2}Y(0,\omega) + \mathrm{i}\gamma \int_0^z \mathrm{e}^{\mathrm{i}\beta_2\omega^2(z-\zeta)/2}F(\zeta,\omega)\,\mathrm{d}\zeta \\ &= H(z,\omega)U(0,\omega) + \mathrm{i}\gamma H(z,\omega) * F(,\omega) \\ &= U_0(z,\omega) + \mathrm{i}\gamma H(z,\omega) * F(z,\omega) \qquad (3.39) \end{aligned}$$

where $U_0(z,\omega) = \mathrm{e}^{\mathrm{i}\beta_2\omega^2 z/2}U(0,\omega)$ is the Fourier transform of solution of the NLSE when $\gamma = 0$. Finally, using the inverse Fourier transform and the definition of $f(z,t)$, we have

$$u(z,t) = u_0(z,t) + \mathrm{i}\gamma h(z,t)\mathrm{e}^{-\alpha z}\left|u(z,t)\right|^2 u(z,t), \tag{3.40}$$

where $u_0(z,t) = u(0,t) * h(z,t)$ is the signal at $z$ in a linear and lossless fiber.

### 3.2.5 Split-Step Fourier Method

Due to its simplicity and computational efficiency, the SSFM [SF08] is probably the most known and commonly used numerical method for solving the NLSE in fiber-optic communications.

The main idea of SSFM is that, for short propagation distance, dispersion and nonlinearity, in general acting together along the fiber, can instead be separated as if they act independently from each other.

#### Dispersion-dominant Regime

When the nonlinear term of the NLSE is negligible, (3.36) reduces to

$$\frac{\partial U(z,\omega)}{\partial z} = \mathrm{i}\frac{\beta_2}{2}\omega^2 U(z,\omega) \tag{3.41}$$

$$U(z,\omega) = \exp\left(\mathbb{i}\frac{\beta_2}{2}\omega^2 z\right) U(0,\omega) = H(z,\omega)U(0,\omega) \tag{3.42}$$

Taking $u(z,t) = \mathbb{e}^{\alpha z/2} v(z,t)$ into account, the term $\exp(-\alpha z/2)$ can be regarded as a constant when taking Fourier transform w.r.t. $t$, and take inverse Fourier transform on both sides

$$V(z,\omega) = \mathbb{e}^{-\alpha z/2} H(z,\omega)V(0,\omega), \qquad v(z,t) = \mathbb{e}^{-\alpha z/2} v(0,t) * h(z,t) \tag{3.43}$$

### Nonlinear-dominant Regime

When the linear term of the NLSE is negligible, then the (3.35) reduces to

$$\frac{\partial u(z,t)}{\partial z} = \mathbb{i}\gamma \exp(-\alpha z) \left|u(z,t)\right|^2 u(z,t) \tag{3.44}$$

The solution for this case is

$$u(z,t) = u(0,t) \exp\left(\mathbb{i}\gamma \left|u(0,t)\right|^2 \frac{1-\mathbb{e}^{-\alpha z}}{\alpha}\right) \tag{3.45}$$

Taking $u(z,t) = \mathbb{e}^{\alpha z/2} v(z,t)$ into account, we have $v(0,t) = u(0,t)$ and thus

$$v(z,t) = v(0,t) \exp\left(-\frac{\alpha z}{2} + \mathbb{i}\gamma \left|v(0,t)\right|^2 \frac{1-\mathbb{e}^{-\alpha z}}{\alpha}\right) \tag{3.46}$$

### Solving NLSE Numerically by SSFM

The simplest SSFM algorithm first cuts the fiber length into small segment with distance $\Delta z$, and for each segment apply (3.43), (3.46) alternatively, i.e.

$$\begin{aligned}
\tilde{v}(z+\Delta z,t) &= \mathbb{e}^{-\alpha z/2} v(z,t) * h(\Delta z,t), \\
v(z+\Delta z,t) &= \tilde{v}(z+\Delta z,t) \exp\left(\mathbb{i}\gamma \left|\tilde{v}(z+\Delta z,t)\right|^2 \Delta z\right).
\end{aligned}$$

The forward and inverse Fourier transform are frequently applied since equation (3.43) is defined in the frequency domain.

### 3.2.6 Fiber-Optic Systems

For fiber-optic systems, the transmitter is first modulated by the signal $s(t)$ into the channel input $v(0,t)$ (e.g./ via a linear pulse shaping operation). Then, the channel input $v(0,t)$ is propagated along the optical fiber link over a long distance.

To overcome the fiber loss and reach the desired distance, the signal has to be amplified periodically. Basically, to model the optical signal over distance $L$, we first cut the distance into $N_{\text{sp}}$ equidistant spans with $L_{\text{sp}} = L/N_{\text{sp}}$, and place an amplifier at the end of each span with gain $G = \mathrm{e}^{\alpha L_{\text{sp}}}$.

Therefore, for $k = 1, \ldots, N_{\text{sp}}$, we have

$$v((k-1)L_{\text{sp}} + \zeta, t) = \mathsf{NLSE}\left(v\left((k-1)L_{\text{sp}}, t\right), \zeta; \alpha, \beta_2, \gamma\right) \tag{3.47}$$

$$v(kL_{\text{sp}}, t) = \sqrt{G}\, v(kL_{\text{sp}} - \varepsilon, t) + n(t), \tag{3.48}$$

where $\varepsilon \to 0^+$, $\zeta \in [0, L_{\text{sp}}]$ and $n(t)$ is the additive Gaussian noise introduced by the amplifier.

Finally, general task of the receiver is to recover $s(t)$ given received signal $y(t) = v(L,t)$.

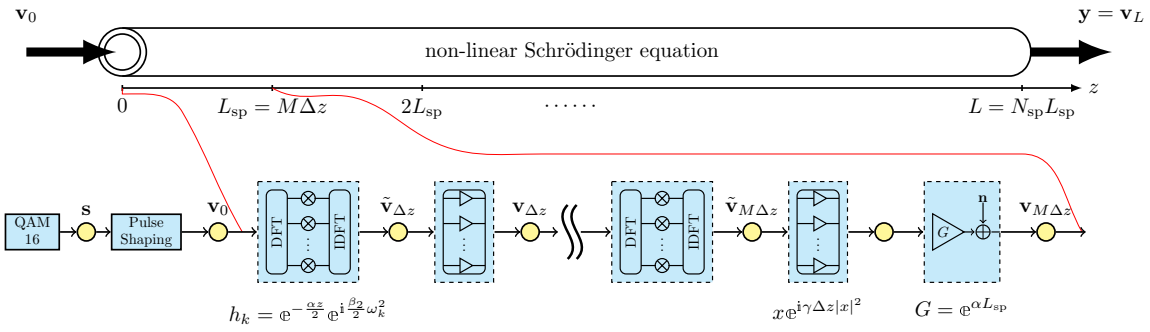



**Figure 3.1**: SSFM for one span of the optical system.

When using the SSFM, each span is first split into $M$ segments of length $\Delta z = L_{\text{sp}}/M$. Denote $\mathbf{v}_z$ as the discrete sequence representing $v(z,t)$, and $\mathbf{h}$ as the discrete sequence

representing $h(\Delta z, t)$. Then for $n = 0, \ldots, N_{\text{sp}-1}$, the SSFM on discrete sequence becomes

$$\mathbf{v}_0 = \mathsf{PulseShaping}(\mathbf{s}) \tag{3.49}$$

$$\tilde{\mathbf{v}}_{(nM+m+1)\Delta z} = \mathsf{IDFT}\left(\mathbf{h} \odot \mathsf{DFT}\left(\mathbf{v}_{(nM+m)\Delta z}\right)\right), \quad \forall\, m = 0, \ldots, M-1 \tag{3.50}$$

$$\mathbf{v}_{(nM+m+1)\Delta z} = \begin{cases} \kappa_{\Delta z}\left(\tilde{\mathbf{v}}_{(nM+m+1)\Delta z}\right), & \text{if } m = 0, \ldots, M-2 \\ \sqrt{G}\,\kappa_{\Delta z}\left(\tilde{\mathbf{v}}_{(nM+m+1)\Delta z}\right) + \mathbf{n}, & \text{if } m = M-1 \end{cases} \tag{3.51}$$

$$\mathbf{y} = \mathbf{v}_L \tag{3.52}$$

where $\odot$ denote element-wise multiplication of two vectors, and the function $\kappa_{\Delta z}(x) = x \exp(\mathbb{i}\gamma\, |x|^2\, \Delta z)$ is applied element-wise to vectors.

### 3.2.7 Previous Works

#### Digital Back Propagation (DBP)

The DBP algorithm [IK08] simply ignores the additive noise introduced in the amplifier and back solves the NLSE directly, write $\mathbf{h}^{-1}$ as element-wise inverse of the vector $\mathbf{h}$ and note that $\kappa_{\Delta z}^{-1}(x) = \kappa_{-\Delta z}(x)$. Thus, for $n = 0, \ldots, N_{\text{sp}} - 1$ we have,

$$\hat{\mathbf{v}}_L = \mathbf{y} \tag{3.53}$$

$$\hat{\tilde{\mathbf{v}}}_{(nM+m+1)\Delta z} = \begin{cases} \kappa_{-\Delta z}\left(\hat{\mathbf{v}}_{(nM+m+1)\Delta z}\right), & \text{if } m = 0, 1, \ldots, M-2 \\ \frac{1}{\sqrt{G}}\kappa_{-\Delta z}\left(\hat{\mathbf{v}}_{(nM+m+1)\Delta z}\right), & \text{if } m = M-1 \end{cases} \tag{3.54}$$

$$\hat{\mathbf{v}}_{(nM+m)\Delta z} = \mathsf{IDFT}\left(\mathbf{h}^{-1} \odot \mathsf{DFT}\left(\hat{\tilde{\mathbf{v}}}_{(nM+m+1)\Delta z}\right)\right), \ \forall\, m = 0, \ldots, M-1 \tag{3.55}$$

$$\hat{\mathbf{s}} = \mathsf{MatchedFilter}(\hat{\mathbf{v}}_0) \tag{3.56}$$

Once $\hat{\mathbf{s}}$ is obtained, signal detection is typically done by choosing the nearest constellation points. The DBP uses nearest neighbor for the signal detection, which is equivalent to assuming that the posterior contour is circular for each symbol.

#### Stochastic Digital Back Propagation (SDBP)

Stochastic Digital Back Propagation (SDBP) [IWJA14] uses a particle filter approach to approximate the posterior $p(\mathbf{s} \,|\, \mathbf{y})$. Let $N_{\text{p}}$ be the number of particles, $\hat{\mathbf{v}}_z^{(k)}$ to be the $k$-th

particle of estimation $\hat{\mathbf{v}}_z^{(k)}$, then for $k = 1, \ldots, N_\mathrm{p}$ the SDBP algorithm gives

$$\hat{\mathbf{v}}_L^{(k)} = \mathbf{y}^{(k)} \tag{3.57}$$

$$\hat{\hat{\mathbf{v}}}_{(nM+m+1)\Delta z}^{(k)} = \begin{cases} \kappa_{-\Delta z}\left(\hat{\mathbf{v}}_{(nM+m+1)\Delta z}^{(k)}\right), & \text{if } m = 0, 1, \ldots, M-2 \\ \sim \mathcal{N}\left(\frac{\kappa_{-\Delta z}\left(\hat{\mathbf{v}}_{(nM+m+1)\Delta z}^{(k)}\right)}{\sqrt{G}}, \frac{\mathbf{\Sigma}}{G}\right), & \text{if } m = M-1 \end{cases} \tag{3.58}$$

$$\hat{\mathbf{v}}_{(nM+m)\Delta z}^{(k)} = \mathsf{IDFT}\left(\mathbf{h}^{-1} \odot \mathsf{DFT}\left(\hat{\hat{\mathbf{v}}}_{(nM+m+1)\Delta z}^{(k)}\right)\right), \ \forall\, m = 0, \ldots, M-1 \tag{3.59}$$

$$\hat{\mathbf{s}}^{(k)} = \mathsf{MatchedFilter}(\hat{\mathbf{v}}_0^{(k)}) \tag{3.60}$$

In contrast to DBP, SDBP returns $N_\mathrm{p}$ particles of $\hat{\mathbf{s}}$, which allow the estimation of the posterior $p(\mathbf{s} \,|\, \mathbf{y})$. For longer link distances and larger nonlinear effects, this has a lower symbol error rate because the symbol-wise posterior is not circular. However the computation complexity of SDBP is almost $N_p$ times of DBP.

## 3.3 Covariance Propagation: Gaussian Approximation of SDBP

As shown in [IWJA14], the contour of the symbol-wise posterior $p(s_i \,|\, \mathbf{y})$ is usually elliptical but when the nonlinear effect is strong it becomes kidney-shaped. Nonetheless, a multivariate Gaussian often gives a good approximation. The basic idea is that, for each step in SDBP, one can replace the particle filter by a multivariate complex Gaussian distribution.

### 3.3.1 Uparrow Notation Parameterization

Consider $\mathbf{x} = \boldsymbol{\mu} + \mathbf{w}$, where $\boldsymbol{\mu}$ is constant and $\mathbf{w}$ is complex Gaussian distributed with

$$\mathbf{w}^{\uparrow} = \begin{bmatrix} w_{1,\mathrm{r}} \\ w_{1,\mathrm{i}} \\ \vdots \\ w_{n,\mathrm{r}} \\ w_{n,\mathrm{i}} \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}), \quad \mathbf{\Sigma} = \begin{bmatrix} \sigma_{1,\mathrm{r}}^2 & \rho_{11}^{\mathrm{ri}}\sigma_{1,\mathrm{r}}\sigma_{1,\mathrm{i}} & \cdots & \rho_{1n}^{\mathrm{rr}}\sigma_{1,\mathrm{r}}\sigma_{n,\mathrm{r}} & \rho_{1n}^{\mathrm{ri}}\sigma_{1,\mathrm{r}}\sigma_{n,\mathrm{i}} \\ \rho_{11}^{\mathrm{ir}}\sigma_{1,\mathrm{r}}\sigma_{1,\mathrm{i}} & \sigma_{1,\mathrm{i}}^2 & & \rho_{1n}^{\mathrm{ir}}\sigma_{1,\mathrm{i}}\sigma_{n,\mathrm{r}} & \rho_{1n}^{\mathrm{ii}}\sigma_{1,\mathrm{i}}\sigma_{n,\mathrm{i}} \\ \vdots & & \ddots & \vdots & \\ \rho_{n1}^{\mathrm{rr}}\sigma_{n,\mathrm{r}}\sigma_{1,\mathrm{r}} & \rho_{n1}^{\mathrm{ri}}\sigma_{n,\mathrm{r}}\sigma_{1,\mathrm{i}} & \cdots & \sigma_{n,\mathrm{r}}^2 & \rho_{nn}^{\mathrm{ri}}\sigma_{n,\mathrm{r}}\sigma_{n,\mathrm{i}} \\ \rho_{n1}^{\mathrm{ir}}\sigma_{n,\mathrm{i}}\sigma_{1,\mathrm{r}} & \rho_{n1}^{\mathrm{ii}}\sigma_{n,\mathrm{i}}\sigma_{1,\mathrm{i}} & & \rho_{nn}^{\mathrm{ir}}\sigma_{n,\mathrm{r}}\sigma_{n,\mathrm{i}} & \sigma_{n,\mathrm{i}}^2 \end{bmatrix}, \tag{3.61}$$

where $\rho^{\mathrm{rr}}_{jj'}$ is the correlation coefficient between $w_{j,\mathrm{r}}$ and $w_{j',\mathrm{r}}$, and similarly for $\rho^{\mathrm{ri}}_{jj'}, \rho^{\mathrm{ir}}_{jj'}, \rho^{\mathrm{ii}}_{jj'}$. Since $\mathbf{\Sigma}$ is symmetric, it also follows that

$$\rho^{\mathrm{rr}}_{jj'} = \rho^{\mathrm{rr}}_{j'j}, \quad \rho^{\mathrm{ii}}_{jj'} = \rho^{\mathrm{ii}}_{j'j}, \quad \rho^{\mathrm{ir}}_{jj'} = \rho^{\mathrm{ri}}_{j'j}. \tag{3.62}$$

The general question is that, given a continuous function $g : \mathbb{C}^n \to \mathbb{C}^n$, how can one find $\tilde{\mathbf{\Sigma}}$ such that

$$\tilde{\mathbf{x}} \triangleq g(\mathbf{x}) = g(\boldsymbol{\mu} + \mathbf{w}) \approx g(\boldsymbol{\mu}) + \tilde{\mathbf{w}}, \text{ where } \tilde{\mathbf{w}}^{\uparrow} \sim \mathcal{N}\left(\mathbf{0}, \tilde{\mathbf{\Sigma}}\right). \tag{3.63}$$

In the following context, we denote $\Sigma_{jj'}$, $\tilde{\Sigma}_{jj'}$ as the $(j, j')$-th $2 \times 2$ block of $\mathbf{\Sigma}$, $\tilde{\mathbf{\Sigma}}$, respectively, which captures the complex correlations between $j$-th and $j'$-th components.

**Theorem 4.** *Assume that the input multivariate complex Gaussian distribution of each step is represented by a dimension-2n Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$ with*

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_{1,\mathrm{r}} \\ \mu_{1,\mathrm{i}} \\ \vdots \\ \mu_{n,\mathrm{r}} \\ \mu_{n,\mathrm{ir}} \end{bmatrix}, \quad \mathbf{\Sigma} = \begin{bmatrix} \sigma^2_{1,\mathrm{r}} & \rho^{\mathrm{ri}}_{11}\sigma_{1,\mathrm{r}}\sigma_{1,\mathrm{i}} & \cdots & \rho^{\mathrm{rr}}_{1n}\sigma_{1,\mathrm{r}}\sigma_{n,\mathrm{r}} & \rho^{\mathrm{ri}}_{1n}\sigma_{1,\mathrm{r}}\sigma_{n,\mathrm{i}} \\ \rho^{\mathrm{ir}}_{11}\sigma_{1,\mathrm{r}}\sigma_{1,\mathrm{i}} & \sigma^2_{1,\mathrm{i}} & & \rho^{\mathrm{ir}}_{1n}\sigma_{1,\mathrm{i}}\sigma_{n,\mathrm{r}} & \rho^{\mathrm{ii}}_{1n}\sigma_{1,\mathrm{i}}\sigma_{n,\mathrm{i}} \\ \vdots & & \ddots & \vdots & \\ \rho^{\mathrm{rr}}_{n1}\sigma_{n,\mathrm{r}}\sigma_{1,\mathrm{r}} & \rho^{\mathrm{ri}}_{n1}\sigma_{n,\mathrm{r}}\sigma_{1,\mathrm{i}} & \cdots & \sigma^2_{n,\mathrm{r}} & \rho^{\mathrm{ri}}_{nn}\sigma_{n,\mathrm{r}}\sigma_{n,\mathrm{i}} \\ \rho^{\mathrm{ir}}_{n1}\sigma_{n,\mathrm{i}}\sigma_{1,\mathrm{r}} & \rho^{\mathrm{ii}}_{n1}\sigma_{n,\mathrm{i}}\sigma_{1,\mathrm{i}} & & \rho^{\mathrm{ir}}_{nn}\sigma_{n,\mathrm{r}}\sigma_{n,\mathrm{i}} & \sigma^2_{n,\mathrm{i}} \end{bmatrix}.$$

*The covariance back-propagation in the uparrow notation parameterization is described as following:*

- *Linear step:*

$$\tilde{\mathbf{\Sigma}} = \left[\mathbf{F}^{\mathsf{H}}\right]^{\Uparrow} \left[\mathsf{diag}(\mathbf{h}^{-1})\right]^{\Uparrow} \mathbf{F}^{\Uparrow} \mathbf{\Sigma} \left[\mathbf{F}^{\mathsf{H}}\right]^{\Uparrow} \left[\mathsf{diag}(\mathbf{h}^{-1})\right]^{\Uparrow} \mathbf{F}^{\Uparrow}. \tag{3.64}$$

- *Nonlinear step: Let $\gamma^* = \gamma \Delta z$ to be the equivalent Kerr effect coefficient for one SSFT segment. Then*

$$\tilde{\mathbf{\Sigma}} = \gamma_*^2 \mathbf{Q}_{\gamma_*}(\boldsymbol{\mu})\, \mathbf{\Sigma}\, \left[\mathbf{Q}_{\gamma_*}(\boldsymbol{\mu})\right]^{\mathsf{T}}, \tag{3.65}$$

*where*

$$\mathbf{Q}_{\gamma_*}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{T}_{\gamma_*}(\mu_1) & & \\ & \ddots & \\ & & \mathbf{T}_{\gamma_*}(\mu_n) \end{bmatrix} \tag{3.66}$$

*and*

$$\mathbf{T}_{\gamma_*}(\mu_j) = \begin{bmatrix} \cos(\gamma_* |\mu_j|^2) & \sin(\gamma_* |\mu_j|^2) \\ -\sin(\gamma_* |\mu_j|^2) & \cos(\gamma_* |\mu_j|^2) \end{bmatrix} \begin{bmatrix} 1/\gamma_* + 2\mu_{j,\mathrm{r}}\mu_{j,\mathrm{i}} & 2\mu_{j,\mathrm{i}}^2 \\ -2\mu_{j,\mathrm{r}}^2 & 1/\gamma_* - 2\mu_{j,\mathrm{r}}\mu_{j,\mathrm{i}} \end{bmatrix}. \tag{3.67}$$

- *Amplification step, let* $G = \exp(\alpha L_{\mathrm{sp}})$ *be the amplitude decay of over span, and denote* $\sigma_{\mathrm{n}}^2$ *as the variance of additive circularly-symmetric complex Gaussian noise, then*

$$\tilde{\mathbf{\Sigma}} = \frac{\mathbf{\Sigma} + \frac{\sigma_{\mathrm{n}}^2}{2}\mathbf{I}}{G}. \tag{3.68}$$

### 3.3.2 Complex Gaussian Parameterization

In last section, we have seen how the covariance matrix evolves for each step of SSFM using the uparrow notation. Using the properties of complex Gaussian distributions, we can now transform the uparrow notation back to complex Gaussian distributions.

Consider $\mathbf{x} = \boldsymbol{\mu} + \mathbf{w}$, where $\boldsymbol{\mu}$ is constant with $\mathbf{w} \sim \mathcal{CN}(\mathbf{0}, \mathbf{G}, \mathbf{C})$. The general question is, given a continuous function $g : \mathbb{C}^n \to \mathbb{C}^n$, what are the $\tilde{\mathbf{\Gamma}}, \tilde{\mathbf{C}}$ such that

$$\tilde{\mathbf{x}} \triangleq g(\mathbf{x}) = g(\boldsymbol{\mu} + \mathbf{w}) \approx g(\boldsymbol{\mu}) + \tilde{\mathbf{w}}, \text{ where } \tilde{\mathbf{w}} \sim \mathcal{CN}\left(\mathbf{0}, \tilde{\mathbf{\Gamma}}, \tilde{\mathbf{C}}\right). \tag{3.69}$$

**Theorem 5.** *Assume that the input multivariate complex Gaussian distribution of each step has the distribution* $\mathcal{CN}(\boldsymbol{\mu}, \mathbf{\Gamma}, \mathbf{C})$*. The covariance back-propagation in the complex Gaussian parameterization is described as following:*

- *Linear step:*

$$\tilde{\mathbf{\Gamma}} = \mathbf{F}^{\mathsf{H}}\mathsf{diag}(\mathbf{h}^{-1})\mathbf{F}\mathbf{\Gamma}\mathbf{F}^{\mathsf{H}}\mathsf{diag}(\overline{\mathbf{h}}^{-1})\mathbf{F} \tag{3.70}$$

$$\tilde{\mathbf{C}} = \mathbf{F}^{\mathsf{H}}\mathsf{diag}(\mathbf{h}^{-1})\mathbf{F}\mathbf{C}\mathbf{F}\mathsf{diag}(\mathbf{h}^{-1})\mathbf{F}^{\mathsf{H}} \tag{3.71}$$

- *Nonlinear step, let* $\gamma^* = \gamma\Delta z$ *to be the equivalent Kerr effect coefficient for one SSFT segment, then*

$$\begin{bmatrix} \tilde{\mathbf{\Gamma}} & \tilde{\mathbf{C}} \\ \overline{\tilde{\mathbf{C}}} & \overline{\tilde{\mathbf{\Gamma}}} \end{bmatrix} = \frac{\gamma_*^2}{4}\mathbf{\Omega}\mathbf{\Pi}\mathbf{Q}_{\gamma_*}(\boldsymbol{\mu})\mathbf{\Pi}^{\mathsf{T}}\mathbf{\Omega}^{\mathsf{H}} \begin{bmatrix} \mathbf{\Gamma} & \mathbf{C} \\ \overline{\mathbf{C}} & \overline{\mathbf{\Gamma}} \end{bmatrix} \mathbf{\Omega}\mathbf{\Pi}\left[\mathbf{Q}_{\gamma_*}(\boldsymbol{\mu})\right]^{\mathsf{T}}\mathbf{\Pi}^{\mathsf{T}}\mathbf{\Omega}^{\mathsf{H}} \tag{3.72}$$

*where matrices* $\mathbf{\Pi}$*,* $\mathbf{\Omega}$ *are defined in* (3.4), (3.10)

- *Amplification step, let* $G = \exp(\alpha L_{\mathrm{sp}})$ *be the amplitude decay of over span, and denote* $\sigma_{\mathrm{n}}^2$ *as the variance of additive circularly-symmetric complex Gaussian noise, then*

$$\tilde{\boldsymbol{\Gamma}} = \frac{\boldsymbol{\Sigma} + \frac{\sigma_{\mathrm{n}}^2}{2}\mathbf{I}}{G}, \qquad \tilde{\mathbf{C}} = \frac{\mathbf{C}}{G}. \tag{3.73}$$

### 3.3.3 Covariance Backpropagation (CBP)

**Algorithm 7** CBP Algorithm

Initialize $\boldsymbol{\mu}_L = \mathbf{y}$, $\boldsymbol{\Gamma}_L = \mathbf{C}_L = \mathbf{I}$.
**for** $n = N_{\mathrm{sp}} - 1$ **to** $0$ **do**
  `/* de-amplify and nonlinear step */`
  update $\tilde{\boldsymbol{\mu}}_{(nM+M)\Delta z}$ according to (3.54)
  update $\tilde{\boldsymbol{\Gamma}}_{(nM+M)\Delta z}$, $\tilde{\mathbf{C}}_{(nM+M)\Delta z}$ according to (3.73), (3.72)
  `/* linear step */`
  update $\boldsymbol{\mu}_{(nM+M-1)\Delta z}$ according to (3.55)
  update $\boldsymbol{\Gamma}_{(nM+M-1)\Delta z}$, $\mathbf{C}_{(nM+M-1)\Delta z}$ according to (3.70), (3.71)
  **for** $m = M - 2$ **to** $0$ **do**
    `/* nonlinear step */`
    update $\tilde{\boldsymbol{\mu}}_{(nM+m+1)\Delta z}$ according to (3.54)
    update $\tilde{\boldsymbol{\Gamma}}_{(nM+m+1)\Delta z}$, $\tilde{\mathbf{C}}_{(nM+m+1)\Delta z}$ according to (3.72)
    `/* linear step */`
    update $\boldsymbol{\mu}_{(nM+m)\Delta z}$ according to (3.55)
    update $\boldsymbol{\Gamma}_{(nM+m)\Delta z}$, $\mathbf{C}_{(nM+m)\Delta z}$ according to (3.70), (3.71)
  **end for**
**end for**
**return** $\boldsymbol{\mu}_0, \boldsymbol{\Gamma}_0, \mathbf{C}_0$

For the Gaussian approximation SDBP, the evolution of mean is exactly same as DBP, but the GA-SDBP in this section is formulated under complex Gaussian parameterization. The algorithm returns $\boldsymbol{\mu}_0, \boldsymbol{\Gamma}_0, \mathbf{C}_0$ such that $p(\mathbf{v}_0 \,|\, \mathbf{y})$ is approximated by $\mathcal{CN}(\boldsymbol{\mu}_0, \boldsymbol{\Gamma}_0, \mathbf{C}_0)$.

## 3.4 Signal Detection

Once the estimate $\hat{\mathbf{v}}_0$ is obtained, we can apply matched filter to estimate $\mathbf{s}$. Since the matched filter is a linear operation, we write

$$\mathbf{s}' = \mathsf{MatchedFilter}(\hat{\mathbf{v}}_0) = \mathbf{M}_{\mathrm{mf}}\hat{\mathbf{v}}_0.$$

From the property of multivariate complex Gaussian distribution, the posterior distribution of $\mathbf{s}$ can be estimated by

$$p(\mathbf{s}\,|\,\mathbf{y}) \approx \mathcal{CN}(\boldsymbol{\mu}_{\mathrm{s}}, \boldsymbol{\Gamma}_{\mathrm{s}}, \mathbf{C}_{\mathrm{s}}) = \mathcal{CN}\left(\mathbf{M}_{\mathrm{mf}}\boldsymbol{\mu}_0, \mathbf{M}_{\mathrm{mf}}\boldsymbol{\Gamma}_0\mathbf{M}_{\mathrm{mf}}^{\mathsf{H}}, \mathbf{M}_{\mathrm{mf}}\mathbf{C}_0\mathbf{M}_{\mathrm{mf}}^{\mathsf{T}}\right).$$

### 3.4.1 Symbol-based Detection

Symbol-based detection starts by computing the posterior marginal of each symbol with

$$s_i \sim p(s_i\,|\,\mathbf{y}) = \mathcal{CN}\left([\boldsymbol{\mu}_{\mathrm{s}}]_i, [\boldsymbol{\Gamma}_{\mathrm{s}}]_{ii}, [\mathbf{C}_{\mathrm{s}}]_{ii}\right) \tag{3.74}$$

and then it chooses the nearest point in the constellation, i.e.

$$\hat{s}_i = \arg\max_{\tilde{s}_i \in \mathcal{S}} p(\tilde{s}_i\,|\,\mathbf{y}) = \arg\max_{\tilde{s}_i \in \mathcal{S}} \mathcal{CN}\left(\tilde{s}_i\,|\,[\boldsymbol{\mu}_{\mathrm{s}}]_i, [\boldsymbol{\Gamma}_{\mathrm{s}}]_{ii}, [\mathbf{C}_{\mathrm{s}}]_{ii}\right) \tag{3.75}$$

where $\mathcal{S}$ is the constellation.

### 3.4.2 Viterbi Detection

If $p(\mathbf{s}\,|\,\mathbf{y})$ follows a Markov structure with memory $L \geq 0$, then we can define the ordered set

$$\mathcal{I}_i = \begin{cases} \emptyset, & \text{if } i = 1 \\ (1, \ldots, i-1), & \text{if } i < L \\ (i-L, \ldots, i-1), & \text{if } i \geq L \end{cases}$$

as the indices of memory buffer for the $i$-th symbol. It follows $p(\mathbf{s}\,|\,\mathbf{y})$ can be factored as

$$p(\mathbf{s}\,|\,\mathbf{y}) = \prod_{i=1}^{n} p(s_i\,|\,\mathbf{y}, \mathbf{s}_{\mathcal{I}_i}) = \prod_{i=1}^{n} \frac{p\left(\mathbf{s}_{\mathcal{I}_i \cup \{i\}}\,|\,\mathbf{y}\right)}{p\left(\mathbf{s}_{\mathcal{I}_i}\,|\,\mathbf{y}\right)} = \prod_{i=1}^{n} \frac{\mathcal{CN}\left(\mathbf{s}_{\mathcal{I}_i \cup \{i\}}\,|\,\boldsymbol{\mu}_{\mathrm{s}}, \boldsymbol{\Gamma}_{\mathrm{s}}, \mathbf{C}_{\mathrm{s}}\right)}{\mathcal{CN}\left(\mathbf{s}_{\mathcal{I}_i}\,|\,\boldsymbol{\mu}_{\mathrm{s}}, \boldsymbol{\Gamma}_{\mathrm{s}}, \mathbf{C}_{\mathrm{s}}\right)}.$$

Thus, MAP estimation yields

$$\hat{\mathbf{s}} = \arg\max_{\tilde{\mathbf{s}} \in \mathcal{S}^n} p(\tilde{\mathbf{s}}\,|\,\mathbf{y}) = \arg\max_{\tilde{\mathbf{s}} \in \mathcal{S}^n} \sum_{i=1}^{n} \psi_i([\tilde{\mathbf{s}}_{\mathcal{I}_i}, \tilde{s}_i]), \tag{3.76}$$

where $\psi_i$ is the branch metric

$$\psi_i([\tilde{\mathbf{s}}_{\mathcal{I}_i}, \tilde{s}_i]) \triangleq \log\left(\mathcal{CN}\left(\tilde{\mathbf{s}}_{\mathcal{I}_i \cup \{i\}}\,\middle|\,\boldsymbol{\mu}_{\mathrm{s}}, \boldsymbol{\Gamma}_{\mathrm{s}}, \mathbf{C}_{\mathrm{s}}\right)\right) - \log\left(\mathcal{CN}\left(\tilde{\mathbf{s}}_{\mathcal{I}_i}\,|\,\boldsymbol{\mu}_{\mathrm{s}}, \boldsymbol{\Gamma}_{\mathrm{s}}, \mathbf{C}_{\mathrm{s}}\right)\right). \tag{3.77}$$

**Algorithm 8** Viterbi Detection Algorithm

Initialize $M^{(0)}(\emptyset) = 0$, $\hat{\mathbf{s}} = \mathbf{0}$
**for** $i = 1$ **to** $n$ **do**
  Initialize $M^{(i)}$, $\Pi^{(i)}$ to store best cumulated metric and best previous state
  **for** all possible states $[\tilde{\mathbf{s}}_{\mathcal{I}_i}, \tilde{s}_i]$ **do**
    let $\mathcal{P}$ be the set of possible previous states of $[\tilde{\mathbf{s}}_{\mathcal{I}_i}, \tilde{s}_i]$
    compute best metric and best previous state for $[\tilde{\mathbf{s}}_{\mathcal{I}_i}, \tilde{s}_i]$

$$M^{(i)}([\tilde{\mathbf{s}}_{\mathcal{I}_i}, \tilde{s}_i]) = \max_{[\tilde{\mathbf{s}}'_{\mathcal{I}_{i-1}}, \tilde{s}'_{i-1}] \in \mathcal{P}} \left[ M^{(i-1)}([\tilde{\mathbf{s}}'_{\mathcal{I}'_{i-1}}, \tilde{s}'_{i-1}]) + \psi_i([\tilde{\mathbf{s}}_{\mathcal{I}_i}, \tilde{s}_i]) \right]$$

$$\Pi^{(i)}([\tilde{\mathbf{s}}_{\mathcal{I}_i}, \tilde{s}_i]) = \arg \max_{[\tilde{\mathbf{s}}'_{\mathcal{I}'_{i-1}}, \tilde{s}'_{i-1}] \in \mathcal{P}} \left[ M^{(i-1)}([\tilde{\mathbf{s}}'_{\mathcal{I}_{i-1}}, \tilde{s}_{i-1}]) + \psi_i([\tilde{\mathbf{s}}_{\mathcal{I}_i}, \tilde{s}_i]) \right]$$

  **end for**
**end for**
set $\hat{\mathbf{s}}_{n-L:n} = \arg \max_{\tilde{\mathbf{s}} \in \mathcal{S}^{L+1}} M^{(n)}(\tilde{\mathbf{s}})$, trace back $\left\{\Pi^{(i)}\right\}_{i=1}^{n}$ to fill $\hat{\mathbf{s}}$.
**return** $\hat{\mathbf{s}}$

The total search space is $\mathcal{S}^n$ and the search space for each step is $\mathcal{S}^{L+1}$. When $L = 0$, the Viterbi algorithm detection degenerates to symbol-based detection.

To reduce the complexity of Viterbi algorithm, the previously decoded symbol can be used to decode the current symbol. For the decision-directed detection, the search space for each step is $\mathcal{S}$.

**Algorithm 9** Decision-Directed Detection

Initialize $\hat{\mathbf{s}} = \mathbf{0}$
**for** $i = 1$ **to** $n$ **do**
  set $\hat{s}_i = \arg \max_{\tilde{s}_i \in \mathcal{S}} \psi_i([\hat{\mathbf{s}}_{\mathcal{I}_i}, \tilde{s}_i])$
**end for**
**return** $\hat{\mathbf{s}}$

### 3.4.3 Successive-Cancellation Detection

While the symbol-based detection ignores the correlation among symbols, successive cancellation can be applied to utilize this side information.

Using conditional distribution of multivariate complex Gaussian distribution, suppose $Z \sim \mathcal{CN}(\mu, \mathbf{\Gamma}, \mathbf{C})$ and let $Z_a = Z_{\sim i}$, which is the vector excluding the $i$-th element, and

$Z_b = Z_i$. Then according to (A.14), (A.15), (A.16)

$$\begin{aligned}
\mathbf{P}_i &= \overline{\mathbf{\Gamma}}_{ii} - \overline{\mathbf{C}}_{ii}\mathbf{\Gamma}_{ii}^{-1}\mathbf{C}_{ii} = \overline{\mathbf{\Gamma}}_{ii} - \overline{\mathbf{C}}_{ii}\mathbf{\Gamma}_{ii}^{-1}\mathbf{C}_{ii} = \frac{|\mathbf{\Gamma}_{ii}|^2 - |\mathbf{C}_{ii}|^2}{\mathbf{\Gamma}_{ii}}, \\
\mathbf{A}_{\sim i,i} &= \left(\mathbf{\Gamma}_{\sim i,i} - \mathbf{C}_{\sim i,i}\overline{\mathbf{\Gamma}}_{ii}^{-1}\overline{\mathbf{C}}_{ii}\right)\overline{\mathbf{P}}_i^{-1} = \frac{\overline{\mathbf{\Gamma}}_{ii}\mathbf{\Gamma}_{\sim i,1} - \overline{\mathbf{C}}_{ii}\mathbf{C}_{\sim i,1}}{|\mathbf{\Gamma}_{ii}|^2 - |\mathbf{C}_{ii}|^2}, \\
\mathbf{B}_{\sim i,i} &= \left(\mathbf{C}_{\sim i,i} - \mathbf{\Gamma}_{\sim i,i}\mathbf{\Gamma}_{ii}^{-1}\mathbf{C}_{ii}\right)\mathbf{P}_i^{-1} = \frac{\mathbf{\Gamma}_{ii}\mathbf{C}_{\sim i,1} - \mathbf{C}_{ii}\mathbf{\Gamma}_{\sim i,1}}{|\mathbf{\Gamma}_{ii}|^2 - |\mathbf{C}_{ii}|^2}.
\end{aligned}$$

Conditioned on $Z_i = z_i$, one can use (3.23)–(3.25), to compute the mean and covariance matrices

$$\boldsymbol{\mu}_{Z_{\sim i}|Z_i=z_i} = \boldsymbol{\mu}_{\sim i} + \mathbf{A}_{\sim i,i}(z_i - \mu_i) + \mathbf{B}_{\sim i,i}(\overline{z}_i - \overline{\mu}_i), \tag{3.78}$$

$$\mathbf{\Gamma}_{Z_{\sim i}|Z_i=z_i} = \mathbf{\Gamma}_{\sim i,\sim i} - \mathbf{A}_{\sim i,i}\mathbf{\Gamma}_{\sim i,1}^{\mathsf{H}} - \mathbf{B}_{\sim i,i}\mathbf{C}_{\sim i,1}^{\mathsf{H}}, \tag{3.79}$$

$$\mathbf{C}_{Z_{\sim i}|Z_i=z_i} = \mathbf{C}_{\sim i,\sim i} - \mathbf{A}_{\sim i,i}\mathbf{C}_{\sim i,1}^{\mathsf{T}} - \mathbf{B}_{\sim i,i}\mathbf{\Gamma}_{\sim i,1}^{\mathsf{T}}. \tag{3.80}$$

Moreover, suppose the exact value of $Z_i$ is not known, but its distribution is given as $\pi_{Z_i}$. Then, the conditional distribution will be a mixture of complex Gaussian distributions. However, a complex Gaussian distribution can be still used as an approximation via matching the first and second order moments. By the law of total variance, this gives

$$\boldsymbol{\mu}_{Z_{\sim i}|Z_i\sim\pi_{Z_i}} = \sum_{z_i} \pi_{Z_i}(z_i)\, \boldsymbol{\mu}_{Z_{\sim i}|Z_i=z_i} \tag{3.81}$$

$$\begin{aligned}
\mathbf{\Gamma}_{Z_{\sim i}|Z_i\sim\pi_{Z_i}} &= \sum_{z_i} \pi_{Z_i}(z_i)\mathbf{\Gamma}_{Z_{\sim i}|Z_i=z_i} \\
&\quad + \sum_{z_i} \pi_{Z_i}(z_i)\left\{\left(\boldsymbol{\mu}_{Z_{\sim i}|Z_i=z_i} - \boldsymbol{\mu}_{Z_{\sim i}|Z_i\sim\pi_{Z_i}}\right)\left(\boldsymbol{\mu}_{Z_{\sim i}|Z_i=z_i} - \boldsymbol{\mu}_{Z_{\sim i}|Z_i\sim\pi_{Z_i}}\right)^{\mathsf{H}}\right\}
\end{aligned} \tag{3.82}$$

$$\begin{aligned}
\mathbf{C}_{Z_{\sim i}|Z_i\sim\pi_{Z_i}} &= \sum_{z_i} \pi_{Z_i}(z_i)\mathbf{C}_{Z_{\sim i}|Z_i=z_i} \\
&\quad + \sum_{z_i} \pi_{Z_i}(z_i)\left\{\left(\boldsymbol{\mu}_{Z_{\sim i}|Z_i=z_i} - \boldsymbol{\mu}_{Z_{\sim i}|Z_i\sim\pi_{Z_i}}\right)\left(\boldsymbol{\mu}_{Z_{\sim i}|Z_i=z_i} - \boldsymbol{\mu}_{Z_{\sim i}|Z_i\sim\pi_{Z_i}}\right)^{\mathsf{T}}\right\}.
\end{aligned} \tag{3.83}$$

To be more specific, the following algorithm describes the successive cancellation. In each step, the most reliable symbol is chosen for signal detection by minimizing the posterior variance. Then the mean and covariance matrices are updated using the corresponding parameters of the conditional distribution.

**Algorithm 10** Successive-Cancellation Detection

Initialize $\hat{\mathbf{s}} = \mathbf{0}, \boldsymbol{\mu}^{(0)} = \boldsymbol{\mu}_{\mathrm{s}}, \boldsymbol{\Gamma}^{(0)} = \boldsymbol{\Gamma}_{\mathrm{s}}, \mathbf{C}^{(0)} = \mathbf{C}_{\mathrm{s}}, \mathcal{A} = \{1, \ldots, n\}$.
**for** $t = 1$ **to** $n$ **do**
  **for** $a \in \mathcal{A}$ **do**
    compute posterior of $s_a$, where $a$ is the $i_a$-th smallest in $\mathcal{A}$

$$\pi_{s_a}(\tilde{s}) = \mathcal{CN}\left(\tilde{s} \,\middle|\, \boldsymbol{\mu}_{ia}^{(t-1)}, \boldsymbol{\Gamma}_{i_a,i_a}^{(t-1)}, \mathbf{C}_{i_a,i_a}^{(t-1)}\right)$$

    compute posterior variance $\tau_a^2 = \sum_{\tilde{s}\in\mathcal{S}} \tilde{s}^2 \pi_{s_a}(\tilde{s}) - \left[\sum_{\tilde{s}\in\mathcal{S}} \tilde{s}\pi_{s_a}(\tilde{s})\right]^2$
  **end for**
  choose most reliable symbol $a_* = \min_{a\in\mathcal{A}} \tau_a^2$.
  make decision for symbol $s_{a_*}$: $\hat{s}_{a_*} = \arg\max_{\tilde{s}\in\mathcal{S}} \pi_{s_{a_*}}(\tilde{s})$
  update $\boldsymbol{\mu}^{(t)}, \boldsymbol{\Gamma}^{(t)}, \mathbf{C}^{(t)}$ on the $i_{a_*}$-th remaining symbol using (3.81), (3.82), (3.83),
  update set of remaining symbols $\mathcal{A} \leftarrow \mathcal{A} \setminus \{a\}$.
**end for**
**return** $\hat{\mathbf{s}}$

## 3.5 Experimental Results

In the pulse shaping step, the transmitter uses a root-raised-cosine pulse with a rolloff factor of 0.25. Each frame is sent with 16 QAM modulation, and has length $K = 256$ symbols. This signal is input to the channel with link length $L = 80$ km and $N_{\mathrm{sp}} = 40$ spans. The parameters used for single-mode fiber are $\alpha = 0.2\,\mathrm{dB/km}$, $\beta_2 = -21.668\,\mathrm{ps}^2/\mathrm{km}$, $\gamma = 1.3\,(\mathrm{W}\cdot\mathrm{km})^{-1}$. Propagation in the fibers is simulated using the SSFM with a segment length indicated by [ZH08], and same segment lengths are used for simulating the channel and for all decoding methods. We consider symbol error rate (SER) as our metric to capture the absolute performance of each detector.

For CBP algorithm, we compare the SER performance for different symbol detectors. Fig 3.2 shows that due to the correlation between real and imaginary parts of the same symbol, even the symbol-based (SB) detector outperforms DBP. Since CBP provides the full description of the correlation among all symbols as well as their real and imaginary parts, the curves of decision-directed (DD) detector, Viterbi algorithm (VA) detector, and successive cancellation (SC) detector indicate that larger gain can be obtained by utilizing the covariance information in a more accurate way.

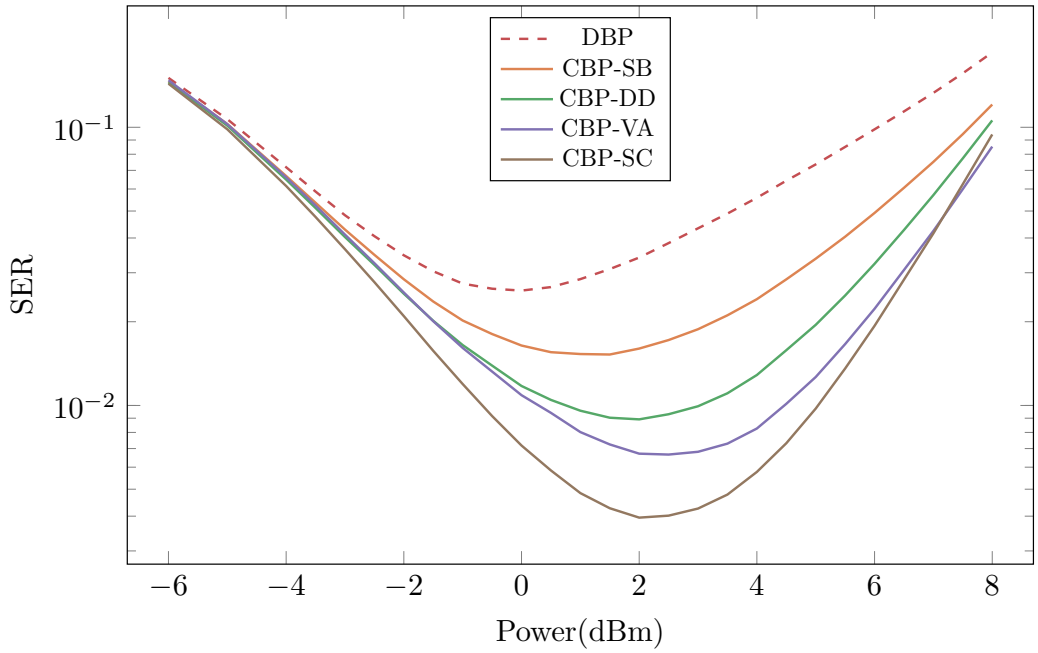



**Figure 3.2**: Comparison of CBP algorithm using different symbol detectors.

For SDBP, the size of particle filters is set to be 512 to match similar computation complexity as CBP algorithm. We compare the best symbol detector for both SDBP and CBP. It can be seen that CBP with successive cancellation detector has lower SER when power is less than 5 dBm. When the power is higher, Gaussian approximation assumption is broken because of the heavy nonlinearity, which explains why the SDBP algorithm with Viterbi detection outperforms CBP.

## 3.6 Conclusion

In this chapter, we investigated the possibility of using multivariate complex Gaussian distributions to approximate the posterior distribution in a fiber-optic signal-detection problem. A covariance back-propagation algorithm is proposed using a complex Gaussian approximation of the SDBP solution. The gain in the symbol error rate comes from the fact that the covariance back-propagation algorithm captures both the correlation between different symbols and the correlation between real and imaginary parts of the same symbol.

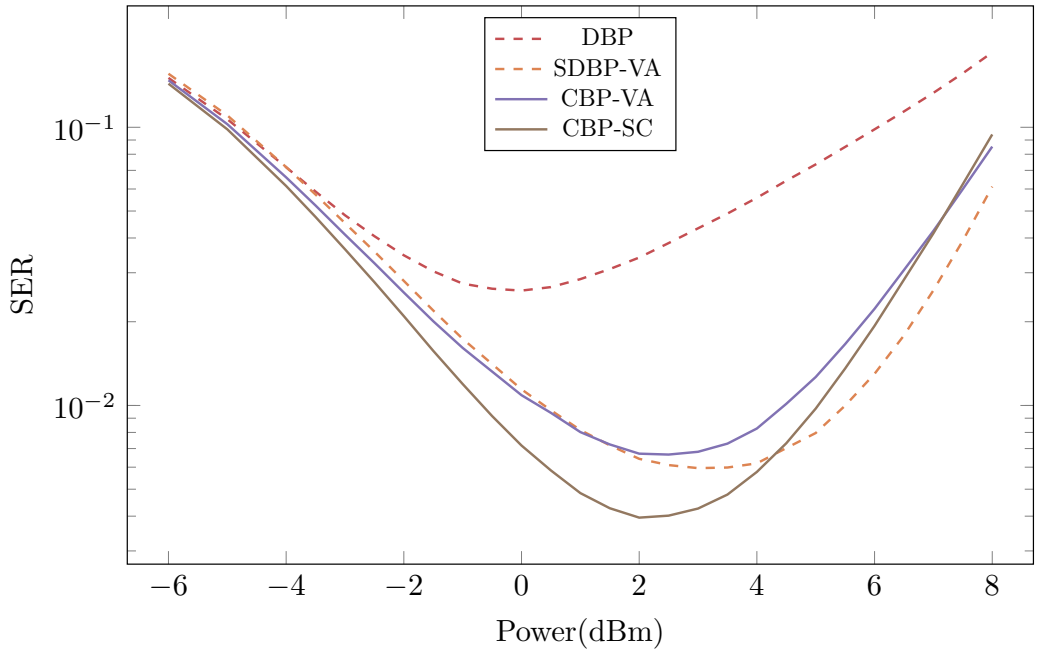



**Figure 3.3**: Comparison between CBP and SDBP

Compared to SDBP with Viterbi symbol detector, for high power region, SDBP shows some advantages because the nonlinearity is too strong for the Gaussian assumption to hold. On the other hand, in low/mid power regions with $P \leq$ 4dBm, covariance back-propagation with successive cancellation symbol detection has a better SER performance because of the comprehensive information contained in the covariance matrix.

# Chapter 4

# Decoding Reed–Muller Codes Using Redundant Code Constraints

## 4.1 Introduction

Reed–Muller (RM) codes were introduced by Muller in [Mul54] and the first decoding algorithm was given by Reed [Ree54]. Under maximum-likelihood (ML) decoding, RM codes achieve capacity on the binary erasure channel [KKM$^+$16] and they also provide excellent performance at low to medium block lengths over the additive white Gaussian noise (AWGN) channel [YA19]. Due to the intractability of ML decoding, an important question is how to approach ML performance with reasonable decoding complexity in practice. Many recent approaches exploit the highly symmetric structure of RM codes in the form of their large automorphism group[1] [SHP18, HDMG18, YA19]. This typically results in code representations with many redundant code constraints, e.g., overcomplete parity-check matrices where the number of rows is much larger than the rank [BH06, SHP18].

This chapter starts by focusing on the recursive projection–aggregation (RPA) algorithm that was recently introduced in [YA19]. RPA decoding achieves excellent performance on low-rate (2nd- and 3rd-order) RM codes. It also significantly outperforms comparable polar codes under successive cancellation list decoding. In this work, we show that the RPA algorithm has a natural interpretation on a factor graph with generalized check constraints and can be seen as a modified version of (weighted) belief propagation (BP) decoding using many redundant code constraints. We use these observations to improve the RPA algorithm

[1] The automorphism group of a code $\mathcal{C}$ is defined as $\{\pi \in \mathcal{S}_N \,|\, \mathbf{x}^\pi \in \mathcal{C}, \forall \mathbf{x} \in \mathcal{C}\}$, where $\mathcal{S}_N$ is the symmetric group on $N$ elements, i.e., $\pi \in \mathcal{S}_N$ is a bijective mapping (or permutation) from $[N]$ to itself, and $\mathbf{x}^\pi$ denotes a permuted vector, i.e., $x_i^\pi = x_{\pi(i)}$.

and to introduce a novel decoding approach tailored to high-rate codes.

To begin, we provide a simple overview of RM codes that illustrates how their recursive structure and their large automorphism group together imply that they satisfy a very large set of code constraints. We then give a high-level description of the algorithms and contributions in this paper with the help of the RM code table shown in Fig. 4.1. To that end, let $\mathrm{RM}(r,m) \subseteq \mathbb{F}_2^N$ denote the set of codewords in the $r$-th order RM code of length $N = 2^m$. The well-known recursive definition of $\mathrm{RM}(r,m)$ is given by [MS78, p. 374]

$$\{(\mathbf{u}, \mathbf{u}+\mathbf{v}) \,|\, \mathbf{u} \in \mathrm{RM}(r,m-1), \mathbf{v} \in \mathrm{RM}(r-1,m-1)\}\,, \tag{4.1}$$

where $(\mathbf{u},\mathbf{w})$ denotes vector concatenation. For $\mathbf{u},\mathbf{w} \in \mathbb{F}_2^{N/2}$ with $(\mathbf{u},\mathbf{w}) \in \mathrm{RM}(r,m)$, this automatically implies that

- $\mathbf{u} \in \mathrm{RM}(r,m-1)$, i.e., *puncturing* the second half of the codeword gives shorter RM codeword of the same order, and
- $\mathbf{v} = \mathbf{u}+\mathbf{w} \in \mathrm{RM}(r-1,m-1)$, i.e., summing the two halves of the codeword *projects* onto shorter RM code with reduced order.

Of course, these statements remain true even after reordering the code bits using a permutation in the code's automorphism group. Thus, there are in fact many different puncturing and projection patterns that result in RM subcodes. With this insight, one can further show that it is possible to define $\mathrm{RM}(r,m)$ either using only $\mathrm{RM}(r,m-1)$ or only $\mathrm{RM}(r-1,m-1)$ subcode constraints.

The RPA algorithm is based on the latter approach and the corresponding factor graph contains generalized check nodes associated with all possible $\mathrm{RM}(r-1,m-1)$ subcodes. These subcodes are decoded recursively until one reaches a 1st-order (i.e., Hadamard) code, for which there exist efficient decoders based on the fast Hadamard transform (FHT) [MS78]. The schematic decoding path taken by RPA is illustrated by the solid red line in Fig. 4.1.

The decoding complexity of RPA increases significantly with each additional recursion stage and RPA decoding is thus limited to low-rate codes in practice. To alleviate this prob-

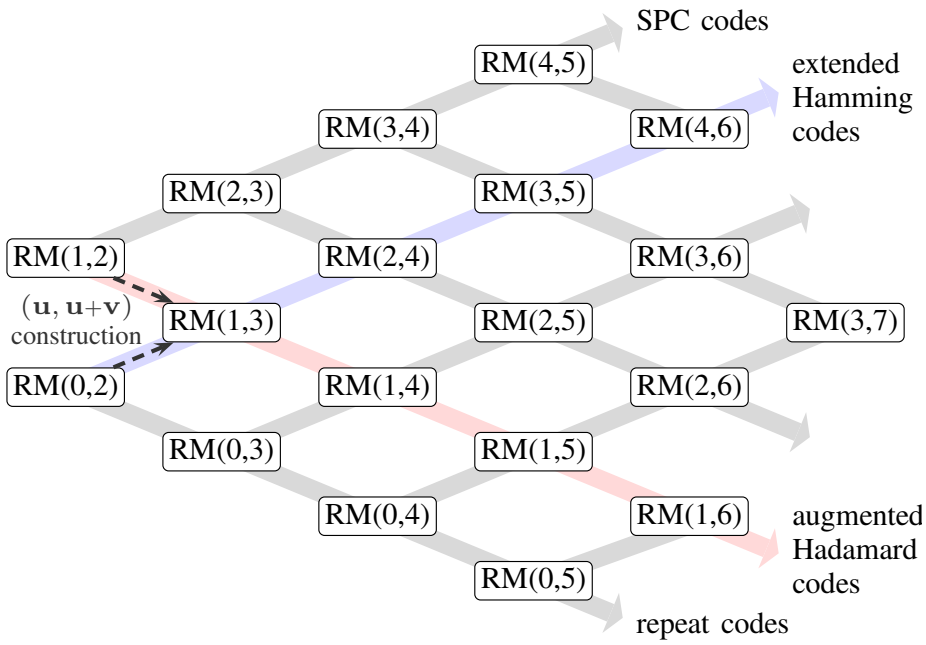



**Figure 4.1**: Standard tableau of Reed–Muller (RM) codes and schematic illustration of all algorithms considered in this chapter

lem, we propose a new algorithm that is similar in spirit to RPA but relies on puncturing instead of projections. The new algorithm is called recursive puncturing–aggregation (RXA) and allows for efficient decoding of high-rate codes. RXA uses all possible RM$(r, m-1)$ subcode constraints and recursively traverses upwards in the RM tableau, as illustrated by the solid blue line in Fig. 4.1. The base case for RXA are RM$(r, r+2)$ (i.e., extended Hamming) codes which can also be decoded using FHTs. For the self-dual RM codes, one can decode using either projection or puncturing with $m = 5, 7, 9$. Simulation results we show that both the decoding complexity and performance are essentially the same.

We further investigate collapsed, i.e., non-recursive, versions of RPA and RXA, respectively. We start by providing a theoretical justification for using collapsed algorithms by showing that multiple recursive stages result in decoders that reuse the same RM subcodes. We then propose two new algorithms, called collapsed projection–aggregation (CPA) and collapsed puncturing–aggregation (CXA), and show that they can achieve similar performance as their recursive counterparts with lower decoding complexity. CPA and CXA

correspond to the dashed lines in Fig. 4.1 and directly project or puncture onto the base codes.

Lastly, we connect RPA and the proposed algorithms in this paper to previous decoding approaches based on over-complete parity-check matrices that contain all minimum-weight dual codewords as rows [BH06, SHP18]. In particular, we show that the factor graph for CXA assuming one additional stage of puncturing (i.e., to single parity-checks instead of extended Hamming codes, see the dotted line in Fig. 4.1) is equivalent to the factor graph that contains all minimum-weight parity-checks (MWPCs).

In summary, the contributions in this work are as follows:

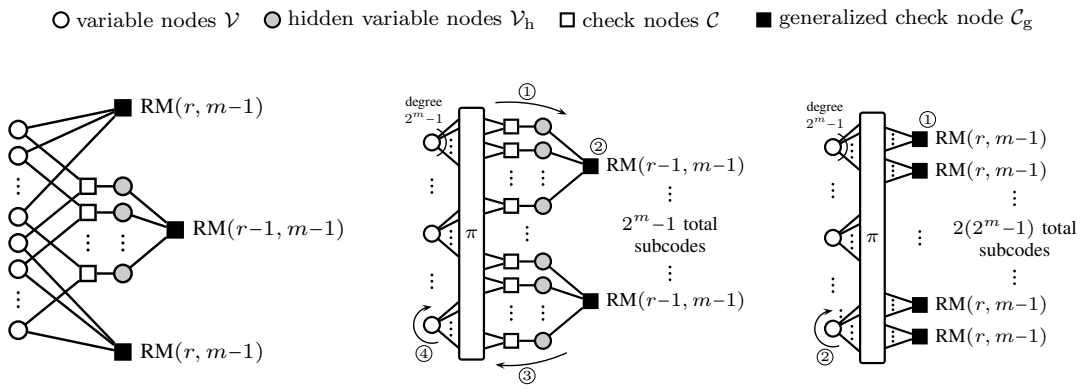
- We show that RPA is closely related to BP decoding by interpreting it as a message-passing algorithm on a highly redundant factor graph representation of the code.
- We propose a new algorithm, called RXA, which is based on puncturing instead of projections and allows for efficient decoding of high-rate RM codes.
- We propose collapsed versions of RPA and RXA, called CPA and CXA, respectively, and show that they sometimes achieve similar performance with lower complexity.
- We highlight the connections between this work, RPA, and previous approaches based on using all MWPCs.

## 4.2 Factor Graphs for Reed–Muller Codes

In order to compare RPA and BP decoding, we start by reviewing various factor graph representations of $\mathrm{RM}(r, m)$. For an introduction to factor graphs, see [KFL01a].

In general, the factor graphs in this chapter contains of four distinct nodes types, which are collected in the sets $(\mathcal{V}, \mathcal{C}, \mathcal{V}_{\mathrm{h}}, \mathcal{C}_{\mathrm{g}})$:

- $\mathcal{V}$: variable nodes (VNs), corresponding to the code bits of $\mathrm{RM}(r, m)$, where $|\mathcal{V}| = 2^m$
- $\mathcal{C}$: check nodes (CNs), corresponding to projections, i.e., the summation of code bits




(a) $(\mathbf{u}, \mathbf{u}+\mathbf{v})$ construction (b) RPA (c) RXA

**Figure 4.2**: Factor graphs for RM$(r, m)$.

- $\mathcal{V}_\mathrm{h}$: hidden VNs (of degree 2), corresponding to the code bits of RM$(r-1, m-1)$ subcodes
- $\mathcal{C}_\mathrm{g}$: generalized CNs, corresponding to either RM$(r-1, m-1)$ or RM$(r, m-1)$ subcode constraints

The number of nodes of each type and the graph connectivity (including the node degrees) depend on the particular code representation. As an example, the factor graph that can be inferred from the $(\mathbf{u}, \mathbf{u}+\mathbf{v})$ construction in (4.1) is shown in Fig. 4.2(a). It consists of one generalized CN corresponding to RM$(r, m-1)$ for the first codeword half and one generalized CN corresponding to RM$(r-1, m-1)$ that constrains the sum of the two codeword halves. Moreover, one may use the fact that RM$(r-1, m) \subset$ RM$(r, m)$ [MS78, p. 377] to see that the second half of the codeword also forms a valid codeword in RM$(r, m-1)$, leading to one additional subcode constraint.

### 4.2.1 Enumerate Distinct Subspaces

For the CPA, CXA algorithms, it is necessary to enumerate all distinct subspaces of $\mathbb{F}_2^m$ according to dimension. Thus, we propose an efficient algorithm to enumerate all $d$-dimensional subspaces in $\mathbb{F}_2^m$.

**Definition 6** (RREF-index)**.** *A d-dimensional subspace* $\mathbb{B} = \mathsf{span}\,(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ *is indexed by an ordered set, the "RREF-index",* $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ *if*

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}_d^\mathsf{T} \\ \vdots \\ \mathbf{b}_1^\mathsf{T} \end{bmatrix}$$

*is in reduced row-echelon form.*

Inspired by the Pascal identities for the Gaussian binomial coefficients

$$\binom{m}{d}_2 = \binom{m}{d-1}_2 + 2^{m-d}\binom{m-1}{d-1}_2,$$

where

$$\binom{m}{d}_2 \triangleq \begin{cases} \frac{(1-2^m)(1-2^{m-1})\cdots(1-2^{m-d+1})}{(1-2)(1-2^2)\cdots(1-2^d)}, & \text{if } d \le m \\ 0, & \text{if } d > m \end{cases}.$$

We observe that the set of all distinct $d$-dimensional subspaces in $\mathbb{F}_2^m$ can be divided into two groups:

- subspaces that are orthogonal to $\mathbf{e}_1$, i.e. the first element of $\mathbf{b}_d$ is 0. If we drop the first coordinate (since it is always zero), then these subspaces reduce to $(d-1)$-dimensional subspaces in $\mathbb{F}_2^{m-1}$.

- subspace that are not orthogonal to $\mathbf{e}_1$, i.e. the first element of $\mathbf{b}_d$ is 1. If we drop the first coordinate, then these subspaces reduce to $(d-1)$-dimensional subspaces in $\mathbb{F}_2^{m-1}$. To guarantee that $\mathbf{B}$ matrix is RREF, $\mathbf{b}_d$ must be zero at the position of lower rows' leading ones. Therefore, there are $2^{(m-1)-(d-1)} = 2^{m-d}$ choices of $[\mathbf{b}_d]_{2:m}$.

Denote $\mathcal{B}(d, m)$ to be the set of $d$-dimensional subspaces in $\mathbb{F}_2^m$, we propose the following recursive algorithm to enumerate all distinct $d$-dimensional subspaces.

**Algorithm 11** `EnumSub` Algorithm

**Input:** subspace dimension $d$, vector space $\mathbb{F}_2^m$
**Output:** $\mathcal{B}(d, m)$

```
if d = 1 then
    B = {(e_i) | i = 1, ..., m}
else if d = m then
    B = {(e_m, e_{m-1}, ..., e_1)}
else
    B(d, m-1) = EnumSub(d, m-1)
    B(d-1, m-1) = EnumSub(d-1, m-1)
    for B ∈ B(d, m-1) do
        Let (b_1, ..., b_d) be the RREF index of B
        B = B ∪ { ( [0; b_1], ..., [0; b_{d-1}], [0; b_d] ) }
    end for
    for B ∈ B(d-1, m-1) do
        Let (b_1, ..., b_{d-1}) be the RREF index of B
        compute index of leading ones J
        J = { min{ j | [b_i]_j = 1 } | i = 1, ..., d-1 }
        for b' ∈ { z ∈ F_2^{m-1} | z_J = 0 } do
            B = B ∪ { ( [0; b_1], ..., [0; b_{d-1}], [1; b'] ) }
        end for
    end for
end if
return ℓ̂
```

### 4.2.2 Subcode Indexing

#### Projection-Puncture System

The vector space $\mathbb{E} = \mathbb{F}_2^m$ has rich geometric properties and is closely related to the structure of RM codes. Due to the large automorphism group of RM codes, there are many choices to index the code bits. This makes it difficult to apply recursive projection.

**Definition 7** (Projection-Puncture System)**.** *A degree-d projection-puncture system over* $\mathbb{E} = \mathbb{F}_2^m$ *is a tuple of matrices* $\mathcal{P} = (\mathbf{U}, \mathbf{V}, \mathbf{W})$ *where*

$$\mathbf{U} \in \mathbb{F}_2^{d \times m}, \quad \mathbf{V} \in \mathbb{F}_2^{(m-d) \times m}, \quad \mathbf{W} \in \mathbb{F}_2^{d \times m}$$

*and*

$$\mathsf{rank}\left(\begin{bmatrix}\mathbf{U}\\ \mathbf{V}\end{bmatrix}\right) = m, \quad \mathbf{U}\mathbf{V}^{\mathsf{T}} = \mathbf{0}, \quad \mathbf{U}\mathbf{W}^{\mathsf{T}} = \mathbf{I}$$

*such that*

$$\mathbf{U}\left(\mathbf{W}^{\mathsf{T}}\mathbf{s} + \mathbf{V}^{\mathsf{T}}\mathbf{t}\right) = \mathbf{s}, \qquad \forall\, \mathbf{s} \in \mathbb{F}_2^d, \quad \mathbf{t} \in \mathbb{F}_2^{m-d}$$

Let $\mathbb{U}$, $\mathbb{V}$, $\mathbb{W}$ be the vector spaces spanned by rows of $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$, respectively. Then, it is easy to see that

- $\mathbb{V}$ and $\mathbb{U}$ are orthogonal complement vector space in $\mathbb{F}_2^m$.
- $\mathbf{U}\colon \mathbb{E} \to \mathbb{F}_2^d$ is a surjective linear map, and splits $\mathbb{F}_2^m$ according to evaluation $\mathbf{s} \in \mathbb{F}_2^d$ in the form $\mathbf{W}^{\mathsf{T}}\mathbf{s} + \mathbb{V} \in \mathbb{E}/\mathbb{U}^{\perp}$.
- Project $\mathbb{F}_2^m$ on $\mathbb{W}$, it splits $\mathbb{F}_2^m$ into $2^{m-d}$ cosets indexed by $\mathbf{t} \in \mathbb{F}_2^{m-d}$ in the form $\mathbb{W} + \mathbf{V}^{\mathsf{T}}\mathbf{t} \in \mathbb{E}/\mathbb{W}$.

**Theorem 8.** *Projection-puncture systems* $\mathcal{P} = (\mathbf{U}, \mathbf{V}, \mathbf{W})$ *and* $\tilde{\mathcal{P}} = \left(\tilde{\mathbf{U}}, \tilde{\mathbf{V}}, \tilde{\mathbf{W}}\right)$ *are equivalent if and only if there exists* $\mathbf{A} \in \mathsf{GL}(2, d)$ *and* $\mathbf{B} \in \mathsf{GL}(2, m-d)$ *satisfying*

$$\tilde{\mathbf{U}} = \mathbf{A}\mathbf{U}, \quad \tilde{\mathbf{V}} = \mathbf{B}\mathbf{V}, \quad \tilde{\mathbf{W}} = \left(\mathbf{A}^{-1}\right)^{\mathsf{T}}\mathbf{W}$$

*Proof.* Since matrices $\mathbf{A}$ and $\mathbf{B}$ are non-singular, it is easy to see the row span of $\tilde{\mathbf{U}}$, $\tilde{\mathbf{V}}$, $\tilde{\mathbf{W}}$ are same as $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}$, and hence

$$\mathsf{rank}\left(\begin{bmatrix}\tilde{\mathbf{U}}\\ \tilde{\mathbf{V}}\end{bmatrix}\right) = \mathsf{rank}\left(\begin{bmatrix}\mathbf{U}\\ \mathbf{V}\end{bmatrix}\right) = m$$

Moreover,

$$\tilde{\mathbf{U}}\tilde{\mathbf{V}}^{\mathsf{T}} = \mathbf{A}\mathbf{U}\mathbf{V}^{\mathsf{T}}\mathbf{B}^{\mathsf{T}} = \mathbf{A}\,\mathbf{0}\,\mathbf{B}^{\mathsf{T}} = \mathbf{0}$$

$$\tilde{\mathbf{U}}\tilde{\mathbf{W}}^{\mathsf{T}} = \mathbf{A}\mathbf{U}\mathbf{W}^{\mathsf{T}}\mathbf{A}^{-1} = \mathbf{A}\,\mathbf{I}\,\mathbf{A}^{-1} = \mathbf{I}$$

□

The above theorem shows that the projection-puncture system is not unique for given $d$ and $m$. However, once $\mathcal{P} = (\mathbf{U}, \mathbf{V}, \mathbf{W})$ is fixed, it defines a bijection between position $\mathbf{z} \in \mathbb{F}_2^m$ and evaluation-coset index $(\mathbf{s}, \mathbf{t}) \in \mathbb{F}_2^d \times \mathbb{F}_2^{m-d}$. For example,

- In one direction, given $(\mathbf{s}, \mathbf{t})$, we can compute $\mathbf{z} = \mathbf{W}^\mathsf{T}\mathbf{s} + \mathbf{V}^\mathsf{T}\mathbf{t}$.
- In the other direction, given $\mathbf{z}$, we can compute $\mathbf{s} = \mathbf{U}\mathbf{z}$, and $\mathbf{t} = \mathbf{V}^* \left(\mathbf{I} - \mathbf{W}^\mathsf{T}\mathbf{U}\right) \mathbf{z}$, where $\mathbf{V}^*$ is complete as follows. First, do elementary row operations $\mathbf{R}$ and apply a column permutation $\mathbf{Q}$ such that $\mathbf{RVQ} = \begin{bmatrix} \mathbf{I} & \mathbf{P} \end{bmatrix}$. Then, choose $\mathbf{V}^* = \mathbf{R}^\mathsf{T} \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{Q}^\mathsf{T}$ such that

$$\mathbf{V}^*\mathbf{V}^\mathsf{T} = \mathbf{R}^\mathsf{T} \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{Q}^\mathsf{T}\mathbf{Q} \begin{bmatrix} \mathbf{I} \\ \mathbf{P}^\mathsf{T} \end{bmatrix} \left(\mathbf{R}^{-1}\right)^\mathsf{T} = \mathbf{I}$$

This explicit indexing scheme is quite useful when we use recursive projection and puncturing to decode subcode codewords. Given codeword $\mathbf{c} = (\mathbf{c}(\mathbf{z}), \mathbf{z} \in \mathbb{F}_2^m)$, under the above projection-puncture system $\mathcal{P}$ we can define

$$\mathbf{c}_\mathbf{s}^{(\mathcal{P})}(\mathbf{t}) = \mathbf{c}\left(\mathbf{W}^\mathsf{T}\mathbf{s} + \mathbf{V}^\mathsf{T}\mathbf{t}\right), \qquad \forall\, \mathbf{s} \in \mathbb{F}_2^d, \quad \mathbf{t} \in \mathbb{F}_2^{m-d}, \tag{4.2}$$

where $\mathbf{c}_\mathbf{s}^{(\mathcal{P})}$ is the punctured sub-codeword of $\mathbf{c}$ obtained by choosing positions $\mathbf{z}$ with $\mathbf{U}\mathbf{z} = \mathbf{s}$, and this sub-codeword is indexed by $\mathbf{t}$.

Moreover, the projected sub-codeword obtained by projecting on $\mathbb{W}$ is defined by

$$\mathbf{c}_{/\mathbb{W}}^{(\mathcal{P})}(\mathbf{t}) = \bigoplus_{\mathbf{s} \in \mathbb{F}_2^d} \mathbf{c}\left(\mathbf{W}^\mathsf{T}\mathbf{s} + \mathbf{V}^\mathsf{T}\mathbf{t}\right), \qquad \forall\, \mathbf{t} \in \mathbb{F}_2^{m-d} \tag{4.3}$$

which is the sum of all punctured sub-codewords followed by the linear map $\mathbf{U}$.

### Subcode Indexing for Projection

For subcode projection, each sub-codeword is determined by the subspace $\mathbb{W}$ on which the projection occurs. Consider a $d$-dimensional subspace $\mathbb{B}$ with RREF index $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$. We then can construct the corresponding projection-puncture system $\mathcal{P} = (\mathbf{U}, \mathbf{V}, \mathbf{W})$ such that

- let $\mathbf{W} = [\mathbf{b}_d, \ldots, \mathbf{b}_1]^\mathsf{T}$,
- rewrite $\mathbf{W} = \begin{bmatrix} \mathbf{I} & \mathbf{P} \end{bmatrix} \mathbf{Q}$, where $\mathbf{P} \in \mathbb{F}_2^{d \times (m-d)}$ and $\mathbf{Q}$ represents column permutation, and
- choose $\mathbf{U}$, $\mathbf{V}$ such that $\begin{bmatrix} \mathbf{U}^\mathsf{T} & \mathbf{V}^\mathsf{T} \end{bmatrix} = \mathbf{Q}^\mathsf{T}$, $\mathbf{V}^* = \mathbf{V}$.

Let $T_\mathbf{t} \in \mathbb{F}_2^m / \mathbb{B}$ be the coset of $\mathbb{B}$ indexed by $\mathbf{t} \in \mathbb{F}_2^{m-d}$. Then, it can be simply computed using

$$T_\mathbf{t} = \left\{ \mathbf{z} \in \mathbb{F}_2^m \,\middle|\, \exists\, \mathbf{s} \in \mathbb{F}_2^d \text{ s.t. } \mathbf{z} = \mathbf{W}^\mathsf{T}\mathbf{s} + \mathbf{V}^\mathsf{T}\mathbf{t} \right\}$$

### Subcode Indexing for Puncturing

For subcode puncturing, each sub-codeword is determined by the linear equation $\mathbf{U}$ and the evaluation $\mathbf{s}$. Consider an $d$-dimensional subspace $\mathbb{B}$ with RREF index $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$, we can construct the corresponding projection-puncture system $\mathcal{P} = (\mathbf{U}, \mathbf{V}, \mathbf{W})$ such that we

- let $\mathbf{U} = [\mathbf{b}_d, \ldots, \mathbf{b}_1]^\mathsf{T}$.
- rewrite $\mathbf{U} = \begin{bmatrix} \mathbf{I} & \mathbf{P} \end{bmatrix} \mathbf{Q}$, where $\mathbf{P} \in \mathbb{F}_2^{d \times (m-d)}$ and $\mathbf{Q}$ represents column permutation.
- choose $\mathbf{V}$ and $\mathbf{W}$ such that

$$\mathbf{V} = \begin{bmatrix} \mathbf{P}^\mathsf{T} & \mathbf{I} \end{bmatrix} \mathbf{Q}, \quad \mathbf{W} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{Q}, \quad \mathbf{V}^* = \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{Q}$$

Let $\tilde{T}_\mathbf{s} \in \mathbb{F}_2^m / \mathbb{B}^\perp$ be the coset of $\mathbb{B}^\perp$ indexed by $\mathbf{s} \in \mathbb{F}_2^d$. Then, it can be simply computed using

$$\tilde{T}_\mathbf{s} = \left\{ \mathbf{z} \in \mathbb{F}_2^m \,\middle|\, \exists\, \mathbf{t} \in \mathbb{F}_2^{m-d} \text{ s.t. } \mathbf{z} = \mathbf{W}^\mathsf{T}\mathbf{s} + \mathbf{V}^\mathsf{T}\mathbf{t} \right\}$$

## 4.2.3 Redundant Factor Graphs

Strictly speaking, the factor graph in Fig. 4.2 (a) is already redundant, in the sense that either one of the two generalized CNs corresponding to $\mathrm{RM}(r, m-1)$ could be removed. There are two obvious redundant factor graphs

- One is based on projecting $\mathrm{RM}(r, m)$ onto $2^m - 1$ $\mathrm{RM}(r-1, m-1)$ codes,
- One is based on the $2(2^m - 1)$ $\mathrm{RM}(r, m-1)$ codes via puncturing $\mathrm{RM}(r, m)$.

For a sparse factor graph with tree-like neighborhoods, it is well-understood why the standard BP decoder performs well [RU01]. However, since redundant factor graphs generally have many more cycles, the decoding performance can be significantly improved by modifying the standard BP decoder [SHP18]. It particular, a standard approach is to add weights that reduce the overconfidence induced by correlated messages.

### 4.2.4 Redundant Factor Graph Construction

#### Factor graph for one stage of RPA

The factor graph for the RPA algorithm, is a redundant factor graph based on projecting $\mathrm{RM}(r, m)$ onto $2^m - 1$ $\mathrm{RM}(r-1, m-1)$ subcodes.

Let $\mathcal{B}(m, 1)$ be the set of distinct one-dimensional subspaces in $\mathbb{F}_2^m$, the factor graph can be construct by the following sets of nodes and edges:

- $\mathcal{V} = \{v_{\mathbf{z}} \,|\, \mathbf{z} \in \mathbb{F}_2^m\}$: VNs corresponding to the code bits of $\mathrm{RM}(r, m)$,
- $\mathcal{C} = \bigcup_{\mathbb{B} \in \mathcal{B}(m,1)} \mathcal{C}_{/\mathbb{B}}$, where $\mathcal{C}_{/\mathbb{B}} = \left\{c_{/\mathbb{B},T} \,\middle|\, T \in \mathbb{F}_2^m/\mathbb{B}\right\}$ is the set of CNs corresponding to projections on subspace $\mathbb{B}$,
- $\mathcal{V}_{\mathrm{h}} = \bigcup_{\mathbb{B} \in \mathcal{B}(m,1)} \mathcal{V}_{\mathrm{h},/\mathbb{B}}$, where $\mathcal{V}_{\mathrm{h},/\mathbb{B}} = \left\{v_{\mathrm{h},/\mathbb{B},T} \,\middle|\, T \in \mathbb{F}_2^m/\mathbb{B}\right\}$ is the set of VNs corresponding to $\mathrm{RM}(r-1, m-1)$ subcodes projected on subspace $\mathbb{B}$,
- $\mathcal{C}_{\mathrm{g}} = \left\{c_{\mathrm{g},/\mathbb{B}} \,\middle|\, \mathbb{B} \in \mathcal{B}(m, 1)\right\}$: generalized CNs for $\mathrm{RM}(r-1, m-1)$ subcodes,
- $\mathcal{E} = \bigcup_{\mathbb{B} \in \mathcal{B}(m,1)} \mathcal{E}_{/\mathbb{B}}$, where $\mathcal{E}_{/\mathbb{B}} = \left\{(v_{\mathbf{z}}, c_{/\mathbb{B},T}) \,\middle|\, T \in \mathbb{F}_2^m/\mathbb{B}, \mathbf{z} \in T\right\}$ is the set of edges to form code projections w.r.t. subspace $\mathbb{B}$,
- $\mathcal{E}_{\mathrm{h}} = \bigcup_{\mathbb{B} \in \mathcal{B}(m,1)} \mathcal{E}_{\mathrm{h},/\mathbb{B}}$, where $\mathcal{E}_{\mathrm{h},/\mathbb{B}} = \left\{(v_{\mathrm{h},/\mathbb{B},T}, c_{/\mathbb{B},T}) \,\middle|\, T \in \mathbb{F}_2^m/\mathbb{B}\right\}$ is the set of edges connecting projection check nodes and subcode variable nodes w.r.t. subspace $\mathbb{B}$,

- $\mathcal{E}_{\mathrm{g}} = \bigcup_{\mathbb{B}\in\mathcal{B}(m,1)} \mathcal{E}_{\mathrm{g},/\mathbb{B}}$, where $\mathcal{E}_{\mathrm{g},/\mathbb{B}} = \left\{(v_{\mathrm{h},/\mathbb{B},T}, c_{\mathrm{g},/\mathbb{B}}) \,\middle|\, T \in \mathbb{F}_2^m/\mathbb{B}\right\}$ is the set of edges connecting subcode variable nodes and their corresponding generalized check nodes w.r.t. subspace $\mathbb{B}$.

### Factor graph for one stage of RXA

The factor graph for the RXA algorithm is a redundant factor graph based on puncturing $\mathrm{RM}(r,m)$ using $2(2^m-1)$ distinct patterns that result in $\mathrm{RM}(r,m-1)$ subcodes.

Let $\mathcal{B}(m,1)$ be the set of distinct one-dimensional subspaces in $\mathbb{F}_2^m$. Then, the factor graph can be constructed using the following sets of nodes and edges:

- $\mathcal{V} = \{v_{\mathbf{z}} \,|\, \mathbf{z} \in \mathbb{F}_2^m\}$: VNs corresponding to the code bits of $\mathrm{RM}(r,m)$,
- $\mathcal{C}_{\mathrm{g}} = \bigcup_{\mathbb{B}\in\mathcal{B}(m,1)} \left\{c_{\mathrm{g},\mathbb{B},\tilde{T}} \,\middle|\, \tilde{T} \in \mathbb{F}_2^m/\mathbb{B}^{\perp}\right\}$, where $c_{\mathrm{g},\mathbb{B},\tilde{T}}$ is the generalized CN for $\mathrm{RM}(r,m-1)$ subcode punctured by selecting VNs in $\mathcal{V}_{\mathbb{B},\tilde{T}} = \left\{\mathbf{z} \in \mathbb{F}_2^m \,\middle|\, \mathbf{z} \in \tilde{T}\right\}$ for $s \in \mathbb{F}_2$,
- $\mathcal{E}_{\mathrm{g}} = \bigcup_{\mathbb{B}\in\mathcal{B}(m,1)} \bigcup_{\tilde{T}\in\mathbb{F}_2^m/\mathbb{B}^{\perp}} \mathcal{E}_{\mathrm{g},\mathbb{B},\tilde{T}}$, where $\mathcal{E}_{\mathrm{g},\mathbb{B},\tilde{T}} = \left\{(v, c_{\mathrm{g},\mathbb{B},\tilde{T}}) \,\middle|\, v \in \mathcal{V}_{\mathbb{B},\tilde{T}}\right\}$ is the set of edges representing puncturing pattern $\tilde{T}$.

### Factor graph for CPA

The factor graph for the CPA algorithm, is a redundant factor graph based on projecting onto all possible distinct $\mathrm{RM}(1,m-r+1)$ subcodes. Let $\mathcal{B}(m,r-1)$ be the set of distinct $(r-1)$-dimensional subspaces in $\mathbb{F}_2^m$. Then, the factor graph can be constructed using the following sets of nodes and edges:

- $\mathcal{V} = \{v_{\mathbf{z}} \,|\, \mathbf{z} \in \mathbb{F}_2^m\}$: VNs corresponding to the code bits of $\mathrm{RM}(r,m)$,
- $\mathcal{C} = \bigcup_{\mathbb{B}\in\mathcal{B}(m,r-1)} \mathcal{C}_{/\mathbb{B}}$, where $\mathcal{C}_{/\mathbb{B}} = \left\{c_{/\mathbb{B},T} \,\middle|\, T \in \mathbb{F}_2^m/\mathbb{B}\right\}$ is the set of CNs corresponding to projections on $(r-1)$-dimensional subspace $\mathbb{B}$,
- $\mathcal{V}_{\mathrm{h}} = \bigcup_{\mathbb{B}\in\mathcal{B}(m,r-1)} \mathcal{V}_{\mathrm{h},/\mathbb{B}}$, where $\mathcal{V}_{\mathrm{h},/\mathbb{B}} = \left\{v_{\mathrm{h},/\mathbb{B},T} \,\middle|\, T \in \mathbb{F}_2^m/\mathbb{B}\right\}$ is the set of VNs corresponding to $\mathrm{RM}(1,m-r+1)$ subcodes projected on subspace $\mathbb{B}$,

- $\mathcal{C}_{\mathrm{g}} = \left\{ c_{\mathrm{g},/\mathbb{B}} \,\middle|\, \mathbb{B} \in \mathcal{B}(m, r-1) \right\}$: generalized CNs for RM$(r-1, m-1)$ subcodes,
- $\mathcal{E} = \bigcup_{\mathbb{B} \in \mathcal{B}(m,r-1)} \mathcal{E}_{/\mathbb{B}}$, where $\mathcal{E}_{/\mathbb{B}} = \left\{ (v_{\mathbf{z}}, c_{/\mathbb{B},T}) \,\middle|\, T \in \mathbb{F}_2^m/\mathbb{B}, \mathbf{z} \in T \right\}$ is the set of edges to form code projections w.r.t. subspace $\mathbb{B}$,
- $\mathcal{E}_{\mathrm{h}} = \bigcup_{\mathbb{B} \in \mathcal{B}(m,r-1)} \mathcal{E}_{\mathrm{h},/\mathbb{B}}$, where $\mathcal{E}_{\mathrm{h},/\mathbb{B}} = \left\{ (v_{\mathrm{h},/\mathbb{B},T}, c_{/\mathbb{B},T}) \,\middle|\, T \in \mathbb{F}_2^m/\mathbb{B} \right\}$ is the set of edges connecting projection check nodes and subcode variable nodes w.r.t. subspace $\mathbb{B}$,
- $\mathcal{E}_{\mathrm{g}} = \bigcup_{\mathbb{B} \in \mathcal{B}(m,r-1)} \mathcal{E}_{\mathrm{g},/\mathbb{B}}$, where $\mathcal{E}_{\mathrm{g},/\mathbb{B}} = \left\{ (v_{\mathrm{h},/\mathbb{B},T}, c_{\mathrm{g},/\mathbb{B}}) \,\middle|\, T \in \mathbb{F}_2^m/\mathbb{B} \right\}$ is the set of edges connecting subcode variable nodes and their corresponding generalized check nodes w.r.t. subspace $\mathbb{B}$.

### Factor graph for CXA

The factor graph for the CXA algorithm is a redundant factor graph based on puncturing down to all possible distinct RM$(r, r+2)$ subcodes. Let $\mathcal{B}(m, m-r-2)$ be the set of distinct $(m-r-2)$-dimensional subspaces in $\mathbb{F}_2^m$. Then, the factor graph can be constructed using the following sets of nodes and edges:

- $\mathcal{V} = \{ v_{\mathbf{z}} \,|\, \mathbf{z} \in \mathbb{F}_2^m \}$: VNs corresponding to the code bits of RM$(r, m)$,
- $\mathcal{C}_{\mathrm{g}} = \bigcup_{\mathbb{B} \in \mathcal{B}(m,m-r-2)} \left\{ c_{\mathrm{g},\mathbb{B},\tilde{T}} \,\middle|\, \tilde{T} \in \mathbb{F}_2^m/\mathbb{B}^{\perp} \right\}$, where $c_{\mathrm{g},\mathbb{B},\tilde{T}}$ is the generalized CN for RM$(r, r+2)$ subcode punctured by selecting VNs in $\mathcal{V}_{\mathbb{B},\tilde{T}} = \left\{ \mathbf{z} \in \mathbb{F}_2^m \,\middle|\, \mathbf{z} \in \tilde{T} \right\}$ for $s \in \mathbb{F}_2$,
- $\mathcal{E}_{\mathrm{g}} = \bigcup_{\mathbb{B} \in \mathcal{B}(m,m-r-2)} \bigcup_{\tilde{T} \in \mathbb{F}_2^m/\mathbb{B}^{\perp}} \mathcal{E}_{\mathrm{g},\mathbb{B},\tilde{T}}$, where $\mathcal{E}_{\mathrm{g},\mathbb{B},\tilde{T}} = \left\{ (v, c_{\mathrm{g},\mathbb{B},\tilde{T}}) \,\middle|\, v \in \mathcal{V}_{\mathbb{B},\tilde{T}} \right\}$ is the set of edges representing puncturing pattern $\tilde{T}$.

## 4.2.5 Weighted Belief Propagation Update Equations

One can run BP decoding once the factor graph is defined, here we consider weight BP update equations.

Check-to-variable-node step:

$$\hat{\lambda}_{c\to v}^{(t)} = 2\tanh^{-1}\left(\prod_{v'\in\partial c\setminus v}\tanh\left(\frac{\lambda_{v'\to c}^{(t)}}{2}\right)\right) \tag{4.4}$$

For fully-weighted BP, the VN update is given by [NML+18]

$$\lambda_{v\to c}^{(t)} = w_v^{(t)}\ell_v + \sum_{c'\in\partial v\setminus c} w_{vc}^{(t)}\hat{\lambda}_{c'\to v}^{(t)} \tag{4.5}$$

We also use the a simple-scaling model [LCHP19].

## 4.3 Recursive Decoding Algorithms

For notation convenience, in the following context we use shorthand notations for arbitrary subset of check nodes $\mathcal{C}'$ and subset of variable nodes $\mathcal{V}'$

$$\boldsymbol{\lambda}_{\mathcal{V}'\to\mathcal{C}'} = \left\{\lambda_{v\to c}\,\middle|\, v\in\mathcal{V}', c\in\mathcal{C}', v \text{ connects to } c\right\}$$
$$\hat{\boldsymbol{\lambda}}_{\mathcal{C}'\to\mathcal{V}'} = \left\{\hat{\lambda}_{c\to v}\,\middle|\, v\in\mathcal{V}', c\in\mathcal{C}', v \text{ connects to } c\right\}$$

to represent the ordered set[2] of messages passed between sets of nodes.

### 4.3.1 Recursive Projection–Aggregation

The RPA algorithm can be described using the following steps. The projection step projects $\boldsymbol{\lambda}_{\mathcal{V}\to\mathcal{C}}^{(t-1)}$ into $(2^m-1)$ subcode LLR sequences based on $(2^m-1)$ distinct one-dimensional subspaces. More precisely, given a $d$-dimensional subspace $\mathbb{B}$ one can construct its corresponding projection-puncture system $\mathcal{P}=(\mathbf{U},\mathbf{V},\mathbf{W})$ according to section 4.2.2. Then, we define

$$\boldsymbol{\ell}_{/\mathbb{B}} = \texttt{project}(\boldsymbol{\lambda}_{\mathcal{V}\to\mathcal{C}}, \mathbb{B})$$

[2]we always choose $\mathcal{V}'$ to form a subcode, we neglect the subcode bit order in this section for high-level description, details can be found in appendix.

where according to equation (4.2) and (4.4)

$$[\boldsymbol{\ell}_{/\mathbb{B}}]_{\mathbf{t}} = 2\tanh^{-1}\left(\prod_{\mathbf{s}\in\mathbb{F}_2^d}\tanh\left(\frac{\lambda_{(\mathbf{W}^\intercal\mathbf{s}+\mathbf{V}^\intercal\mathbf{t})\to(\mathbb{B},\mathbf{t})}}{2}\right)\right), \quad \forall\,\mathbf{t}\in\mathbb{F}_2^{m-d} \tag{4.6}$$

The recursion step decodes the subcode LLR sequences and returns the extrinsic symbol-wise APP. The backward BP step collects BP messages $\boldsymbol{\lambda}_{\mathcal{V}\to\mathcal{C}}^{(t-1)}$ and $\boldsymbol{\lambda}_{\mathcal{V}_\mathrm{h}\to\mathcal{C}}^{(t)}$ to update $\hat{\boldsymbol{\lambda}}_{\mathcal{C}\to\mathcal{V}}^{(t)}$, again according to (4.4). The forward BP step uses $\hat{\boldsymbol{\lambda}}_{\mathcal{C}\to\mathcal{V}}^{(t)}$ to compute the marginal and update $\boldsymbol{\lambda}_{\mathcal{V}\to\mathcal{C}}^{(t)}$ for the next iteration. There are two versions of the forward BP step

- in `RPA_RM` [3], only node-based information is stored, i.e. for $v\in\mathcal{V}$

$$\lambda_{v\to c}^{(t)} = \frac{1}{2^m-1}\sum_{c\in\partial v}\hat{\lambda}_{c\to v}^{(t)}, \quad \forall\, c\in\partial v \tag{4.7}$$

- in `RPA_BP`, edge-based information is stored, (4.5) is used with given scale factors, i.e. for $v\in\mathcal{V}$

$$\lambda_{v\to c}^{(t)} = \ell_v + \frac{1}{2^m-1}\sum_{c'\in\partial v\setminus c}\hat{\lambda}_{c'\to v}^{(t)}, \quad \forall\, c\in\partial v \tag{4.8}$$

Finally, the aggregation step computes the approximated posterior marginal for the decoded codeword by

$$\hat{\boldsymbol{\ell}} = \texttt{aggregation}(\boldsymbol{\ell}, \hat{\boldsymbol{\lambda}}_{\mathcal{C}\to\mathcal{V}}), \tag{4.9}$$

where

$$\hat{\ell}_v = \ell_v + \frac{1}{2^m-1}\sum_{c\in\partial v}\hat{\lambda}_{c\to v}^{(t)}, \quad v\in\mathcal{V}. \tag{4.10}$$

[3] A comparison between RPA and previous decoding algorithms can be found in [YA19, Sec. V-B].

**Table 4.1**: Comparison between BP and RPA

| | step | nodes | BP | RPA |
|---|---|---|---|---|
| 1. | check node update | $\forall c \in \mathcal{C},\, v \in \mathcal{V}_{\mathrm{h}}$ | compute (4.4) | same as BP |
| 2. | component decoding | $\forall v \in \mathcal{V}_{\mathrm{h}}, c \in \mathcal{C}_{\mathrm{g}}$ | extrinsic symbol-wise APP | RPA or codeword ML for 1st-order RM |
| 3. | check node update | $\forall c \in \mathcal{C},\, v \in \mathcal{V}$ | compute (4.4) | same as BP (but one message is binary) |
| 4. | variable node update | $\forall c \in \mathcal{C},\, v \in \mathcal{V}$ | $\lambda_{v\to c}^{(t)} = \ell_v + \sum_{c' \in \partial v \backslash c} \lambda_{c'\to v}^{(t)}$ | $\lambda_{v\to c}^{(t)} = \frac{1}{2^m-1} \sum_{c' \in \partial v} \lambda_{c'\to v}^{(t)}$ |

**Algorithm 12** `RPA` Algorithm

**Input:** LLR $\boldsymbol{\ell}$, Reed-Muller parameter $r, m$
**Output:** posterior LLR $\hat{\boldsymbol{\ell}}$

```
if r = 1 then
    ℓ̂ = FHT_RM1_decode(ℓ)
else
    initialize BP messages λ^(0)_{v→c} = ℓ_v for v ∈ V, c ∈ ∂v
    for t = 1, ..., T_max do
        for i = 1, ..., 2^m − 1 do
            ℓ_{/B_i} = project(λ^(t−1)_{V→C}, B_i) by Eq (4.6)
            ℓ̂_{/B_i} = RPA(ℓ_{/B_i}, r − 1, m − 1)
        end for
        λ^(t)_{V_h→C} = {ℓ̂_{/B_i}}_{i=1}^{2^m−1}
        update λ̂^(t)_{C→V} by Eq (4.4) using λ^(t−1)_{V→C}, λ^(t)_{V_h→C}
        update λ^(t)_{V→C} by Eq (4.7) or (4.8) using ℓ, λ̂^(t)_{C→V}
        ℓ̂ = aggregation(ℓ, λ̂^(t)_{C→V}), by Eq (4.10)
    end for
end if
return ℓ̂
```

### 4.3.2 Recursive Puncturing–Aggregation

Since there is no projections, the factor graph of the RXA algorithm only contains the variable node set $\mathcal{V}$ and the generalized check node set $\mathcal{C}_{\mathrm{g}}$.

The RXA algorithm can be described in following steps. The puncturing step punctures $\boldsymbol{\lambda}_{\mathcal{V}\to\mathcal{C}_{\mathrm{g}}}^{(t-1)}$ into $2(2^m-1)$ subcode LLR sequences based on $(2^m-1)$ distinct one-dimensional subspaces, which are simply permutation of a subset of $\boldsymbol{\lambda}_{\mathcal{V}\to\mathcal{C}_{\mathrm{g}}}^{(t-1)}$. More precisely, given a $d$-dimensional subspace $\mathbb{B}$ one can construct its corresponding projection-puncture system

$\mathcal{P} = (\mathbf{U}, \mathbf{V}, \mathbf{W})$ according to section 4.2.2. Then, we define

$$\{\boldsymbol{\ell}_{\mathbb{B},\mathbf{s}}\}_{\mathbf{s}\in\mathbb{F}_2^d} = \texttt{puncture}(\boldsymbol{\lambda}_{\mathcal{V}\to\mathcal{C}}, \mathbb{B})$$

where according to equation (4.3)

$$[\boldsymbol{\ell}_{\mathbb{B},\mathbf{s}}]_{\mathbf{t}} = \lambda_{(\mathbf{W}^\intercal\mathbf{s}+\mathbf{V}^\intercal\mathbf{t})\to(\mathbb{B},\mathbf{s},\mathbf{t})}, \quad \forall\, \mathbf{t} \in \mathbb{F}_2^{m-d} \tag{4.11}$$

The recursion step decodes the subcode LLR sequences and return the extrinsic symbol-wise APP. The forward BP step uses $\hat{\boldsymbol{\lambda}}^{(t)}_{\mathcal{C}_g\to\mathcal{V}}$ to update $\boldsymbol{\lambda}^{(t)}_{\mathcal{V}\to\mathcal{C}_\mathrm{g}}$ by Eq (4.8). Finally, the aggregation step computes the approximated posterior marginal for the decoded codeword by Eq (4.10).

**Algorithm 13** `RXA` Algorithm

**Input:** LLR $\boldsymbol{\ell}$, Reed-Muller parameter $r, m$
**Output:** posterior LLR $\hat{\boldsymbol{\ell}}$

```
if r = 1 then
    ℓ̂ = FHT_extHamming_decode(ℓ)
else
    initialize BP messages λ^(0)_{v→c} = ℓ_v for v ∈ V, c ∈ ∂v
    for t = 1, ..., T_max do
        ℓ_{B_i,0}, ℓ_{B_i,1} = puncture(λ^(t−1)_{V→C}, B_i), by Eq (4.11)
        for s ∈ F_2 do
            ℓ̂_{B_i,s} = RXA(ℓ_{/B_i,s}, r − 1, m − 1)
        end for
        λ̂^(t)_{C_g→V} = {ℓ̂_{B_i,0}, ℓ̂_{B_i,1}}_{i=1}^{2^m−1}
        ℓ̂ = aggregation(ℓ, λ̂^(t)_{C→V}), by Eq (4.10)
    end for
end if
return ℓ̂
```

## 4.4 Non-Recursive (Collapsed) Algorithms

While a collapsed version of RPA is briefly suggested in [YA19, Sec. VI], no algorithm or simulation results are provided. To avoid decoding the same subcodes multiple times, one can collapse the recursion and construct a redundant factor graph whose generalized check nodes are the base case subcodes $\mathrm{RM}(1, m-r+1)$ or $\mathrm{RM}(r, r+2)$

### 4.4.1 Collapsed RPA

The CPA algorithm is almost the same as RPA algorithm except that, instead of recursion, it directly projects to $\mathrm{RM}(1, m-r+1)$. Let $\mathcal{B}(r-1,m)$ be the set of all distinct $(r-1)$-dimensional subspaces in $\mathbb{F}_2^m$.

**Algorithm 14** `CPA` Algorithm

**Input:** LLR $\boldsymbol{\ell}$, Reed-Muller parameter $r, m$
**Output:** posterior LLR $\hat{\boldsymbol{\ell}}$
  initialize BP messages $\lambda_{v\to c}^{(0)} = \ell_v$ for $v \in \mathcal{V}, c \in \partial v$
  **for** $t = 1, \ldots, T_{\max}$ **do**
    **for** $i = 1, \ldots, 2^m - 1$ **do**
      $\boldsymbol{\ell}_{/\mathbb{B}_i} = \texttt{project}(\boldsymbol{\lambda}_{\mathcal{V}\to\mathcal{C}}^{(t-1)}, \mathbb{B}_i)$ by Eq (4.6)
      $\hat{\boldsymbol{\ell}}_{/\mathbb{B}_i} = \texttt{FHT\_RM1\_decode}(\boldsymbol{\ell}_{/\mathbb{B}_i})$
    **end for**
    $\boldsymbol{\lambda}_{\mathcal{V}_\mathrm{h}\to\mathcal{C}}^{(t)} = \{\hat{\boldsymbol{\ell}}_{/\mathbb{B}_i}\}_{i=1}^{2^m-1}$
    update $\hat{\boldsymbol{\lambda}}_{\mathcal{C}\to\mathcal{V}}^{(t)}$ by Eq (4.4) using $\boldsymbol{\lambda}_{\mathcal{V}\to\mathcal{C}}^{(t-1)}$, $\boldsymbol{\lambda}_{\mathcal{V}_\mathrm{h}\to\mathcal{C}}^{(t)}$
    update $\boldsymbol{\lambda}_{\mathcal{V}\to\mathcal{C}}^{(t)}$ by Eq (4.7) or (4.8) using $\boldsymbol{\ell}$, $\hat{\boldsymbol{\lambda}}_{\mathcal{C}\to\mathcal{V}}^{(t)}$
    $\hat{\boldsymbol{\ell}} = \texttt{aggregation}(\boldsymbol{\ell}, \hat{\boldsymbol{\lambda}}_{\mathcal{C}\to\mathcal{V}}^{(t)})$, by Eq (4.10)
  **end for**
  **return** $\hat{\boldsymbol{\ell}}$

### 4.4.2 Collapsed Puncturing

Similar to the CPA algorithm, the CXA algorithm collapses the recursion in RXA algorithm and directly decode $\mathrm{RM}(r, r+2)$. Let $\mathcal{B}(m-r-2,m)$ be the set of all distinct $(m-r-2)$-dimensional subspaces in $\mathbb{F}_2^m$. Then, each subspace in $\mathcal{B}(m-r-2,m)$ can puncture a codeword in $\mathrm{RM}(r,m)$ into $2^{m-r-2}$ subcode codeword in $\mathrm{RM}(r, r+2)$.

**Algorithm 15** `CXA` Algorithm

**Input:** LLR $\boldsymbol{\ell}$, Reed-Muller parameter $r, m$
**Output:** posterior LLR $\hat{\boldsymbol{\ell}}$

```
initialize BP messages λ_{v→c}^{(0)} = ℓ_v for v ∈ V, c ∈ ∂v
for t = 1, ..., T_max do
    ℓ_{B_i,0}, ℓ_{B_i,1} = puncture(λ_{V→C}^{(t-1)}, B_i), by Eq (4.11)
    for s ∈ F_2 do
        ℓ̂_{B_i,s} = FHT_extHamming_decode(ℓ_{B_i,s})
    end for
    λ̂_{C_g→V}^{(t)} = {ℓ̂_{B_i,0}, ℓ̂_{B_i,1}}_{i=1}^{2^m-1}
    ℓ̂ = aggregation(ℓ, λ̂_{C→V}^{(t)}), by Eq (4.10)
end for
return ℓ̂
```

### 4.4.3 Connection to Decoding with MWPCs

Figure 4.1 show that single-parity-check codes are Reed-Muller codes $\mathrm{RM}(r, r+1)$. As discussed in previous section, $\mathrm{RM}(r, r+1)$ can be obtained by puncturing $\mathrm{RM}(r, r+2)$, which is just an extra puncturing stage for the RXA algorithm.

In another direction, for the RPA algorithm, we can project two more stages to obtain $\mathrm{RM}(-1, m-r-1)$ subcodes. $\mathrm{RM}(-1, m-r-1)$ is a trivial code that only contains the all-zero codeword, therefore each variable node in this subcode must be zero, and is obtained by summing $2^{r+1}$ variable nodes in $\mathrm{RM}(r, m)$. One can view variable nodes in these $\mathrm{RM}(-1, m-r-1)$ subcodes as the minimum-weight parity-checks.

Finally, weighted BP with the over-complete parity-check matrix [SHP18] is a variant of CXA algorithm that considers subspaces in $\mathcal{B}(m, m-r+1)$ instead of $\mathcal{B}(m, m-r+2)$, and uses (4.4) to decode the single parity-check subcodes.

## 4.5 Simulation Results

The described decoding algorithms for Reed-Muller codes are tested with code length 128, 256, and 512 over AWGN channels. For the low-order Reed–Muller codes ($\mathrm{RM}(2, m)$, $\mathrm{RM}(3, m)$), we compare the RPA_RM algorithm in [YA19] and the CPA algorithm, we also

consider the nested decoders wrapped with a list decoder. For the high-order Reed–Muller codes (RM$(m-2,m)$, RM$(m-3,m)$), we compare the CXA algorithm with its list decoder version. For all list decoders, the list size is set to be 16. Ordered statistics decoding (OSD) is used as a benchmark, whose performance is close to ML [FS95].

Due to the computational complexity, a self-dual Reed Muller code is used to compare performance between CPA and CXA, where RM$(3,7)$ is the smallest non-trivial case. For RM$(3,7)$, the algorithms RPA_RM, CPA, and CXA all perform similar. They also approach the OSD decoder performance when wrapped with a list decoder with list size 16. However, CPA and CXA has a lower computation complexity since their factor graphs are built with distinct subspaces, whereas RPA_RM will decode the same base case subcode codeword 3 times.

For CPA on RM$(2,8)$ and CXA on RM$(5,8)$, Fig 4.4 and 4.5 shows that the there is a 0.3 dB gap from the OSD decoder, and this gap disappears once the list decoder is used.

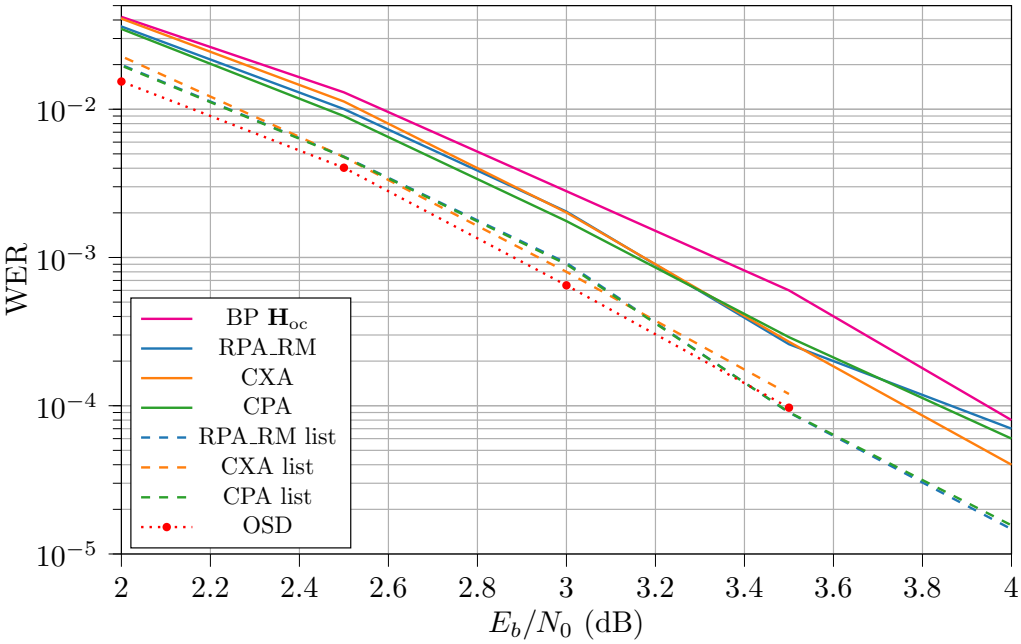



**Figure 4.3**: Comparison of RPA_RM, CPA, and CXA for RM$(3,7)$

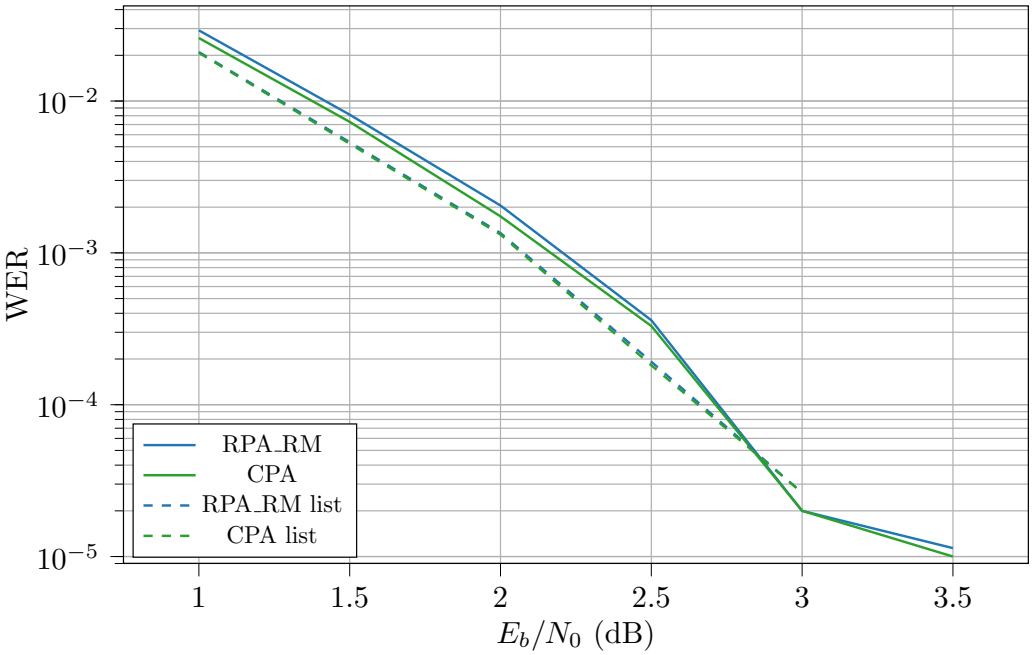



**Figure 4.4**: Comparison of RPA_RM, CPA, and CXA for RM$(2, 8)$

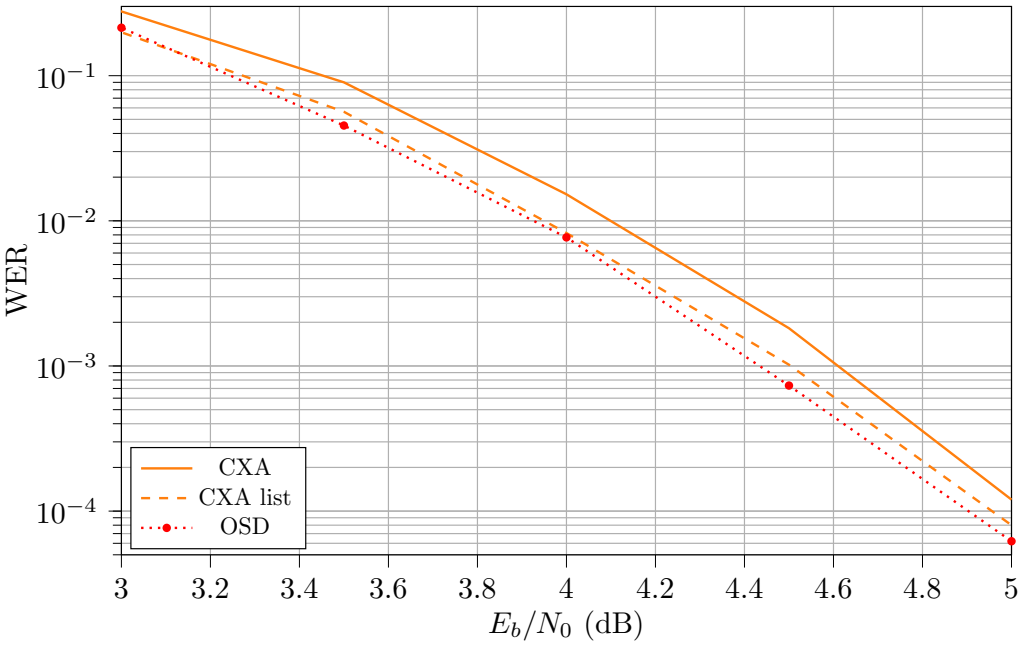



**Figure 4.5**: Comparison of RPA_RM, CPA, and CXA for RM$(5, 8)$

## 4.6 Conclusions and Future Work

In this chapter, we investigated the rich algebraic structure of Reed–Muller code family and connected the recent RPA decoding algorithm for Reed–Muller codes to message passing decoding on a factor graph, the RXA algorithm is also introduced to decode high-order Reed–Muller codes analogous to RPA_RM algorithm, using the duality between projection and puncturing.

To reduce the unnecessary redundant computation of subcode codewords, a novel indexing framework for the subcode is proposed to build the connection between recursive algorithm and their variants after collapsing the recursions. The CPA and CXA algorithms show their power by matching the performance of corresponding recursive algorithms with smaller computation complexity.

For future work, one might consider combinations of projection *and* puncturing. For example, the code $\mathrm{RM}(3,7)$ can be defined in terms of many redundant $\mathrm{RM}(2,5)$ code constraints by using one stage of both puncturing and projection. This would correspond to following zig-zag paths in the tableau in Fig. 4.1.

# Chapter 5

# Learned Belief–Propagation Decoding with Simple Scaling and SNR Adaptation

## 5.1 Introduction

Recent progress in machine learning and off-the-shelf learning packages have made it tractable to add many parameters to existing communication algorithms and optimize. One example of this approach is the weighted belief-propagation (WBP) decoder recently proposed by Nachmani, Be'ery, and Burshtein [NBB16], where different weights (or scale factors) are introduced for each edge in the Tanner graph. These weights are then optimized empirically using tools and software from deep learning. Their results show that WBP can provide significant gains over standard BP when applied to the parity-check (PC) matrices of short BCH codes. A more comprehensive treatment of this idea can be found in [NML$^+$18].

While the performance gains of WBP decoding are worth investigating, the additional complexity of optimizing, storing, and applying one weight per edge is significant. In this paper, we focus on *simple-scaling* models for WBP that share weights across edges to reduce the storage and computational burden. In these models, only three scalar parameters are used per iteration: a message weight, a channel weight, and a damping factor. We show that such simple-scaling models are often sufficient to obtain gains similar to the full parameterization.

For WBP, the average binary cross-entropy across bit positions is used as the optimization loss function in [NBB16, NML$^+$18]. This approach is also adopted in related works, see, e.g., [GCHtB17, BCK18]. On the other hand, we show that minimizing average binary cross-entropy does not necessarily minimize the bit error rate (BER). We propose a new

loss function which can be regarded as a "soft" version of BER. This loss function can lead to performance gains when optimizing WBP with highly redundant PC matrices, e.g., for Reed–Muller codes.

As a last contribution, we propose a simple solution to the problem that the optimal WBP parameters may be different for different signal-to-noise ratios (SNRs). In particular, we use a parameter adapter network (PAN) that learns the relation between the SNR and the optimal WBP parameters. The usefulness of this approach is illustrated with several examples.

## 5.2 Background

Consider an $(N, K)$ binary linear code $\mathcal{C}$ defined by an $M \times N$ PC matrix $\mathbf{H}$, where $N$ is the code length, $K$ is the code dimension, and $M \geq N - K$. We assume transmission over the additive white Gaussian noise channel according to $y_v = (-1)^{x_v} + z_v$, where $y_v$ is the $v$-th output symbol, $x_v$ is the corresponding bit in the transmitted codeword $\boldsymbol{x}$, $z_v \sim \mathcal{N}(0, (2RE_\mathrm{b}/N_0)^{-1})$, and $R = K/N$ is the code rate. We refer to $\rho \triangleq E_\mathrm{b}/N_0$ as the signal-to-noise ratio (SNR).

Given any PC matrix $\mathbf{H}$, one can construct a bipartite Tanner graph $\mathcal{G} = (V, C, E)$, where $V = \{1, 2, \ldots, N\} \triangleq [N]$ and $C = [M]$ are sets of variable nodes and check nodes. The edges, $E = \{(v, c) \in V \times C \,|\, H_{cv} \neq 0\}$, connect all parity checks to the variables involved in them. By convention, the boundary symbol $\partial$ denotes the neighborhood operator defined by $\partial v \triangleq \{c \,|\, (v, c) \in E\}$ and $\partial c \triangleq \{v \,|\, (v, c) \in E\}$.

### 5.2.1 Weighted Belief-Propagation Decoding

WBP is an iterative algorithm that passes messages in the form of log-likelihood ratios (LLRs) along the edges of $\mathcal{G}$ [NBB16]. In the variable-to-check-node step, the pre-update

message is

$$\lambda'^{(t)}_{v\to c} = w^{(t)}_v \ell_v + \sum_{c'\in\partial v\setminus c} w^{(t)}_{vc'} \hat{\lambda}^{(t-1)}_{c'\to v}, \tag{5.1}$$

where $w^{(t)}_{vc}$ is a weight assigned to the edge $(v, c)$ and $w^{(t)}_v$ is a weight assigned to the channel message

$$\ell_v \triangleq \log\left(\frac{\Pr(y_v \,|\, x_v = 0)}{\Pr(y_v \,|\, x_v = 1)}\right) = 4R\rho y_v. \tag{5.2}$$

In the check-to-variable-node step, the pre-update message is

$$\hat{\lambda}'^{(t)}_{c\to v} = 2\tanh^{-1}\left(\prod_{v'\in\partial c\setminus v} \tanh\left(\frac{\lambda^{(t)}_{v'\to c}}{2}\right)\right). \tag{5.3}$$

To mitigate oscillations and enhance convergence, we apply *damping* to complete the message updates [PFY07]. In particular, the final messages are a convex combination of the previous value and the pre-update value according to

$$\lambda^{(t)}_{v\to c} = \gamma\lambda^{(t-1)}_{v\to c} + (1-\gamma)\lambda'^{(t)}_{v\to c}, \tag{5.4}$$

$$\hat{\lambda}^{(t)}_{c\to v} = \gamma\hat{\lambda}^{(t-1)}_{c\to v} + (1-\gamma)\hat{\lambda}'^{(t)}_{c\to v}, \tag{5.5}$$

where $\gamma \in [0, 1]$ is the damping factor and $\lambda^{(0)}_{v\to c} = \hat{\lambda}^{(0)}_{c\to v} = 0$ for all $(v, c) \in E$.[1] Finally, output LLRs are computed as

$$m^{(t)}_v = w^{(t)}_v \ell_v + \sum_{c'\in\partial v} w^{(t)}_{vc'} \hat{\lambda}^{(t)}_{c'\to v}. \tag{5.6}$$

The sigmoid function $\sigma(x) = (1 + e^{-x})^{-1}$ maps $m^{(t)}_v$ to an estimate of the probability that $x_v = 1$ according to $o^{(t)}_v = 1 - \sigma(m^{(t)}_v)$. The corresponding hard decision is denoted by $\hat{o}^{(t)}_v$. For convenience, we also define $o_v \triangleq o^{(T)}_v$, $\hat{o}_v \triangleq \hat{o}^{(T)}_v$, and $m_v \triangleq m^{(T)}_v$, where $T$ is the total number of iterations. Setting all weights to 1 and $\gamma = 0$ recovers standard BP.

### 5.2.2 Random Redundant Decoding

The performance of BP can be improved by using redundant PC matrices where $M > N - K$. This can for example be realized by adding dual codewords as rows to a standard

[1] Damping is referred to as "relaxed BP" in [NML+18], where it is studied in the context of weighted min-sum decoding.

PC matrix [YCF02]. Another approach, referred to as random redundant decoding (RRD), is to use different PC matrices in each iteration [JN04, HC06]. This can be implemented efficiently by exploiting the code's automorphism group. In particular, let $\mathcal{S}_N$ be the symmetric group on $N$ elements, i.e., $\pi \in \mathcal{S}_N$ is a permutation on $[N]$. The automorphism group of a code $\mathcal{C}$ is defined as $\mathsf{Aut}(\mathcal{C}) \triangleq \{\pi \in \mathcal{S}_N \,|\, \boldsymbol{x}^\pi \in \mathcal{C}, \forall\, \boldsymbol{x} \in \mathcal{C}\}$, where $\boldsymbol{x}^\pi$ denotes a permuted vector, i.e., $x_i^\pi = x_{\pi(i)}$. RRD cascades $T_{\text{out}}$ BP decoders, each run with $T_{\text{in}}$ iterations, where the permuted output LLRs of the last decoder serve as soft input for the next decoder. A new, randomly sampled permutation $\pi_\tau \in \mathsf{Aut}(\mathcal{C})$ is used for each outer iteration $\tau \in [T_{\text{out}}]$. This effectively uses $T_{\text{out}}$ different PC matrices while fixing the Tanner graph for decoding. Similar to [NML+18], we consider weighted RRD by cascading several WBP decoders.

For RRD, the computation of output LLRs is typically modified by introducing a scale factor before the sum in (5.6) [JN04, Eq. (4)]. We use a similar, but slightly different, approach. In particular, the soft input to the $\tau$-th decoder is modified to be a convex combination of the initial channel LLRs and the output LLRs of the $(\tau-1)$-th decoder according to

$$\boldsymbol{\ell}^{(\tau)} = \left[\beta\boldsymbol{\ell} + (1-\beta)\boldsymbol{m}^{((\tau-1)T_{\text{in}})}\right]^{\pi_\tau}, \tag{5.7}$$

where $\beta \in [0,1]$ is referred to as the mixing factor and $\boldsymbol{m}^{(0)} = \boldsymbol{\ell}$. Here, all BP messages and weights are iteration-indexed consecutively for $t \in [T]$, where $T = T_{\text{out}}T_{\text{in}}$. An example of the resulting feed-forward computation graph for RRD with $T_{\text{out}} = 2$ and $T_{\text{in}} = 2$ is shown in Fig. 5.1.

## 5.3 Optimizing Weighted Belief-Propagation

It is well known that BP performs exact marginalization when the Tanner graph is a tree. However, good codes typically have loopy Tanner graphs with short cycles. To improve the BP performance, one can optimize the weights $w_{vc}^{(t)}$ and $w_v^{(t)}$ in all iterations [NBB16]. The damping and mixing factors $\gamma$, $\beta$ can also be optimized. In the following, $\boldsymbol{o} = f(\boldsymbol{y};\theta)$

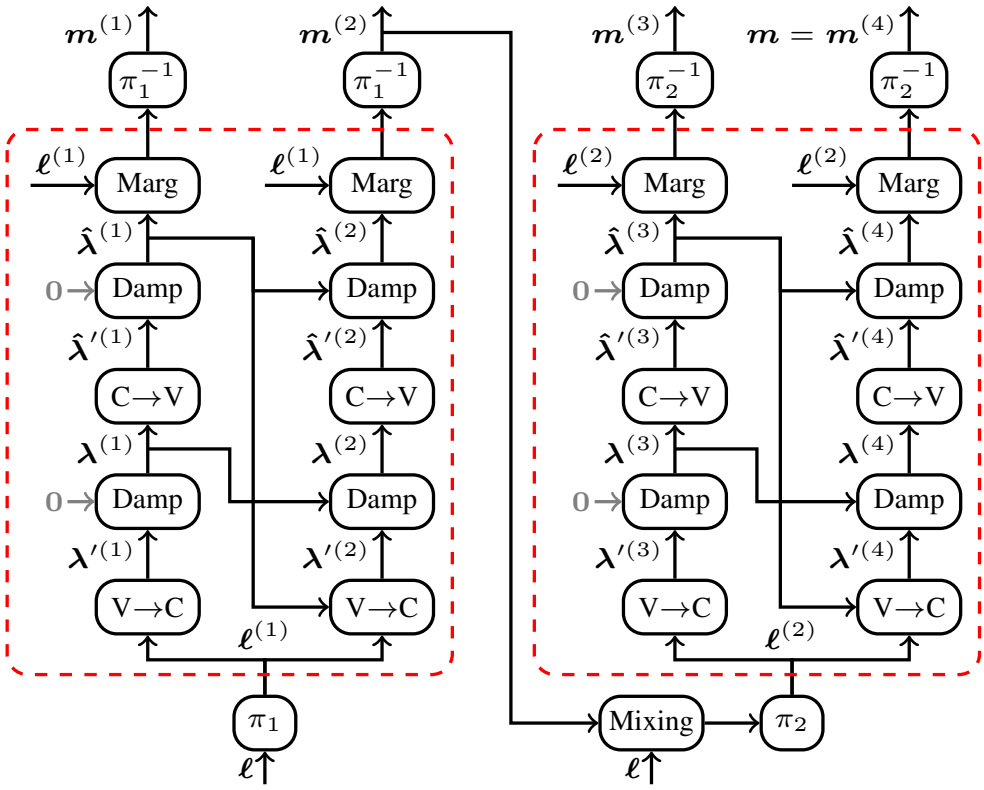



**Figure 5.1**: Feed-forward computation graph for RRD with $T_{\text{out}} = T_{\text{in}} = 2$.

denotes the entire WBP mapping, where $\theta$ comprises all parameters.

### 5.3.1 Deep Learning via Stochastic Gradient Descent

In [NBB16], the authors propose to optimize $\theta$ using stochastic gradient descent (SGD) (or a variant thereof) based on many codeword–observation pairs $(\boldsymbol{x}, \boldsymbol{y})$. In particular, the empirical loss $\mathcal{L}_A(\theta)$ for a finite set $A \subset \mathcal{C} \times \mathbb{R}^N$ of pairs is defined by

$$\mathcal{L}_A(\theta) \triangleq \frac{1}{|A|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in A} L\big(\boldsymbol{x}, f(\boldsymbol{y};\theta)\big), \tag{5.8}$$

where $L(\boldsymbol{a}, \hat{\boldsymbol{a}})$ is the loss associated with returning the output $\hat{\boldsymbol{a}}$ when $\boldsymbol{a}$ is correct. Mini-batch SGD then uses the parameter update $\theta_{i+1} = \theta_i - \alpha \nabla_\theta \mathcal{L}_{B_i}(\theta_i)$, where $\alpha$ is the learning rate and $B_i$ is the mini-batch used in the $i$-th step. Due to channel and decoder symmetry, transmission of the all-zero codeword $\boldsymbol{x} = \boldsymbol{0}$ can be assumed for the optimization [NBB16].

### 5.3.2 Optimization Loss Function

For supervised classification problems, one typically uses cross-entropy loss. However, since the number of classes (i.e., codewords) scales exponentially with the block length, it is more practical to assume that the overall loss is the average of *bit-wise* losses according to

$$L(\boldsymbol{x}, \boldsymbol{o}) = \frac{1}{N} \sum_{v=1}^{N} L_{\text{bit}}(x_v, o_v), \tag{5.9}$$

where $L_{\text{bit}}$ is a bit-wise loss function. For optimizing WBP, binary cross-entropy $L_{\text{bit}}(a, b) = -\log(b^a(1-b)^{1-a})$ is used in [NBB16, NML+18]. However, our experiments show that minimizing (5.9) using binary cross entropy does not necessarily minimize the BER. To see why this may be the case, note that the negative bit success rate (per codeword) can be written as

$$\frac{1}{N} \sum_{v=1}^{N} L_{\text{bit}}(x_v, \hat{o}_v) = -\frac{1}{N} \sum_{v=1}^{N} \hat{o}_v^{x_v} (1 - \hat{o}_v)^{1-x_v}, \tag{5.10}$$

**Table 5.1**: Comparison of bit-wise loss functions.

| name | $L_{\text{bit}}(a,b)$ | $L_{\text{bit}}(0,b)$ |
|---|---|---|
| binary cross-entropy | $-\log(b^a(1-b)^{1-a})$ | $-\log(1-b)$ |
| negative soft bit success | $-b^a(1-b)^{1-a}$ | $-(1-b)$ |
| soft bit error | $(1-b)^a b^{1-a}$ | $b$ |

where $L_{\text{bit}}(a,b) = -b^a(1-b)^{1-a}$. On the other hand, inserting binary cross entropy into (5.9) leads to

$$L(\boldsymbol{x},\boldsymbol{o}) = -\log\left(\prod_{v=1}^{N} o_v^{x_v}(1-o_v)^{1-x_v}\right)^{\frac{1}{N}}. \tag{5.11}$$

Besides the log, the main difference between (5.10) and (5.11) is that arithmetic mean is used instead of geometric mean.

We propose a new loss function, where $L_{\text{bit}}(a,b) = (1-b)^a b^{1-a}$ is used in (5.9). This can be regarded as a "soft" version of BER since $(1-b)^a b^{1-a}$ for binary variables corresponds to $a$ XOR $b$, i.e., $L_{\text{bit}}(a,b)$ indicates a bit error. We refer to the resulting loss function as soft-BER. Tab. 5.1 summarizes the different binary loss functions and their simplification for the all-zero codeword. Also note that maximizing negative soft bit success is equivalent to minimizing soft bit error.

### 5.3.3 Multi-Loss Optimization

The optimization behavior for WBP can be improved by using a multi-loss function [NBB16, NML+18] (see also [BCK18])

$$L(\boldsymbol{x},\{\boldsymbol{o}^{(t)}\}_{t=1}^{T}) \triangleq \frac{1}{\sum_{t=1}^{T}\eta^{T-t}}\sum_{t=1}^{T}\eta^{T-t}L(\boldsymbol{x},\boldsymbol{o}^{(t)}), \tag{5.12}$$

where $\eta \in [0,1]$ is a discount factor. Multi-loss optimization takes into account the output after every iteration which helps to increase the magnitude of gradients corresponding to earlier iterations. We found that, rather than using a fixed discount factor as in [NBB16, NML+18, BCK18], it is beneficial to decay $\eta$ during SGD, i.e., gradually moving from $\eta = 1$

(where the outputs of all BP iterations are considered with equal importance) towards $\eta = 0$ (where only the last BP iteration is taken into account).

### 5.3.4 Weight Sharing

Excluding the damping/mixing factors, the total number of weights is $T(|E| + N)$ and we refer to this case as the fully-weighted (FW) decoder. In order to reduce the optimization complexity, one can share the weights, e.g., as follows:

- Temporal weight sharing (across decoding iterations), i.e.,

$$w_{vc}^{(t)} \equiv w_{vc}, \qquad w_{v}^{(t)} \equiv w_{v}, \qquad \forall\, t \in [T],$$

  is referred to as RNN-FW, due to the similarity with recurrent neural networks (RNNs) [NBB16].

- Spatial weight sharing (across edges), i.e.,

$$w_{vc}^{(t)} \equiv w_{\text{msg}}^{(t)}, \qquad w_{v}^{(t)} \equiv w_{\text{ch}}^{(t)}, \qquad \forall\, (v,c) \in E,$$

  gives the simple-scaling (SS) model with two weights per iteration: one message and one channel weight.

- Temporal *and* spatial weight sharing gives two parameters in total. This is referred to as RNN-SS.

It is shown in [NBB16, NML$^+$18] that the RNN-FW structure gives similar gains as the FW decoder, i.e., there is little improvement when making parameters iteration-dependent. We further show that the RNN-SS structure incurs little to no performance penalty in many cases.

### 5.3.5 Training SNR and Parameter Adapter Network

In general, the optimal WBP parameters may be different for different SNRs [NML$^+$18]. On the other hand, optimizing WBP separately for each SNR and storing the resulting

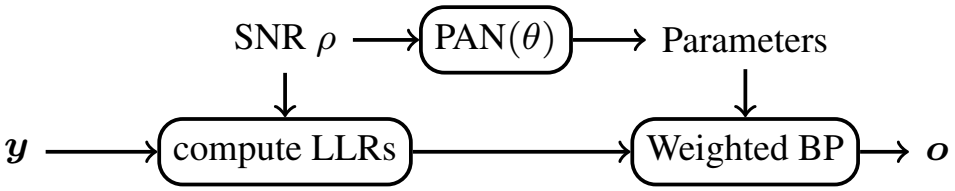



**Figure 5.2**: Block diagram illustrating the parameter adapter network (PAN).

weights is impractical if the set of possible SNRs is large or infinite. One general approach is to instead optimize assuming a range of different training SNRs [NBB16, NML$^+$18]. This leads to parameters that achieve a compromise between different channel conditions.

We propose a different approach where a PAN is used to learn the relation between the SNR $\rho$ and the corresponding optimal parameters.[2] Once trained, the PAN can be used to adaptively choose the best parameters for WBP corresponding to the channel conditions. The basic idea is illustrated in Fig. 5.2. In general, one can choose any structure to construct the PAN, e.g., a vanilla neural network. It is also possible to make only a subset of parameters SNR-adaptive.

In this paper, we use several shallow neural networks with one hidden layer of dimension 20 and output dimension 1 to model the SNR-dependency for each WBP parameter separately. ReLU activations are used for the hidden layer. The output layer uses sigmoid activations to ensure that the parameters satisfy their domain constraints, e.g., the damping factor is in the range $[0, 1]$. For regular weights, we further scale the sigmoid outputs by 10 to increase the range to $[0, 10]$. As an example, for WBP with the RNN-SS structure including damping, there are three parameters $w_{\text{msg}}$, $w_{\text{ch}}$, and $\gamma$. Thus, the PAN describes an SNR-parameterization according to $\text{PAN}(\rho) = [w_{\text{msg}}(\rho),\, w_{\text{ch}}(\rho),\, \gamma(\rho)] \in [0, 10]^2 \times [0, 1]$.

[2]We assume perfect knowledge of the SNR. This knowledge is also required implicitly to compute channel LLRs. In practice, SNR is typically estimated and the SNR estimate can then be used as the input to the PAN.

## 5.4 Numerical Results

The various decoding architectures in this paper are implemented in the PyTorch framework and optimized using the RMSprop optimizer which is a variant of mini-batch SGD. Each mini-batch contains 100 observation pairs and the SNR for each pair is chosen from 10 equidistant points in the interval $[1\,\mathrm{dB}, 8\,\mathrm{dB}]$ such that exactly 10 pairs have the same SNR. The discount decay for the multi-loss optimization is implemented by starting with an initial discount factor $\eta = 1$ and multiplying $\eta$ by 0.5 after every 5000th SGD step. The same schedule is used to decay the learning rate, starting from $\alpha = 10^{-3}$ and using a decay rate of 0.8 instead of 0.5. To avoid numerical issues, a gradient clipping threshold of 0.1 is applied and the absolute values of the LLRs $\lambda_{v\to c}^{(t)}$ are clipped into the range $[-\log(\tanh(L_{\max}/2)), L_{\max}]$ with $L_{\max} = 15$.

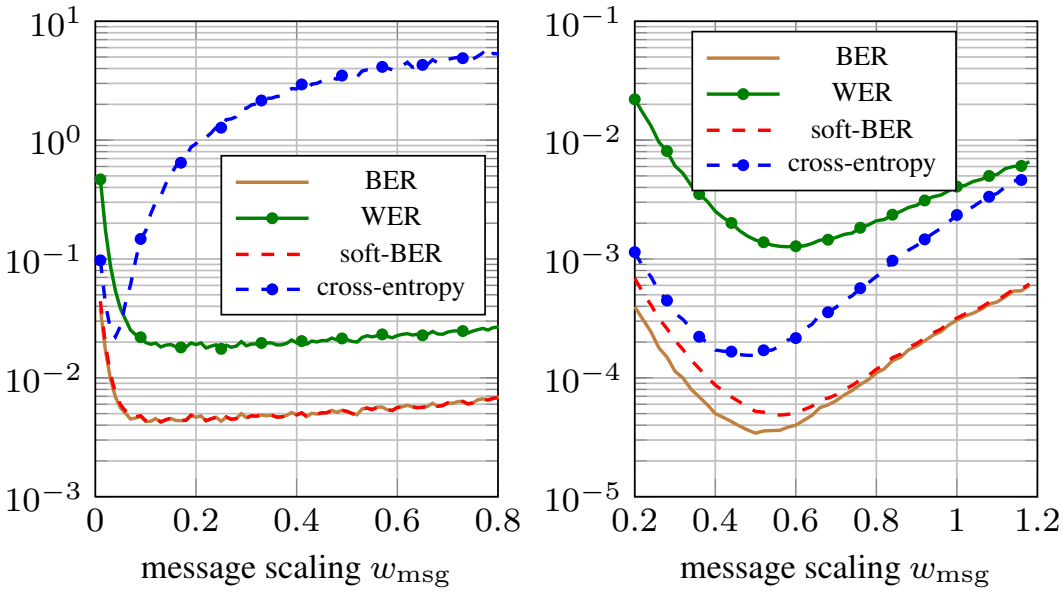
The following Reed–Muller (RM) and Bose–Chaudhuri–Hocquenghem (BCH) codes are considered:

- RM$(32, 16)$ with standard PC matrix $\mathbf{H}_{\mathrm{std}}$ (size $16\times32$) and overcomplete PC matrix $\mathbf{H}_{\mathrm{oc}}$ ($620 \times 32$) whose rows are all minimum-weight dual codewords, see [SHP18]
- BCH$(63, 36)$ with cycle-reduced PC matrix $\mathbf{H}_{\mathrm{cr}}$ ($27\times63$) and right-regular PC matrix $\mathbf{H}_{\mathrm{rr}}$ ($27 \times 63$), see [HSG$^+$19]
- BCH$(127, 64)$ with cycle-reduced PC matrix $\mathbf{H}_{\mathrm{cr}}$ ($63 \times 127$), see [HSG$^+$19]

Ordered statistics decoding (OSD) is used as a benchmark whose performance is close to maximum-likelihood [FS95].

### 5.4.1 Comparison of Loss Functions

We start by considering two RNN-SS structures with fixed $w_{\mathrm{ch}} = 1$ and $\gamma = 0$ (i.e., no damping): (a) RM$(32, 16)$ with $\mathbf{H}_{\mathrm{oc}}$ and (b) BCH$(63, 36)$ with $\mathbf{H}_{\mathrm{cr}}$. The different loss functions for $T = 3$ are plotted in Fig. 5.3 as a function of $w_{\mathrm{msg}}$, which is the only trainable parameter. For the RM code, cross-entropy has a sharp minimum at $w_{\mathrm{msg}} \approx 0.05$, whereas




(a) RM$(32, 16)$, overcomplete $\mathbf{H}_{\text{oc}}$ (b) BCH$(63, 36)$, cycle-reduced $\mathbf{H}_{\text{cr}}$

**Figure 5.3**: Comparison of loss functions for RNN-SS with $w_{\text{ch}} = 1$, $\gamma = 0$, and $T = 3$. The SNR is $E_{\text{b}}/N_0 = 3\,\text{dB}$ in (a) and $E_{\text{b}}/N_0 = 7\,\text{dB}$ in (b).

soft-BER overlaps with BER and has a flat minimum at $w_{\text{msg}} \approx 0.15$. For the BCH code, the minima for cross-entropy, soft-BER, and BER all occur close to each other, but at slightly different locations.

In order to explain the distinct behavior of cross-entropy in Fig. 5.3(a), note that if a bit is decoded incorrectly, binary cross-entropy gives a penalty close to the magnitude of the output LLR $|m_v|$. This is problematic in cases where the decoder is wrong, but very sure about its decision. Indeed, this behavior is characteristic for BP with highly redundant PC matrices and such failure cases tend to dominate the average loss. This effect is even more pronounced for large $T$ since the average LLR magnitude tends to grow with the iteration number.

The results in Fig. 5.3 show that, in general, neither cross-entropy nor soft-BER are guaranteed to minimize BER. All scenarios in this paper were optimized using both functions. We found that they give comparable results, with the exception of highly redundant PC matrices where soft-BER is preferable.

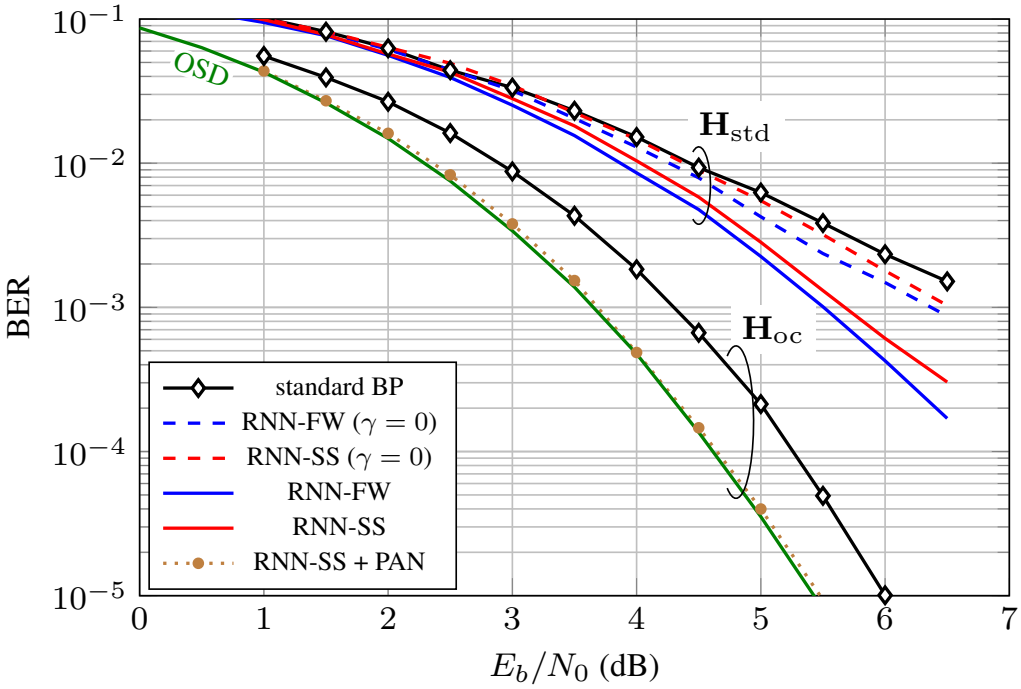



**Figure 5.4**: Results for RM$(32, 16)$ with $\mathbf{H}_{\text{oc}}$ $(T = 5)$ and $\mathbf{H}_{\text{std}}$ $(T = 20)$.

### 5.4.2 Reed–Muller Codes

Results for RM$(32, 16)$ assuming both RNN-FW and RNN-SS structures are shown in Fig. 5.4. For $\mathbf{H}_{\text{std}}$ with $T = 20$ iterations, simple scaling results in a performance loss of up to 0.3 dB. Damping gives considerable performance improvements in both cases, at the expense of additional computational complexity and storage requirements. For $\mathbf{H}_{\text{oc}}$ with $T = 5$, the RNN-SS structure is sufficient to achieve close-to-optimal performance and the overlapping results for RNN-FW are omitted. For this case, we note that the optimization with soft-BER gives lower BER than for CE, as expected from the discussion in the previous subsection.

### 5.4.3 BCH Codes

For the BCH codes, the parameters are chosen to facilitate a direct comparison with [NML+18]. In particular, we fix $T = 5$ and $\gamma = 0$, i.e., no damping is used. Results

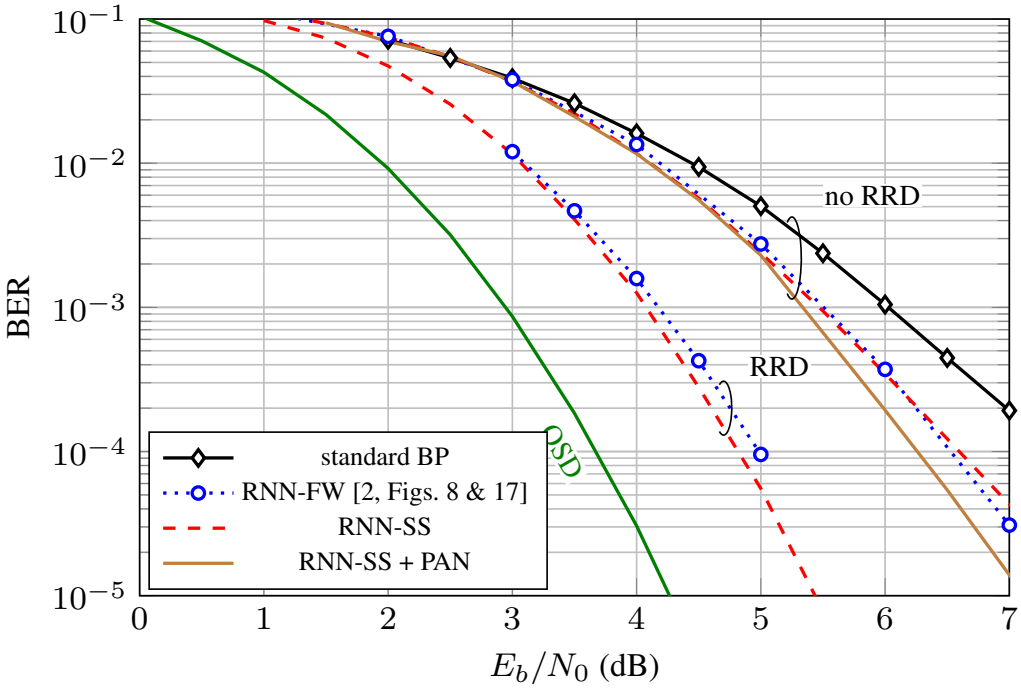



**Figure 5.5**: Results for BCH$(63, 36)$, with cycle-reduced $\mathbf{H}_{\text{cr}}$

for BCH$(63, 36)$ with $\mathbf{H}_{\text{cr}}$ are shown in Fig. 5.5, where we compare with the best results in [NML$^+$18, Fig. 8] for the same parameters. The RNN-SS with two trainable parameters achieves similar gains as the RNN-FW in [NML$^+$18] for BERs $> 10^{-4}$. For lower BERs, the performance starts to deviate. The situation can be improved by making the parameters SNR-adaptive using the proposed PAN approach. In this case, the performance improves markedly for high SNRs. This is due to the fact that training over a range of SNRs without the PAN tends to focus almost exclusively on low-SNR/high-BER regions. Similar observations can be made for the same code with $\mathbf{H}_{\text{rr}}$, as shown in Fig. 5.6. In this case, RNN-FW performs slightly better than RNN-SS for some SNRs, i.e., two parameters are not sufficient to obtain the full gain. Finally, results for BCH$(127, 64)$ with $\mathbf{H}_{\text{cr}}$ are shown in Fig. 5.7. We caution the reader that these results are not directly comparable because our standard BP performs better than what is shown in [NML$^+$18, Fig. 12]. This is likely due to different cycle-reduced PC matrices. However, we were not able to improve upon the shown RNN-SS results using RNN-FW, which indicates that the simple-scaling

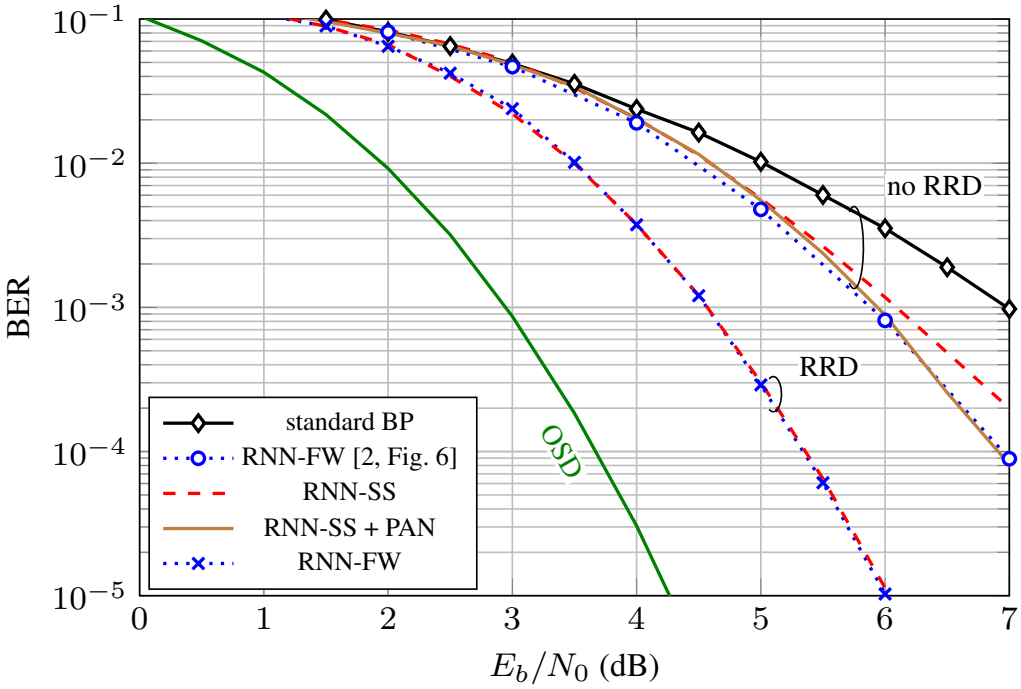



**Figure 5.6**: Results for BCH$(63, 36)$, with right-regular $\mathbf{H}_{\text{rr}}$

approach is also sufficient in this case.

We also consider weighted RRD for BCH$(63, 36)$ and BCH$(127, 64)$ where $T_{\text{in}} = 2$, $T_{\text{out}} = 30$, and the mixing factor is treated as an additional optimization parameter. Results for $\mathbf{H}_{\text{cr}}$ are shown in Fig. 5.5 and we compare to the corresponding results in [NML$^+$18, Fig. 12] labeled as "mRRD-RNN(1)". We obtain slightly better performance using RNN-SS even without a PAN. This can be attributed to the improved training methodology, particularly the discount decay for the multi-loss optimization. For $\mathbf{H}_{\text{rr}}$, no RRD results are available in [NML$^+$18] and we compare to our own results. Both RNN-FW and RNN-SS give virtually the same performance as shown in Fig. 5.6. Fig. 5.7 shows that the PAN also gives some extra performance gain for RRD decoding. Additional simulation results for larger $T$ can be found in [LHP19].

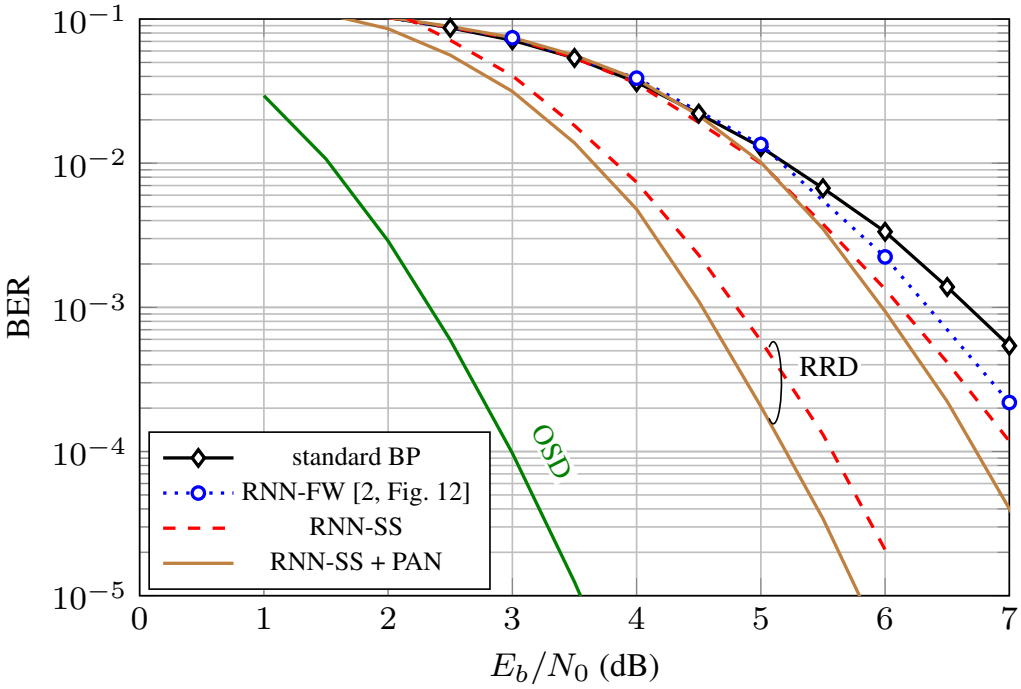



**Figure 5.7**: Results for BCH$(127, 64)$, with cycle-reduced $\mathbf{H}_{\text{cr}}$

## 5.5 Discussion and Conclusion

In this chapter, we have considered WBP decoding of short Reed–Muller and BCH codes. Our experiments support the observations in [NBB16, NML+18] that optimizing WBP can provide meaningful gains. In addition, we have shown that simple-scaling models with fewer parameters are often sufficient to achieve gains similar to the full parameterization. This can lead to a considerably simpler optimization procedure and greatly reduce complexity, e.g., in terms of storage requirements. In general, the performance loss incurred by simple scaling depends on the employed PC matrix. Small penalties were observed for matrices with highly irregular degree distributions (e.g., $\mathbf{H}_{\text{std}}$ for RM(32,16) or $\mathbf{H}_{\text{rr}}$ for BCH(63, 36)), whereas the loss appears to be negligible if the degree distribution is regular ($\mathbf{H}_{\text{oc}}$ for RM(32,16)) or RRD is employed.

It was also shown that choosing a suitable loss function for the optimization is scenario-dependent. For highly redundant PC matrices, it was found that binary cross-entropy

penalizes too hard on bit errors where the decoder is very sure about its decision. In such cases, optimizing with the proposed soft-BER loss leads to better performance. Lastly, we built on the observation in [NML+18] that the optimal WBP parameters are SNR-dependent and proposed a simple solution based on parameter adapter networks. This approach allows us to learn optimal parameters for multiple SNRs in a single training process without trading off performance between channel conditions.

# Chapter 6

# Conclusions

This dissertation primarily addresses the application and improvement of belief propagation algorithms for various communication systems. Chapter 1 examines the performance of existing algorithms for several classes of ill-conditioned measurement matrices in a compressed sensing problem. Simulation results of the RBP-SVD algorithm show that, after an SVD preprocess, the reformulated compressed sensing problems have better convergence behavior over a larger class of measurement matrices. If the measurement matrix can be written as the product of multiple sparse matrices, then the RBP-SEQ algorithm leverages the sparsity of the underlying layered factor graph and reduces the computational complexity.

Chapter 2 discusses about how to apply belief propagation for fiber-optic systems. There are two key challenges: one is that the signal in optical communications are complex-valued, and the other is that the noise is colored during the back-propagation. These elements introduce correlation between different symbols and correlation between the real and imaginary parts of same symbol. A novel algorithm, covariance back-propagation, is derived by modeling the signal as multivariate complex Gaussian distribution and back solving the nonlinear Schrödinger equation using SSFM and first-order approximation. Compared with the single-point estimate given by DBP and the particle-filter approximation given by SDBP, CBP provides comprehensive correlation information between all symbols. When the Gaussian assumption holds, this yields better symbol detection performance.

The remaining chapters focus on applying belief propagation for channel coding. Chapter 3 investigates rich algebraic structure of the Reed–Muller codes. We build the connection between belief propagation decoding over redundant factor graph and the existing RPA_RM algorithm. A novel algorithm for decoding high-rate Reed–Muller codes is proposed by

decoding extended Hamming subcodes associated with specific puncturing patterns. Additionally, we improve the RPA_RM decoding algorithm by collapsing the recursion into one stage. This reduces the computational complexity because it avoids decoding the same subcode multiple times. In Chapter 4, we combine belief propagation decoding and deep learning. The deep unfolding idea is used to unroll an iterative algorithm up to a fixed number of iterations. Our focus is on simple-scaling models that share the same weights across certain edges to reduce the storage and computational burden. The main contribution is to show that simple scaling with few parameters often achieves the same gain as the full parameterization. Moreover, several training improvements for WBP are proposed. For example, it is shown that minimizing average binary cross-entropy is suboptimal in general in terms of bit error rate and a new "soft-BER" loss is proposed which can lead to better performance. We also investigate parameter adapter networks that learn the relation between the signal-to-noise ratio and the WBP parameters.

In summary, recent progress in machine learning and off-the-shelf learning packages have made it tractable to add many parameters to existing communication algorithms and optimize. One key issue with of deep learning models is their lack of interpretability. On the other hand, decades of research in system design and signal processing provides solid theoretical background to solve problems in communication systems. This allows one to incorporate domain knowledge into deep learning models, which is an emerging research area. In this dissertation, instead of building an end-to-end deep neural network, we separate larger problems into several meaningful independent tasks and train neural network to solve subtasks, so that the output of neural networks can be interpreted. Another promising future direction is to use reinforcement learning and model the problem as an interaction between an agent and its environment. By design, the domain knowledge can be embedded in the reward, state space, action space, and state transitions given the action. The ultimate goal is to find an optimal policy that achieves maximum expected reward. This viewpoint is natural when one thinks the decoding process is data-specific and the number of iterations to converge is unknown. I believe this can it could bring vitality

to the area of communication research and mesh well with the new era of interpretable machine learning.

# Appendix A

# Proof of Properties of Multivariate Complex Gaussian Distribution

## A.1 Linear Transformation Property

**Lemma 9.** *If* $\mathbf{Z} \in \mathbb{C}^n$, $\mathbf{A} \in \mathbb{C}^{m\times n}$, $\mathbf{b} \in \mathbb{C}^m$ *and* $\mathbf{Z} \sim \mathcal{CN}(\boldsymbol{\mu}, \boldsymbol{\Gamma}, \mathbf{C})$, *then the linear transformation* $\mathbf{Z}' = \mathbf{AZ} + \mathbf{b}$ *has the distribution* $\mathcal{CN}(\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{A}\boldsymbol{\Gamma}\mathbf{A}^{\mathsf{H}}, \mathbf{A}\mathbf{C}\mathbf{A}^{\mathsf{T}})$.

*Proof.* Denote $\boldsymbol{\mu}'$, $\boldsymbol{\Gamma}'$, $\mathbf{C}'$ as the mean, covariance matrix, and pseudo-covariance matrix of $\mathbf{Z}'$, respectively. Using the linearity of expectation,

$$\boldsymbol{\mu}' = \mathbb{E}[\mathbf{Z}'] = \mathbf{A}\mathbb{E}[\mathbf{Z}] + \mathbf{b} = \mathbf{A}\boldsymbol{\mu} + \mathbf{b} \tag{A.1}$$

$$\begin{aligned}\boldsymbol{\Gamma}' &= \mathbb{E}\left[\left(\mathbf{Z}' - \mathbb{E}[\mathbf{Z}']\right)\left(\mathbf{Z}' - \mathbb{E}[\mathbf{Z}']\right)^{\mathsf{H}}\right] \\ &= \mathbb{E}\left[\left(\mathbf{AZ} - \mathbf{A}\boldsymbol{\mu}\right)\left(\mathbf{AZ} - \mathbf{A}\boldsymbol{\mu}\right)^{\mathsf{H}}\right] \\ &= \mathbf{A}\mathbb{E}\left[\left(\mathbf{Z} - \boldsymbol{\mu}\right)\left(\mathbf{Z} - \boldsymbol{\mu}\right)^{\mathsf{H}}\right]\mathbf{A}^{\mathsf{H}} \\ &= \mathbf{A}\boldsymbol{\Gamma}\mathbf{A}^{\mathsf{H}}\end{aligned} \tag{A.2}$$

$$\begin{aligned}\mathbf{C}' &= \mathbb{E}\left[\left(\mathbf{Z}' - \mathbb{E}[\mathbf{Z}']\right)\left(\mathbf{Z}' - \mathbb{E}[\mathbf{Z}']\right)^{\mathsf{T}}\right] \\ &= \mathbb{E}\left[\left(\mathbf{AZ} - \mathbf{A}\boldsymbol{\mu}\right)\left(\mathbf{AZ} - \mathbf{A}\boldsymbol{\mu}\right)^{\mathsf{T}}\right] \\ &= \mathbf{A}\mathbb{E}\left[\left(\mathbf{Z} - \boldsymbol{\mu}\right)\left(\mathbf{Z} - \boldsymbol{\mu}\right)^{\mathsf{T}}\right]\mathbf{A}^{\mathsf{T}} \\ &= \mathbf{A}\mathbf{C}\mathbf{A}^{\mathsf{T}}.\end{aligned} \tag{A.3}$$

Therefore, after linear transformation, the random vector $\mathbf{Z}' = \mathbf{AZ} + \mathbf{b}$ has the distribution

$$\mathcal{CN}(\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{A}\boldsymbol{\Gamma}\mathbf{A}^{\mathsf{H}}, \mathbf{A}\mathbf{C}\mathbf{A}^{\mathsf{T}}).$$

□

## A.2 Independent Sum Property

**Lemma 10.** *If* $\mathbf{Z}_1 \sim \mathcal{CN}(\boldsymbol{\mu}_1, \boldsymbol{\Gamma}_1, \mathbf{C}_1)$ *and* $\mathbf{Z}_2 \sim \mathcal{CN}(\boldsymbol{\mu}_2, \boldsymbol{\Gamma}_2, \mathbf{C}_2)$ *are independent, then* $\mathbf{Z}' = \mathbf{Z}_1 + \mathbf{Z}_2$ *has the distribution* $\mathcal{CN}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2, \boldsymbol{\Gamma}_1 + \boldsymbol{\Gamma}_2, \mathbf{C}_1 + \mathbf{C}_2)$.

*Proof.* Denote $\boldsymbol{\mu}'$, $\boldsymbol{\Gamma}'$, $\mathbf{C}'$ as the mean, covariance matrix, and pseudo-covariance matrix of $\mathbf{Z}'$, respectively. Using the linearity of expectation and the fact that $\mathbf{Z}_1$ and $\mathbf{Z}_2$ are independent, we have

$$\boldsymbol{\mu}' = \mathbb{E}[\mathbf{Z}'] = \mathbb{E}[\mathbf{Z}_1] + \mathbb{E}[\mathbf{Z}_2] = \boldsymbol{\mu}_1 + \boldsymbol{\mu}_2 \tag{A.4}$$

$$\begin{aligned}
\boldsymbol{\Gamma}' &= \mathbb{E}\left[\left(\mathbf{Z}' - \mathbb{E}[\mathbf{Z}']\right)\left(\mathbf{Z}' - \mathbb{E}[\mathbf{Z}']\right)^{\mathsf{H}}\right] \\
&= \mathbb{E}\left[\left((\mathbf{Z}_1 - \boldsymbol{\mu}_1) + (\mathbf{Z}_2 - \boldsymbol{\mu}_2)\right)\left((\mathbf{Z}_1 - \boldsymbol{\mu}_1) + (\mathbf{Z}_2 - \boldsymbol{\mu}_2)\right)^{\mathsf{H}}\right] \\
&= \mathbb{E}\left[\left(\mathbf{Z}_1 - \boldsymbol{\mu}_1\right)\left(\mathbf{Z}_1 - \boldsymbol{\mu}_1\right)^{\mathsf{H}}\right] + \mathbb{E}\left[\left(\mathbf{Z}_2 - \boldsymbol{\mu}_2\right)\left(\mathbf{Z}_2 - \boldsymbol{\mu}_2\right)^{\mathsf{H}}\right] \\
&= \boldsymbol{\Gamma}_1 + \boldsymbol{\Gamma}_2
\end{aligned} \tag{A.5}$$

$$\begin{aligned}
\boldsymbol{\Gamma}' &= \mathbb{E}\left[\left(\mathbf{Z}' - \mathbb{E}[\mathbf{Z}']\right)\left(\mathbf{Z}' - \mathbb{E}[\mathbf{Z}']\right)^{\mathsf{T}}\right] \\
&= \mathbb{E}\left[\left((\mathbf{Z}_1 - \boldsymbol{\mu}_1) + (\mathbf{Z}_2 - \boldsymbol{\mu}_2)\right)\left((\mathbf{Z}_1 - \boldsymbol{\mu}_1) + (\mathbf{Z}_2 - \boldsymbol{\mu}_2)\right)^{\mathsf{T}}\right] \\
&= \mathbb{E}\left[\left(\mathbf{Z}_1 - \boldsymbol{\mu}_1\right)\left(\mathbf{Z}_1 - \boldsymbol{\mu}_1\right)^{\mathsf{T}}\right] + \mathbb{E}\left[\left(\mathbf{Z}_2 - \boldsymbol{\mu}_2\right)\left(\mathbf{Z}_2 - \boldsymbol{\mu}_2\right)^{\mathsf{T}}\right] \\
&= \mathbf{C}_1 + \mathbf{C}_2.
\end{aligned} \tag{A.6}$$

Therefore, after linear transformation, the random vector $\mathbf{Z}' = \mathbf{A}\mathbf{Z} + \mathbf{b}$ has the distribution

$$\mathcal{CN}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2, \boldsymbol{\Gamma}_1 + \boldsymbol{\Gamma}_2, \mathbf{C}_1 + \mathbf{C}_2).$$

□

## A.3 Block LDU Decomposition Property

Using the block LDU decomposition [Zha05]

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{A}_{21}\mathbf{A}_{11}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{A}_{11}^{-1}\mathbf{A}_{12} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{A.7}$$

and the fact that $\mathbf{\Gamma}$ is Hermitian, $\mathbf{C}$ is symmetric, it follows that

$$\begin{bmatrix} \mathbf{\Gamma} & \mathbf{C} \\ \overline{\mathbf{C}} & \overline{\mathbf{\Gamma}} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \overline{\mathbf{C}}\mathbf{\Gamma}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{\Gamma} & \mathbf{0} \\ \mathbf{0} & \overline{\mathbf{\Gamma}} - \overline{\mathbf{C}}\mathbf{\Gamma}^{-1}\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{\Gamma}^{-1}\mathbf{C} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{C}^{\mathsf{H}}\mathbf{\Gamma}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{\Gamma} & \mathbf{0} \\ \mathbf{0} & \overline{\mathbf{\Gamma}} - \overline{\mathbf{C}}\mathbf{\Gamma}^{-1}\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \left[\mathbf{C}\mathbf{\Gamma}^{-1}\right]^{\mathsf{H}} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

Thus, the definition of matrices $\mathbf{R}$ and $\mathbf{P}$ are obtained by matching terms.

## A.4 Block Inverse Property

According to [LS02], when $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ are both invertible, the inverse of $2 \times 2$ block matrix has two equivalent forms:

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}^{-1} = \begin{bmatrix} \tilde{\mathbf{A}}_1^{-1} & -\tilde{\mathbf{A}}_1^{-1}\mathbf{A}_{12}\mathbf{A}_{22}^{-1} \\ -\mathbf{A}_{22}^{-1}\mathbf{A}_{21}\tilde{\mathbf{A}}_1^{-1} & \mathbf{A}_{22}^{-1} + \mathbf{A}_{22}^{-1}\mathbf{A}_{21}\tilde{\mathbf{A}}_1^{-1}\mathbf{A}_{12}\mathbf{A}_{22}^{-1} \end{bmatrix} \tag{A.8}$$
$$= \begin{bmatrix} \mathbf{A}_{11}^{-1} + \mathbf{A}_{11}^{-1}\mathbf{A}_{12}\tilde{\mathbf{A}}_2^{-1}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & -\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\tilde{\mathbf{A}}_2^{-1} \\ -\tilde{\mathbf{A}}_2^{-1}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & \tilde{\mathbf{A}}_2^{-1} \end{bmatrix} \tag{A.9}$$

where

$$\tilde{\mathbf{A}}_1 = \mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21} \tag{A.10}$$

$$\tilde{\mathbf{A}}_2 = \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12} \tag{A.11}$$

Note that $\mathbf{C}$ is symmetric so that

$$\overline{\mathbf{P}} = \mathbf{\Gamma} - \mathbf{C}^{\mathsf{T}}\overline{\mathbf{\Gamma}}^{-1}\overline{\mathbf{C}} = \mathbf{\Gamma} - \mathbf{C}\mathbf{\Gamma}^{-1}\overline{\mathbf{C}}$$
$$\mathbf{P} = \overline{\mathbf{\Gamma}} - \mathbf{C}^{\mathsf{H}}\mathbf{\Gamma}^{-1}\mathbf{C} = \overline{\mathbf{\Gamma}} - \overline{\mathbf{C}}\mathbf{\Gamma}^{-1}\mathbf{C}$$

Thus, we write

$$\begin{bmatrix} \mathbf{\Gamma} & \mathbf{C} \\ \overline{\mathbf{C}} & \overline{\mathbf{\Gamma}} \end{bmatrix}^{-1} = \begin{bmatrix} \overline{\mathbf{P}}^{-1} & -\overline{\mathbf{P}}\mathbf{C}\overline{\mathbf{\Gamma}}^{-1} \\ -\overline{\mathbf{\Gamma}}^{-1}\overline{\mathbf{C}}\,\overline{\mathbf{P}}^{-1} & \overline{\mathbf{\Gamma}}^{-1} + \overline{\mathbf{\Gamma}}^{-1}\overline{\mathbf{C}}\,\overline{\mathbf{P}}^{-1}\mathbf{C}\overline{\mathbf{\Gamma}}^{-1} \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{\Gamma}^{-1} + \mathbf{\Gamma}^{-1}\mathbf{C}\mathbf{P}^{-1}\overline{\mathbf{C}}\mathbf{\Gamma}^{-1} & -\mathbf{\Gamma}^{-1}\mathbf{C}\mathbf{P}^{-1} \\ -\mathbf{P}^{-1}\overline{\mathbf{C}}\mathbf{\Gamma}^{-1} & \mathbf{P}^{-1} \end{bmatrix}$$

Since the four blocks have the same size, by picking terms from the two equivalent inverse forms, we have

$$\begin{bmatrix} \mathbf{\Gamma} & \mathbf{C} \\ \overline{\mathbf{C}} & \overline{\mathbf{\Gamma}} \end{bmatrix}^{-1} = \begin{bmatrix} \overline{\mathbf{P}}^{-1} & -\mathbf{\Gamma}^{-1}\mathbf{C}\mathbf{P}^{-1} \\ -\overline{\mathbf{\Gamma}}^{-1}\overline{\mathbf{C}}\,\overline{\mathbf{P}}^{-1} & \mathbf{P}^{-1} \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{P}}^{-1} & -\mathbf{R}^{\mathsf{H}}\mathbf{P}^{-1} \\ -\mathbf{R}^{\mathsf{T}}\,\overline{\mathbf{P}}^{-1} & \mathbf{P}^{-1} \end{bmatrix}$$

## A.5 Proof of Theorem 2

*Proof.* Since $\mathbf{Z}$ is partitioned such that

$$\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_a \\ \mathbf{Z}_b \end{bmatrix} = \begin{bmatrix} \mathbf{X}_a \\ \mathbf{X}_b \end{bmatrix} + \mathbb{i} \begin{bmatrix} \mathbf{Y}_a \\ \mathbf{Y}_b \end{bmatrix}, \tag{A.12}$$

with

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}, \quad \mathbf{\Gamma} = \begin{bmatrix} \mathbf{\Gamma}_a & \mathbf{\Gamma}_{ab} \\ \mathbf{\Gamma}_{ab}^{\mathsf{H}} & \mathbf{\Gamma}_b \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} \mathbf{C}_a & \mathbf{C}_{ab} \\ \mathbf{C}_{ab}^{\mathsf{T}} & \mathbf{C}_b \end{bmatrix}$$

To compute the conditional distribution $p\left(\mathbf{Z}_a \,|\, \mathbf{Z}_b = \mathbf{z}_b\right)$, it is equivalent to compute the distribution $p\left(\mathbf{X}_a, \mathbf{Y}_a \,|\, \mathbf{X}_b = \mathbf{x}_b, \mathbf{Y}_b = \mathbf{y}_b\right)$.

Similar to multivariate Gaussian distribution, the conditional distribution of a complex Gaussian distribution is still a complex Gaussian distribution. The only difference is that for complex Gaussian distribution one needs three parameters to fully describe the distribution, while for real Gaussian distribution only two parameters are required.

For the conditional mean, we have

$$\begin{aligned}
&\mathbb{E}\left[\mathbf{Z}_a \,|\, \mathbf{Z}_b = \mathbf{z}_b\right] = \mathbb{E}\left[\mathbf{X}_a \,|\, \mathbf{X}_b = \mathbf{x}_b, \mathbf{Y}_b = \mathbf{y}_b\right] + \mathbb{i}\,\mathbb{E}\left[\mathbf{Y}_a \,|\, \mathbf{X}_b = \mathbf{x}_b, \mathbf{Y}_b = \mathbf{y}_b\right] \\
&\quad = \left\{ \boldsymbol{\mu}_{\mathbf{X}_a} + \mathbf{V}_{\mathbf{X}_a,(\mathbf{X}_b,\mathbf{Y}_b)} \mathbf{V}^{-1}_{(\mathbf{X}_b,\mathbf{Y}_b),(\mathbf{X}_b,\mathbf{Y}_b)} \left( \begin{bmatrix} \mathbf{x}_b \\ \mathbf{y}_b \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_{\mathbf{X}_b} \\ \boldsymbol{\mu}_{\mathbf{Y}_b} \end{bmatrix} \right) \right\} \\
&\qquad + \mathbb{i} \left\{ \boldsymbol{\mu}_{\mathbf{Y}_a} + \mathbf{V}_{\mathbf{Y}_a,(\mathbf{X}_b,\mathbf{Y}_b)} \mathbf{V}^{-1}_{(\mathbf{X}_b,\mathbf{Y}_b),(\mathbf{X}_b,\mathbf{Y}_b)} \left( \begin{bmatrix} \mathbf{x}_b \\ \mathbf{y}_b \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_{\mathbf{X}_b} \\ \boldsymbol{\mu}_{\mathbf{Y}_b} \end{bmatrix} \right) \right\} \\
&\quad = \boldsymbol{\mu}_{\mathbf{Z}_a} + \begin{bmatrix} \mathbf{V}_{\mathbf{X}_a,\mathbf{X}_b} + \mathbb{i}\mathbf{V}_{\mathbf{Y}_a,\mathbf{X}_b} & \mathbf{V}_{\mathbf{X}_a,\mathbf{Y}_b} + \mathbb{i}\mathbf{V}_{\mathbf{Y}_a,\mathbf{Y}_b} \end{bmatrix} \\
&\qquad \mathsf{Cov} \left( \begin{bmatrix} \mathsf{Re}\left(\mathbf{Z}_b\right) \\ \mathsf{Im}\left(\mathbf{Z}_b\right) \end{bmatrix} \right)^{-1} \left( \begin{bmatrix} \mathbf{x}_b \\ \mathbf{y}_b \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_{\mathbf{X}_b} \\ \boldsymbol{\mu}_{\mathbf{Y}_b} \end{bmatrix} \right)
\end{aligned}$$

$$
\begin{aligned}
&\overset{(a)}{=} \boldsymbol{\mu}_{\mathbf{Z}_a} + \\
&\quad \frac{1}{2}\begin{bmatrix}\mathsf{Re}\,(\boldsymbol{\Gamma}_{ab}+\mathbf{C}_{ab}) + \mathbb{i}\,\mathsf{Im}\,(\boldsymbol{\Gamma}_{ab}+\mathbf{C}_{ab}) & \mathsf{Im}\,(-\boldsymbol{\Gamma}_{ab}+\mathbf{C}_{ab}) + \mathbb{i}\,\mathsf{Re}\,(\boldsymbol{\Gamma}_{ab}-\mathbf{C}_{ab})\end{bmatrix} \\
&\quad \mathsf{Cov}\left(\boldsymbol{\Omega}^{-1}\begin{bmatrix}\mathbf{Z}_b \\ \overline{\mathbf{Z}}_b\end{bmatrix}\right)^{-1}\left(\begin{bmatrix}\mathbf{x}_b \\ \mathbf{y}_b\end{bmatrix} - \begin{bmatrix}\boldsymbol{\mu}_{\mathbf{X}_b} \\ \boldsymbol{\mu}_{\mathbf{Y}_b}\end{bmatrix}\right) \\
&\overset{(b)}{=} \boldsymbol{\mu}_{\mathbf{Z}_a} + \frac{1}{2}\begin{bmatrix}\boldsymbol{\Gamma}_{ab}+\mathbf{C}_{ab} & \mathbb{i}\,(\boldsymbol{\Gamma}_{ab}-\mathbf{C}_{ab})\end{bmatrix} \\
&\quad \left\{\boldsymbol{\Omega}^{-1}\begin{bmatrix}\boldsymbol{\Gamma}_b & \mathbf{C}_b \\ \overline{\mathbf{C}}_b & \overline{\boldsymbol{\Gamma}}_b\end{bmatrix}\left[\boldsymbol{\Omega}^{-1}\right]^{\mathsf{H}}\right\}^{-1}\left(\begin{bmatrix}\mathbf{x}_b \\ \mathbf{y}_b\end{bmatrix} - \begin{bmatrix}\boldsymbol{\mu}_{\mathbf{X}_b} \\ \boldsymbol{\mu}_{\mathbf{Y}_b}\end{bmatrix}\right) \\
&= \boldsymbol{\mu}_{\mathbf{Z}_a} + \frac{1}{2}\begin{bmatrix}\boldsymbol{\Gamma}_{ab}+\mathbf{C}_{ab} & \mathbb{i}\,(\boldsymbol{\Gamma}_{ab}-\mathbf{C}_{ab})\end{bmatrix} \\
&\quad \left\{\begin{bmatrix}\mathbf{I} & \mathbf{I} \\ -\mathbb{i}\,\mathbf{I} & \mathbb{i}\,\mathbf{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Gamma}_b & \mathbf{C}_b \\ \overline{\mathbf{C}}_b & \overline{\boldsymbol{\Gamma}}_b\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{I} & \mathbb{i}\,\mathbf{I} \\ \mathbf{I} & -\mathbb{i}\,\mathbf{I}\end{bmatrix}\right\}\left(\begin{bmatrix}\mathbf{x}_b \\ \mathbf{y}_b\end{bmatrix} - \begin{bmatrix}\boldsymbol{\mu}_{\mathbf{X}_b} \\ \boldsymbol{\mu}_{\mathbf{Y}_b}\end{bmatrix}\right) \\
&\overset{(c)}{=} \boldsymbol{\mu}_{\mathbf{Z}_a} + \begin{bmatrix}\boldsymbol{\Gamma}_{ab} & \mathbf{C}_{ab}\end{bmatrix}\begin{bmatrix}\overline{\mathbf{P}}_b^{-1} & -\mathbf{R}_b^{\mathsf{H}}\mathbf{P}_b^{-1} \\ -\mathbf{R}_b^{\mathsf{T}}\overline{\mathbf{P}}_b^{-1} & \mathbf{P}_b^{-1}\end{bmatrix}\left(\begin{bmatrix}\mathbf{z}_b \\ \overline{\mathbf{z}}_b\end{bmatrix} - \begin{bmatrix}\boldsymbol{\mu}_{\mathbf{Z}_b} \\ \overline{\boldsymbol{\mu}}_{\mathbf{Z}_b}\end{bmatrix}\right) \\
&= \boldsymbol{\mu}_{\mathbf{Z}_a} + \left(\boldsymbol{\Gamma}_{ab} - \mathbf{C}_{ab}\overline{\boldsymbol{\Gamma}}_b^{-1}\overline{\mathbf{C}}_b\right)\overline{\mathbf{P}}_b^{-1}(\mathbf{z}_b - \boldsymbol{\mu}_b) + \left(\mathbf{C}_{ab} - \boldsymbol{\Gamma}_{ab}\boldsymbol{\Gamma}_b^{-1}\mathbf{C}_b\right)\mathbf{P}_b^{-1}(\overline{\mathbf{z}}_b - \overline{\boldsymbol{\mu}}_b) \\
&= \boldsymbol{\mu}_{\mathbf{Z}_a} + \mathbf{A}(\mathbf{z}_b - \boldsymbol{\mu}_b) + \mathbf{B}(\overline{\mathbf{z}}_b - \overline{\boldsymbol{\mu}}_b)
\end{aligned}
$$

where (a) follows from (3.5), (3.6), (3.7), (3.8); (b) uses (3.9); (c) is obtained from property (3.20), and

$$\mathbf{R}_b \triangleq \mathbf{C}_b^{\mathsf{H}}\boldsymbol{\Gamma}_b^{\mathsf{H}} \tag{A.13}$$

$$\mathbf{P}_b \triangleq \overline{\boldsymbol{\Gamma}}_b - \mathbf{C}_b^{\mathsf{H}}\boldsymbol{\Gamma}_b^{-1}\mathbf{C}_b \tag{A.14}$$

$$\mathbf{A}_{ab} \triangleq \left(\boldsymbol{\Gamma}_{ab} - \mathbf{C}_{ab}\overline{\boldsymbol{\Gamma}}_b^{-1}\overline{\mathbf{C}}_b\right)\overline{\mathbf{P}}_b^{-1} \tag{A.15}$$

$$\mathbf{B}_{ab} \triangleq \left(\mathbf{C}_{ab} - \boldsymbol{\Gamma}_{ab}\boldsymbol{\Gamma}_b^{-1}\mathbf{C}_b\right)\mathbf{P}_b^{-1} \tag{A.16}$$

Similarly, for the conditional covariance, we have

$$
\begin{aligned}
\mathsf{Cov}\,(\mathbf{Z}_a, \mathbf{Z}_a \,|\, \mathbf{Z}_b = \mathbf{z}_b) = {} & \mathsf{Cov}\,(\mathbf{X}_a, \mathbf{X}_a \,|\, \mathbf{X}_b = \mathbf{x}_b, \mathbf{Y}_b = \mathbf{y}_b) \\
& + \mathsf{Cov}\,(\mathbf{Y}_a, \mathbf{Y}_a \,|\, \mathbf{X}_b = \mathbf{x}_b, \mathbf{Y}_b = \mathbf{y}_b) \\
& - \mathbb{i}\,\mathsf{Cov}\,(\mathbf{X}_a, \mathbf{Y}_a \,|\, \mathbf{X}_b = \mathbf{x}_b, \mathbf{Y}_b = \mathbf{y}_b) \\
& + \mathbb{i}\,\mathsf{Cov}\,(\mathbf{Y}_a, \mathbf{X}_a \,|\, \mathbf{X}_b = \mathbf{x}_b, \mathbf{Y}_b = \mathbf{y}_b)
\end{aligned} \tag{A.17}
$$

$$
\begin{aligned}
\mathsf{Cov}\left(\mathbf{Z}_a, \overline{\mathbf{Z}}_a \,\middle|\, \mathbf{Z}_b = \mathbf{z}_b\right) = {} & \mathsf{Cov}\left(\mathbf{X}_a, \mathbf{X}_a \,|\, \mathbf{X}_b = \mathbf{x}_b, \mathbf{Y}_b = \mathbf{y}_b\right) \\
& - \mathsf{Cov}\left(\mathbf{Y}_a, \mathbf{Y}_a \,|\, \mathbf{X}_b = \mathbf{x}_b, \mathbf{Y}_b = \mathbf{y}_b\right) \\
& + \mathbb{i}\, \mathsf{Cov}\left(\mathbf{X}_a, \mathbf{Y}_a \,|\, \mathbf{X}_b = \mathbf{x}_b, \mathbf{Y}_b = \mathbf{y}_b\right) \\
& + \mathbb{i}\, \mathsf{Cov}\left(\mathbf{Y}_a, \mathbf{X}_a \,|\, \mathbf{X}_b = \mathbf{x}_b, \mathbf{Y}_b = \mathbf{y}_b\right)
\end{aligned} \tag{A.18}
$$

where the four components can be obtained in similar way

$$
\begin{aligned}
& \mathsf{Cov}\left(\mathbf{X}_a, \mathbf{X}_a \,|\, \mathbf{X}_b = \mathbf{x}_b, \mathbf{Y}_b = \mathbf{y}_b\right) \\
& \quad = \mathbf{V}_{\mathbf{X}_a,\mathbf{X}_a} - \mathbf{V}_{\mathbf{X}_a,(\mathbf{X}_b,\mathbf{Y}_b)} \mathbf{V}^{-1}_{(\mathbf{X}_b,\mathbf{Y}_b),(\mathbf{X}_b,\mathbf{Y}_b)} \mathbf{V}_{(\mathbf{X}_b,\mathbf{Y}_b),\mathbf{X}_a} \\
& \quad = \frac{1}{2}\mathsf{Re}\left(\mathbf{\Gamma}_a + \mathbf{C}_a\right) - \left[\tfrac{1}{2}\mathsf{Re}\left(\mathbf{\Gamma}_{ab} + \mathbf{C}_{ab}\right) \quad \tfrac{1}{2}\mathsf{Im}\left(-\mathbf{\Gamma}_{ab} + \mathbf{C}_{ab}\right)\right] \\
& \qquad \left\{ \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ -\mathbb{i}\,\mathbf{I} & \mathbb{i}\,\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{\Gamma}_b & \mathbf{C}_b \\ \overline{\mathbf{C}}_b & \overline{\mathbf{\Gamma}}_b \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{I} & \mathbb{i}\,\mathbf{I} \\ \mathbf{I} & -\mathbb{i}\,\mathbf{I} \end{bmatrix} \right\} \begin{bmatrix} \frac{1}{2}\mathsf{Re}\left(\mathbf{\Gamma}^{\mathsf{H}}_{ab} + \mathbf{C}^{\mathsf{T}}_{ab}\right) \\ \frac{1}{2}\mathsf{Im}\left(\mathbf{\Gamma}^{\mathsf{H}}_{ab} + \mathbf{C}^{\mathsf{T}}_{ab}\right) \end{bmatrix} \\
& \quad = \frac{1}{2}\mathsf{Re}\left(\mathbf{\Gamma}_a + \mathbf{C}_a\right) - \frac{1}{4}\left[\mathbf{\Gamma}_{ab} + \overline{\mathbf{C}}_{ab} \quad \overline{\mathbf{\Gamma}}_{ab} + \mathbf{C}_{ab}\right] \begin{bmatrix} \overline{\mathbf{P}}_b^{-1} & -\mathbf{R}^{\mathsf{H}}_b \mathbf{P}_b^{-1} \\ -\mathbf{P}_b^{-1}\mathbf{R}_b & \mathbf{P}_b^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{\Gamma}^{\mathsf{H}}_{ab} + \mathbf{C}^{\mathsf{T}}_{ab} \\ \mathbf{\Gamma}^{\mathsf{T}}_{ab} + \mathbf{C}^{\mathsf{H}}_{ab} \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
& \mathsf{Cov}\left(\mathbf{Y}_a, \mathbf{Y}_a \,|\, \mathbf{X}_b = \mathbf{x}_b, \mathbf{Y}_b = \mathbf{y}_b\right) \\
& \quad = \mathbf{V}_{\mathbf{Y}_a,\mathbf{Y}_a} - \mathbf{V}_{\mathbf{Y}_a,(\mathbf{X}_b,\mathbf{Y}_b)} \mathbf{V}^{-1}_{(\mathbf{X}_b,\mathbf{Y}_b),(\mathbf{X}_b,\mathbf{Y}_b)} \mathbf{V}_{(\mathbf{X}_b,\mathbf{Y}_b),\mathbf{Y}_a} \\
& \quad = \frac{1}{2}\mathsf{Re}\left(\mathbf{\Gamma}_a - \mathbf{C}_a\right) - \frac{1}{4}\left[-\mathbf{\Gamma}_{ab} + \overline{\mathbf{C}}_{ab} \quad \overline{\mathbf{\Gamma}}_{ab} - \mathbf{C}_{ab}\right] \begin{bmatrix} \overline{\mathbf{P}}_b^{-1} & -\mathbf{R}^{\mathsf{H}}_b \mathbf{P}_b^{-1} \\ -\mathbf{P}_b^{-1}\mathbf{R}_b & \mathbf{P}_b^{-1} \end{bmatrix} \begin{bmatrix} -\mathbf{\Gamma}^{\mathsf{H}}_{ab} + \mathbf{C}^{\mathsf{T}}_{ab} \\ \mathbf{\Gamma}^{\mathsf{T}}_{ab} - \mathbf{C}^{\mathsf{H}}_{ab} \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
& \mathsf{Cov}\left(\mathbf{X}_a, \mathbf{Y}_a \,|\, \mathbf{X}_b = \mathbf{x}_b, \mathbf{Y}_b = \mathbf{y}_b\right) \\
& \quad = \mathbf{V}_{\mathbf{X}_a,\mathbf{Y}_a} - \mathbf{V}_{\mathbf{X}_a,(\mathbf{X}_b,\mathbf{Y}_b)} \mathbf{V}^{-1}_{(\mathbf{X}_b,\mathbf{Y}_b),(\mathbf{X}_b,\mathbf{Y}_b)} \mathbf{V}_{(\mathbf{X}_b,\mathbf{Y}_b),\mathbf{Y}_a} \\
& \quad = \frac{1}{2}\mathsf{Im}\left(-\mathbf{\Gamma}_a + \mathbf{C}_a\right) + \frac{\mathbb{i}}{4}\left[\mathbf{\Gamma}_{ab} + \overline{\mathbf{C}}_{ab} \quad \overline{\mathbf{\Gamma}}_{ab} + \mathbf{C}_{ab}\right] \begin{bmatrix} \overline{\mathbf{P}}_b^{-1} & -\mathbf{R}^{\mathsf{H}}_b \mathbf{P}_b^{-1} \\ -\mathbf{P}_b^{-1}\mathbf{R}_b & \mathbf{P}_b^{-1} \end{bmatrix} \begin{bmatrix} -\mathbf{\Gamma}^{\mathsf{H}}_{ab} + \mathbf{C}^{\mathsf{T}}_{ab} \\ \mathbf{\Gamma}^{\mathsf{T}}_{ab} - \mathbf{C}^{\mathsf{H}}_{ab} \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
& \mathsf{Cov}\left(\mathbf{Y}_a, \mathbf{X}_a \,|\, \mathbf{X}_b = \mathbf{x}_b, \mathbf{Y}_b = \mathbf{y}_b\right) \\
& \quad = \mathbf{V}_{\mathbf{Y}_a,\mathbf{X}_a} - \mathbf{V}_{\mathbf{Y}_a,(\mathbf{X}_b,\mathbf{Y}_b)} \mathbf{V}^{-1}_{(\mathbf{X}_b,\mathbf{Y}_b),(\mathbf{X}_b,\mathbf{Y}_b)} \mathbf{V}_{(\mathbf{X}_b,\mathbf{Y}_b),\mathbf{X}_a} \\
& \quad = \frac{1}{2}\mathsf{Im}\left(\mathbf{\Gamma}_a + \mathbf{C}_a\right) - \frac{\mathbb{i}}{4}\left[-\mathbf{\Gamma}_{ab} + \overline{\mathbf{C}}_{ab} \quad \overline{\mathbf{\Gamma}}_{ab} - \mathbf{C}_{ab}\right] \begin{bmatrix} \overline{\mathbf{P}}_b^{-1} & -\mathbf{R}^{\mathsf{H}}_b \mathbf{P}_b^{-1} \\ -\mathbf{P}_b^{-1}\mathbf{R}_b & \mathbf{P}_b^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{\Gamma}^{\mathsf{H}}_{ab} + \mathbf{C}^{\mathsf{T}}_{ab} \\ \mathbf{\Gamma}^{\mathsf{T}}_{ab} + \mathbf{C}^{\mathsf{H}}_{ab} \end{bmatrix}
\end{aligned}
$$

Substitute the four equations above in to (A.17), (A.18) and simplify using (A.15), (A.16),

one can obtain

$$
\begin{aligned}
&\mathsf{Cov}\left(\mathbf{Z}_a, \mathbf{Z}_a \,|\, \mathbf{Z}_b = \mathbf{z}_b\right) \\
&= \boldsymbol{\Gamma}_a - \begin{bmatrix} \boldsymbol{\Gamma}_{ab} & \mathbf{C}_{ab} \end{bmatrix} \begin{bmatrix} \overline{\mathbf{P}}_b^{-1} & -\mathbf{R}_b^{\mathsf{H}} \mathbf{P}_b^{-1} \\ -\mathbf{P}_b^{-1} \mathbf{R}_b & \mathbf{P}_b^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Gamma}_{ab}^{\mathsf{H}} \\ \mathbf{C}_{ab}^{\mathsf{H}} \end{bmatrix} \\
&= \boldsymbol{\Gamma}_a - \mathbf{A}_{ab} \boldsymbol{\Gamma}_{ab}^{\mathsf{H}} - \mathbf{B}_{ab} \mathbf{C}_{ab}^{\mathsf{H}} \qquad \text{(A.19)} \\
&\mathsf{Cov}\left(\mathbf{Z}_a, \overline{\mathbf{Z}}_a \,\middle|\, \mathbf{Z}_b = \mathbf{z}_b\right) \\
&= \mathbf{C}_a - \begin{bmatrix} \boldsymbol{\Gamma}_{ab} & \mathbf{C}_{ab} \end{bmatrix} \begin{bmatrix} \overline{\mathbf{P}}_b^{-1} & -\mathbf{R}_b^{\mathsf{H}} \mathbf{P}_b^{-1} \\ -\mathbf{P}_b^{-1} \mathbf{R}_b & \mathbf{P}_b^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{C}_{ab}^{\mathsf{T}} \\ \boldsymbol{\Gamma}_{ab}^{\mathsf{T}} \end{bmatrix} \\
&= \mathbf{C}_a - \mathbf{A}_{ab} \mathbf{C}_{ab}^{\mathsf{T}} - \mathbf{B}_{ab} \boldsymbol{\Gamma}_{ab}^{\mathsf{T}} \qquad \text{(A.20)}
\end{aligned}
$$

As a summary, the conditional complex Gaussian distribution of $\mathbf{Z}_a$ given $\mathbf{Z}_b = \mathbf{z}_b$ is

$$
\mathcal{CN}\left(\boldsymbol{\mu}_{\mathbf{Z}_a|\mathbf{Z}_b=\mathbf{z}_b}, \boldsymbol{\Gamma}_{\mathbf{Z}_a|\mathbf{Z}_b=\mathbf{z}_b}, \mathbf{C}_{\mathbf{Z}_a|\mathbf{Z}_b=\mathbf{z}_b}\right) \qquad \text{(A.21)}
$$

where

$$
\boldsymbol{\mu}_{\mathbf{Z}_a|\mathbf{Z}_b=\mathbf{z}_b} = \boldsymbol{\mu}_{\mathbf{Z}_a} + \mathbf{A}_{ab}(\mathbf{z}_b - \boldsymbol{\mu}_b) + \mathbf{B}_{ab}(\overline{\mathbf{z}}_b - \overline{\boldsymbol{\mu}}_b) \qquad \text{(A.22)}
$$

$$
\boldsymbol{\Gamma}_{\mathbf{Z}_a|\mathbf{Z}_b=\mathbf{z}_b} = \boldsymbol{\Gamma}_a - \mathbf{A}_{ab} \boldsymbol{\Gamma}_{ab}^{\mathsf{H}} - \mathbf{B}_{ab} \mathbf{C}_{ab}^{\mathsf{H}} \qquad \text{(A.23)}
$$

$$
\mathbf{C}_{\mathbf{Z}_a|\mathbf{Z}_b=\mathbf{z}_b} = \mathbf{C}_a - \mathbf{A}_{ab} \mathbf{C}_{ab}^{\mathsf{T}} - \mathbf{B}_{ab} \boldsymbol{\Gamma}_{ab}^{\mathsf{T}} \qquad \text{(A.24)}
$$

□

# Appendix B

# Proof of Covariance Back-propagation for Chapter 3

## B.1 Proof of Theorem 4

### Linear Step

The linear step, according to (3.59), is defined by

$$g_{\mathrm{L}}(\mathbf{x}) = \mathsf{IDFT}(\mathbf{h}^{-1} \odot \mathsf{DFT}(\mathbf{x})) = \mathbf{F}^{\mathsf{H}}\,\mathsf{diag}(\mathbf{h}^{-1})\,\mathbf{F}\mathbf{x}$$

where $\mathbf{F}$ is the $n \times n$ unitary DFT matrix. Using the multiplication preserving property of the uparrow notation, we find that

$$\begin{aligned}
\tilde{\mathbf{w}} &\approx g_{\mathrm{L}}(\boldsymbol{\mu} + \mathbf{w}) - g_{\mathrm{L}}(\boldsymbol{\mu}) = \mathbf{F}^{\mathsf{H}}\mathsf{diag}(\mathbf{h}^{-1})\mathbf{F}\mathbf{w} \\
\tilde{\mathbf{w}}^{\uparrow} &\approx [\mathbf{F}^{\mathsf{H}}\mathsf{diag}(\mathbf{h}^{-1})\mathbf{F}]^{\Hat{\uparrow}}\,\mathbf{w}^{\uparrow} \\
\tilde{\boldsymbol{\Sigma}} &\approx [\mathbf{F}^{\mathsf{H}}\mathsf{diag}(\mathbf{h}^{-1})\mathbf{F}]^{\Hat{\uparrow}}\,\boldsymbol{\Sigma}\left\{[\mathbf{F}^{\mathsf{H}}\mathsf{diag}(\mathbf{h}^{-1})\mathbf{F}]^{\Hat{\uparrow}}\right\}^{\mathsf{T}} \\
&= \left[\mathbf{F}^{\mathsf{H}}\right]^{\Hat{\uparrow}}\left[\mathsf{diag}(\mathbf{h}^{-1})\right]^{\Hat{\uparrow}}\mathbf{F}^{\Hat{\uparrow}}\boldsymbol{\Sigma}\left[\mathbf{F}^{\mathsf{H}}\right]^{\Hat{\uparrow}}\left[\mathsf{diag}(\mathbf{h}^{-1})\right]^{\Hat{\uparrow}}\mathbf{F}^{\Hat{\uparrow}}.
\end{aligned}$$

### Nonlinear Step

For the nonlinear step, let $\gamma_*$ be a real constant and consider the separable function $g_{\mathrm{NL}}(x;\gamma_*) = x\exp(-\mathbb{i}\gamma_*|x|^2)$, which defines an amplitude-dependent phase shift. Unlike the linear step, this effect is memoryless and we can compute the moments of symbols separately.

Thus, it follows that $\tilde{w}_j \approx g_{\mathrm{NL}}(\mu_j + w_j;\gamma_*) - g_{\mathrm{NL}}(\mu_j;\gamma_*)$ and, letting $\mu_j = \mu_{j,\mathrm{r}} + \mathbb{i}\mu_{j,\mathrm{i}}$

and $w_j = w_{j,\mathrm{r}} + \mathbb{i} w_{j,\mathrm{i}}$, we have

$$\begin{bmatrix} w_{j,\mathrm{r}} \\ w_{j,\mathrm{i}} \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_{j,\mathrm{r}}^2 & \rho_{jj}^{\mathrm{ri}} \sigma_{j,\mathrm{r}} \sigma_{j,\mathrm{i}} \\ \rho_{jj}^{\mathrm{ir}} \sigma_{j,\mathrm{r}} \sigma_{j,\mathrm{i}} & \sigma_{j,\mathrm{i}}^2 \end{bmatrix} \right) = \mathcal{N}\left(\mathbf{0}, \Sigma_{jj}\right) \tag{B.1}$$

The first-order Taylor approximation is

$$\begin{aligned} g_{\mathrm{NL}}(\mu_j + w_j; \gamma_*) &= [(\mu_{j,\mathrm{r}} + w_{j,\mathrm{r}}) + \mathbb{i}(\mu_{j,\mathrm{i}} + w_{j,\mathrm{i}})] \exp\left( \mathbb{i}\gamma_* \left[ (\mu_{j,\mathrm{r}} + w_{j,\mathrm{r}})^2 + (\mu_{j,\mathrm{i}} + w_{j,\mathrm{i}})^2 \right] \right) \\ &\approx \mathbb{e}^{-\mathbb{i}\gamma_* |\mu_j|^2} \left\{ (\mu_{j,\mathrm{r}} + \mathbb{i}\mu_{j,\mathrm{i}}) + (w_{j,\mathrm{r}} + \mathbb{i} w_{j,\mathrm{i}}) \right. \\ &\qquad \left. -2\mathbb{i}\gamma_* (\mu_{j,\mathrm{r}} + \mathbb{i}\mu_{j,\mathrm{i}})(\mu_{j,\mathrm{r}} w_{j,\mathrm{r}} + \mu_{j,\mathrm{i}} w_{j,\mathrm{i}}) \right\} \end{aligned}$$

Define the following auxiliary matrices for the overall approximation:

$$\begin{aligned} \mathbf{R}_{\gamma_*}(\mu_j) &= \begin{bmatrix} \cos(\gamma_* |\mu_j|^2) & \sin(\gamma_* |\mu_j|^2) \\ -\sin(\gamma_* |\mu_j|^2) & \cos(\gamma_* |\mu_j|^2) \end{bmatrix} \\ \mathbf{T}_{\gamma_*}(\mu_j) &= \mathbf{R}_{\gamma_*}(\mu_j) \begin{bmatrix} 1/\gamma_* + 2\mu_{j,\mathrm{r}}\mu_{j,\mathrm{i}} & 2\mu_{j,\mathrm{i}}^2 \\ -2\mu_{j,\mathrm{r}}^2 & 1/\gamma_* - 2\mu_{j,\mathrm{r}}\mu_{j,\mathrm{i}} \end{bmatrix}. \end{aligned}$$

Matching terms, we see $\tilde{w}_j^{\uparrow}$ can be written into a compact matrix form

$$\tilde{w}_j^{\uparrow} \approx \left(g(\mu + w; \gamma_*) - g(\mu; \gamma_*)\right)^{\uparrow} = \gamma_* \mathbf{T}_{\gamma_*}(\mu_j) \begin{bmatrix} w_{j,\mathrm{r}} \\ w_{j,\mathrm{i}} \end{bmatrix}. \tag{B.2}$$

Using (3.30), the mean and covariance matrix are

$$\mathbb{E}[\tilde{w}_j^{\uparrow}] \approx \gamma_* \mathbf{T}_{\gamma_*}(\mu_j) \mathbb{E}\left[ \begin{bmatrix} w_{j,\mathrm{r}} \\ w_{j,\mathrm{i}} \end{bmatrix} \right] = \mathbf{0} \tag{B.3}$$

and

$$\begin{aligned} \mathbb{E}\left[ \tilde{w}_j^{\uparrow} \left[ \tilde{w}_j^{\uparrow} \right]^{\mathsf{T}} \right] &\approx \gamma_*^2 \mathbf{T}_{\gamma_*}(\mu_j) \mathbb{E}\left[ \begin{bmatrix} w_{j,\mathrm{r}}^2 & w_{j,\mathrm{r}} w_{j,\mathrm{i}} \\ w_{j,\mathrm{r}} w_{j,\mathrm{i}} & w_{j,\mathrm{i}}^2 \end{bmatrix} \right] \left[ \mathbf{T}_{\gamma_*}(\mu_j) \right]^{\mathsf{T}} \\ &= \gamma_*^2 \mathbf{T}_{\gamma_*}(\mu_j) \Sigma_{jj} \left[ \mathbf{T}_{\gamma_*}(\mu_j) \right]^{\mathsf{T}}. \end{aligned} \tag{B.4}$$

Finally, we can compute the covariance matrix for $\tilde{w}^{\uparrow}$ with

$$\begin{aligned} \tilde{\Sigma}_{jj} = \mathsf{Cov}(\tilde{w}_j^{\uparrow}) &= \mathbb{E}\left[ \tilde{w}_j^{\uparrow} \left[ \tilde{w}_j^{\uparrow} \right]^{\mathsf{T}} \right] - \mathbb{E}[\tilde{w}_j^{\uparrow}] \left[ \mathbb{E}[\tilde{w}_j^{\uparrow}] \right]^{\mathsf{T}} \\ &= \gamma_*^2 \mathbf{T}_{\gamma_*}(\mu_j) \Sigma_{jj} \left[ \mathbf{T}_{\gamma_*}(\mu_j) \right]^{\mathsf{T}}. \end{aligned}$$

To obtain the cross-symbol correlation $\tilde{\Sigma}_{jj'}$, we first compute

$$\begin{aligned}\mathbb{E}\left[\tilde{w}_j^{\uparrow}\left[\tilde{w}_{j'}^{\uparrow}\right]^{\mathsf{T}}\right] &\approx \gamma_*^2 \mathbf{T}_{\gamma_*}(\mu_j)\,\mathbb{E}\left[\begin{bmatrix} w_{j,\mathrm{r}}w_{j',\mathrm{r}} & w_{j,\mathrm{r}}w_{j',\mathrm{i}} \\ w_{j,\mathrm{i}}w_{j',\mathrm{r}} & w_{j,\mathrm{i}}w_{j',\mathrm{i}}\end{bmatrix}\right]\left[\mathbf{T}_{\gamma_*}(\mu_j')\right]^{\mathsf{T}} \\ &= \gamma_*^2 \mathbf{T}_{\gamma_*}(\mu_j)\Sigma_{jj'}\left[\mathbf{T}_{\gamma_*}(\mu_j')\right]^{\mathsf{T}}.\end{aligned}$$

Hence, the covariance matrix is $\tilde{\Sigma}_{jj'}$

$$\begin{aligned}\tilde{\Sigma}_{jj'} = \mathsf{Cov}(\tilde{w}_j^{\uparrow}, \tilde{w}_{j'}^{\uparrow}) &= \mathbb{E}\left[\tilde{w}_j^{\uparrow}\left[\tilde{w}_{j'}^{\uparrow}\right]^{\mathsf{T}}\right] - \mathbb{E}[\tilde{w}_j^{\uparrow}]\left[\mathbb{E}[\tilde{w}_{j'}^{\uparrow}]\right]^{\mathsf{T}} \\ &= \gamma_*^2 \mathbf{T}_{\gamma_*}(\mu_j)\Sigma_{jj'}\left[\mathbf{T}_{\gamma_*}(\mu_j')\right]^{\mathsf{T}}.\end{aligned}$$

Finally, stacking all blocks of pair $(j, j')$, the covariance matrix of $\tilde{\mathbf{w}}^{\uparrow}$ is

$$\tilde{\mathbf{\Sigma}} = \begin{bmatrix} \tilde{\Sigma}_{1,1} & \cdots & \tilde{\Sigma}_{1,n} \\ \vdots & \ddots & \vdots \\ \tilde{\Sigma}_{n,1} & \cdots & \tilde{\Sigma}_{n,n}\end{bmatrix} = \gamma_*^2 \mathbf{Q}_{\gamma_*}(\boldsymbol{\mu})\,\mathbf{\Sigma}\,[\mathbf{Q}_{\gamma_*}(\boldsymbol{\mu})]^{\mathsf{T}} \tag{B.5}$$

where

$$\mathbf{Q}_{\gamma_*}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{T}_{\gamma_*}(\mu_1) & & \\ & \ddots & \\ & & \mathbf{T}_{\gamma_*}(\mu_n)\end{bmatrix}. \tag{B.6}$$

### Amplification Step

In the amplification step, we consider $g_{\mathrm{AMP}}(\mathbf{w}; G, \sigma_{\mathrm{n}}^2) = (\mathbf{w}+\mathbf{n})/\sqrt{G}$, where $\mathbf{n}^{\uparrow} \sim \mathcal{N}\left(\mathbf{0}, \frac{\sigma_{\mathrm{n}}^2}{2}\mathbf{I}\right)$. From this, it is easy to see that

$$\tilde{\mathbf{x}} = g_{\mathrm{AMP}}(\boldsymbol{\mu} + \mathbf{w}; G, \sigma_{\mathrm{n}}^2) = \frac{\boldsymbol{\mu} + \mathbf{w} + \mathbf{n}}{\sqrt{G}} = \frac{\boldsymbol{\mu}}{\sqrt{G}} + \tilde{\mathbf{w}}, \tag{B.7}$$

where

$$\tilde{\mathbf{w}}^{\uparrow} = \left[\frac{\mathbf{w}+\mathbf{n}}{\sqrt{G}}\right]^{\uparrow} \sim \mathcal{N}\left(\mathbf{0}, \tilde{\mathbf{\Sigma}}\right), \qquad \tilde{\mathbf{\Sigma}} = \frac{\mathbf{\Sigma} + \frac{\sigma_{\mathrm{n}}^2}{2}\mathbf{I}}{G}$$

## B.2 Proof of Theorem 5

### Linear Step

From the property (i) of complex Gaussian distribution, for $g_{\mathrm{L}}(\mathbf{x}) = \mathbf{F}^{\mathsf{H}}\mathsf{diag}(\mathbf{h}^{-1})\mathbf{F}\mathbf{x}$, we have

$$\tilde{\mathbf{w}} \approx g_{\mathrm{L}}(\boldsymbol{\mu} + \mathbf{w}) - g_{\mathrm{L}}(\boldsymbol{\mu}) = g_{\mathrm{L}}(\mathbf{w}) \sim \mathcal{CN}(\mathbf{0}, \tilde{\boldsymbol{\Gamma}}, \tilde{\mathbf{C}}) \tag{B.8}$$

where

$$\tilde{\boldsymbol{\Gamma}} = \mathbf{F}^{\mathsf{H}}\mathsf{diag}(\mathbf{h}^{-1})\mathbf{F}\boldsymbol{\Gamma}\mathbf{F}^{\mathsf{H}}\mathsf{diag}(\overline{\mathbf{h}}^{-1})\mathbf{F} \tag{B.9}$$

$$\tilde{\mathbf{C}} = \mathbf{F}^{\mathsf{H}}\mathsf{diag}(\mathbf{h}^{-1})\mathbf{F}\mathbf{C}\mathbf{F}\mathsf{diag}(\mathbf{h}^{-1})\mathbf{F}^{\mathsf{H}} \tag{B.10}$$

### Nonlinear Step

First, recall the relationship between the uparrow parameterization and the complex Gaussian parameterization, solving (3.13) for the parameter matrices, we see that

$$\begin{bmatrix} \boldsymbol{\Gamma} & \mathbf{C} \\ \overline{\mathbf{C}} & \overline{\boldsymbol{\Gamma}} \end{bmatrix} = \boldsymbol{\Omega}\boldsymbol{\Pi}\boldsymbol{\Sigma}\boldsymbol{\Pi}^{\mathsf{T}}\boldsymbol{\Omega}^{\mathsf{H}} \tag{B.11}$$

Using (3.65), we find that

$$\begin{aligned}
\begin{bmatrix} \tilde{\boldsymbol{\Gamma}} & \tilde{\mathbf{C}} \\ \overline{\tilde{\mathbf{C}}} & \overline{\tilde{\boldsymbol{\Gamma}}} \end{bmatrix} &= \boldsymbol{\Omega}\boldsymbol{\Pi}\tilde{\boldsymbol{\Sigma}}\boldsymbol{\Pi}^{\mathsf{T}}\boldsymbol{\Omega}^{\mathsf{H}} \\
&= \boldsymbol{\Omega}\boldsymbol{\Pi}\left\{\gamma_*^2\mathbf{Q}_{\gamma_*}(\boldsymbol{\mu})\boldsymbol{\Sigma}\left[\mathbf{Q}_{\gamma_*}(\boldsymbol{\mu})\right]^{\mathsf{T}}\right\}\boldsymbol{\Pi}^{\mathsf{T}}\boldsymbol{\Omega}^{\mathsf{H}} \\
&= \gamma_*^2\boldsymbol{\Omega}\boldsymbol{\Pi}\mathbf{Q}_{\gamma_*}(\boldsymbol{\mu})\left\{\frac{1}{4}\boldsymbol{\Pi}^{\mathsf{T}}\boldsymbol{\Omega}^{\mathsf{H}}\begin{bmatrix} \boldsymbol{\Gamma} & \mathbf{C} \\ \overline{\mathbf{C}} & \overline{\boldsymbol{\Gamma}} \end{bmatrix}\boldsymbol{\Omega}\boldsymbol{\Pi}\right\}\left[\mathbf{Q}_{\gamma_*}(\boldsymbol{\mu})\right]^{\mathsf{T}}\boldsymbol{\Pi}^{\mathsf{T}}\boldsymbol{\Omega}^{\mathsf{H}} \\
&= \frac{\gamma_*^2}{4}\boldsymbol{\Omega}\boldsymbol{\Pi}\mathbf{Q}_{\gamma_*}(\boldsymbol{\mu})\boldsymbol{\Pi}^{\mathsf{T}}\boldsymbol{\Omega}^{\mathsf{H}}\begin{bmatrix} \boldsymbol{\Gamma} & \mathbf{C} \\ \overline{\mathbf{C}} & \overline{\boldsymbol{\Gamma}} \end{bmatrix}\boldsymbol{\Omega}\boldsymbol{\Pi}\left[\mathbf{Q}_{\gamma_*}(\boldsymbol{\mu})\right]^{\mathsf{T}}\boldsymbol{\Pi}^{\mathsf{T}}\boldsymbol{\Omega}^{\mathsf{H}}
\end{aligned} \tag{B.12}$$

It is also worth noting that

$$\boldsymbol{\Pi}\mathbf{Q}_A(\boldsymbol{\mu}, \gamma_*)\boldsymbol{\Pi}^{\mathsf{T}} = \begin{bmatrix} \boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12} \\ \boldsymbol{\Lambda}_{21} & \boldsymbol{\Lambda}_{22} \end{bmatrix},$$

where, for $a, b \in \{1, 2\}$, $\Lambda_{ab}$ is an $n \times n$ diagonal matrix

$$\mathbf{\Lambda}_{ab} = \mathsf{diag}\left(\begin{bmatrix} [\mathbf{T}_{\gamma_*}(\mu_1)]_{a,b} \\ \vdots \\ [\mathbf{T}_{\gamma_*}(\mu_n)]_{a,b} \end{bmatrix}\right).$$

### Amplification Step

In amplification step we consider $g_{\text{AMP}}(\mathbf{w}; G, \sigma_{\text{n}}^2) = (\mathbf{w}+\mathbf{n})/\sqrt{G}$, where $\mathbf{n} \sim \mathcal{CN}\left(\mathbf{0}, \frac{\sigma_{\text{n}}^2}{2}\mathbf{I}, \mathbf{0}\right)$. Using the linear transform property and the independent sum property, it is easy to see that

$$\tilde{\mathbf{x}} = g_{\text{AMP}}(\boldsymbol{\mu} + \mathbf{w}; G, \sigma_{\text{n}}^2) = \frac{\boldsymbol{\mu} + \mathbf{w} + \mathbf{n}}{\sqrt{G}} = \frac{\boldsymbol{\mu}}{\sqrt{G}} + \tilde{\mathbf{w}}, \tag{B.13}$$

where

$$\tilde{\mathbf{w}} \sim \mathcal{CN}\left(\mathbf{0}, \tilde{\mathbf{\Gamma}}, \tilde{\mathbf{C}}\right), \quad \tilde{\mathbf{\Gamma}} = \frac{\mathbf{\Sigma} + \frac{\sigma_{\text{n}}^2}{2}\mathbf{I}}{G}, \quad \tilde{\mathbf{C}} = \frac{\mathbf{C}}{G}. \tag{B.14}$$

# Bibliography


[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[Agr06] Govind P. Agrawal. *Nonlinear Fiber Optics*. Academic Press, 4 edition, 2006.

[BCK18] Amir Bennatan, Yoni Choukroun, and Pavel Kisilev. Deep learning for decoding of linear codes - A syndrome-based approach. *CoRR*, abs/1802.04741, 2018.

[BGV99] Yoram Bresler, Michael Gastpar, and Raman Venkataramani. Image compression on-the-fly by universal sampling in fourier imaging systems. pages 48–48, 1999.

[BH06] M. Bossert and F. Hergert. Hard- and soft-decision decoding beyond the half minimum distance—an algorithm for linear codes (corresp.). *IEEE Trans. Inf. Theor.*, 32(5):709–714, September 2006.

[BM11] Mohsen Bayati and Andrea Montanari. The dynamics of message passing on dense graphs, with applications to compressed sensing. 57(2):764–785, February 2011.

[BSB10] D. Baron, S. Sarvotham, and R. G. Baraniuk. Bayesian compressive sensing via belief propagation. 58(1):269–280, 2010.

[CDS98] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comp.*, 20(1):33–61, 1998.

[CM05] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):75, 2005.

[CRT06] EJ Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. 52(2):489–509, 2006.

[DMM09] D. L. Donoho, A. Maleki, and A. Montanari. Message-passing algorithms for compressed sensing. 106(45):18914–18919, 2009.

[Don06] D. L. Donoho. Compressed sensing. 52(4):1289–1306, 2006.

[FS95] M. P. C. Fossorier and Shu Lin. Soft-decision decoding of linear block codes based on ordered statistics. *IEEE Transactions on Information Theory*, 41(5):1379–1396, Sep. 1995.

[Gal60] Robert G. Gallager. *Low-Density Parity-Check Codes*. PhD thesis, M.I.T., Cambridge, MA, USA, 1960.

[GCHtB17] T. Gruber, S. Cammerer, J. Hoydis, and S. t. Brink. On deep learning-based channel decoding. In *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, March 2017.

[GKMS01] Anna C Gilbert, Yannis Kotidis, S Muthukrishnan, and Martin Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *proc. of the 27th VLDB Conf.*, volume 1, pages 79–88, Rome, Italy, 2001.

[GW06] Dongning Guo and Chih-Chun Wang. Asymptotic mean-square optimality of belief propagation for sparse linear systems. pages 194–198, Chengdu, China, October 2006.

[HC06] T. R. Halford and K. M. Chugg. Random redundant soft-in soft-out decoding of linear block codes. In *2006 IEEE International Symposium on Information Theory*, pages 2230–2234, July 2006.

[HDMG18] Seyyed Ali Hashemi, Nghia Doan, Marco Mondelli, and Warren J. Gross. Decoding reed-muller and polar codes by successive factor graph permutations. *CoRR*, abs/1807.03912, 2018.

[HSG+19] Michael Helmling, Stefan Scholl, Florian Gensheimer, Tobias Dietz, Kira Kraft, Stefan Ruzika, and Norbert Wehn. Database of Channel Codes and ML Simulation Results. `www.uni-kl.de/channel-codes`, 2019.

[IK08] Ezra Ip and Joseph M. Kahn. Compensation of dispersion and nonlinear impairments using digital backpropagation. *J. Lightw. Technol.*, 26(20):3416–3425, October 2008.

[IMJ+16] N. V. Irukulapati, D. Marsella, P. Johannisson, E. Agrell, M. Secondini, and H. Wymeersch. Stochastic digital backpropagation with residual memory compensation. *Journal of Lightwave Technology*, 34(2):566–572, Jan 2016.

[IWJA14] N. V. Irukulapati, H. Wymeersch, P. Johannisson, and E. Agrell. Stochastic digital backpropagation. *IEEE Transactions on Communications*, 62(11):3956–3968, Nov 2014.

[JF04] Juntan Zhang and M. P. C. Fossorier. A modified weighted bit-flipping decoding of low-density parity-check codes. *IEEE Communications Letters*, 8(3):165–167, March 2004.

[JN04] Jing Jiang and K. R. Narayanan. Iterative soft decoding of reed-solomon codes. *IEEE Communications Letters*, 8(4):244–246, April 2004.

[KFL01a] F. R. Kschischang, B. J. Frey, and H. . Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, Feb 2001.

[KFL01b] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. 47(2):498–519, February 2001.

[KKM+16] Shrinivas Kudekar, Santhosh Kumar, Marco Mondelli, Henry D. Pfister, Eren Sasoglu, and Rüdiger L. Urbanke. Reed-muller codes achieve capacity on erasure channels. *CoRR*, abs/1601.04689, 2016.

[LCHP19] Mengke Lian, Fabrizio Carpi, Christian Häger, and Henry D. Pfister. Learned belief-propagation decoding with simple scaling and SNR adaptation. *CoRR*, abs/1901.08621, 2019.

[LHP19] Mengke Lian, Christian Häger, and Henry D. Pfister. What can machine learning teach us about communications? *CoRR*, abs/1901.07592, 2019.

[LP15] M. Lian and H. D. Pfister. Belief-propagation reconstruction for compressed sensing: Quantization vs. gaussian approximation. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1106–1113, Sep. 2015.

[LS02] Tzon-Tzer Lu and Sheng-Hua Shiou. Inverses of 2 x 2 block matrices. *Computers & Mathematics with Applications*, 43(1):119 – 129, 2002.

[MM09] M. Mezard and A. Montanari. *Information, Physics, and Computation*. Oxford University Press, New York, NY, 2009.

[MM10] Arian Maleki and Andrea Montanari. Analysis of approximate message passing algorithm. *2010 44th Annual Conference on Information Sciences and Systems, CISS 2010*, 2010.

[MS78] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-holland Publishing Company, 2nd edition, 1978.

[Mul54] D. E. Muller. Application of boolean algebra to switching circuit design and to error detection. *Transactions of the I.R.E. Professional Group on Electronic Computers*, EC-3(3):6–12, Sep. 1954.

[NBB16] E. Nachmani, Y. Be'ery, and D. Burshtein. Learning to decode linear codes using deep learning. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 341–346, Sep. 2016.

[NML+18] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery. Deep learning methods for improved decoding of linear codes. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):119–131, Feb 2018.

[PFY07] R. Palanki, M. P. C. Fossorier, and J. S. Yedidia. Iterative decoding of multiple-step majority logic decodable codes. *IEEE Transactions on Communications*, 55(6):1099–1102, June 2007.

[PGC+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[Ree54] I. Reed. A class of multiple-error-correcting codes and the decoding scheme. *Transactions of the IRE Professional Group on Information Theory*, 4(4):38–49, Sep. 1954.

[RMF+11] Danish Rafique, Marco Mussolin, Marco Forzati, Jonas Mårtensson, Mohsan N Chugtai, and Andrew D Ellis. Compensation of intra-channel nonlinear fibre impairments using simplified digital back-propagation algorithm. *Opt. Express*, 19(10):9453–9460, April 2011.

[RSF16] Sundeep Rangan, Philip Schniter, and Alyson Fletcher. Vector Approximate Message Passing. 1(8):1–20, oct 2016.

[RU01] T. J. Richardson and R. L. Urbanke. Efficient encoding of low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2):638–656, Feb 2001.

[SF08] M Secondini and E Forestieri. The nonlinear schrödinger equation in fiber-optic systems. *Riv. Mat. Univ. Parma*, 8:69–98, 2008.

[SHP18] Elia Santi, Christian Häger, and Henry D. Pfister. Decoding reed-muller codes using minimum-weight parity checks. *CoRR*, abs/1804.10319, 2018.

[van95] A. van den Bos. The multivariate complex normal distribution-a generalization. *IEEE Transactions on Information Theory*, 41(2):537–539, March 1995.

[VMB02] Martin Vetterli, Pina Marziliano, and Thierry Blu. Sampling signals with finite rate of innovation. 50(6):1417–1428, 2002.

[WIS+15] H. Wymeersch, N. V. Irukulapati, I. A. Sackey, P. Johannisson, and E. Agrell. Backward particle message passing. In *2015 IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 450–454, June 2015.

[WNF09a] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, July 2009.

[WNF09b] Stephen J. Wright, Robert D. Nowak, and M??rio A T Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

[YA19] Min Ye and Emmanuel Abbe. Recursive projection-aggregation decoding of reed-muller codes. *CoRR*, abs/1902.01470, 2019.

[YCF02] J.S. Yedidia, J. Chen, and M. Fossorier. Generating code representations suitable for belief propagation decoding. In *Annual Allerton Conference on Communications, Control and Computing*, October 2002.

[ZH08] Q. Zhang and M. I. Hayee. Symmetrized split-step fourier scheme to control global simulation accuracy in fiber-optic communication systems. *Journal of Lightwave Technology*, 26(2):302–316, Jan 2008.

[Zha05] Fuzhen Zhang. *The Schur Complement and its Applications*, volume 4 of *Numerical Methods and Algorithms*. Springer, New York, 2005.

# Biography

Mengke Lian received a B.S. in Astronautic Automation from Harbin Institute of Technology in July 2012, an M.S. in Electrical and Compute Engineering from Texas A&M University in 2014. He graduated with a Doctor of Philosophy in December 2019 from Duke University.

He had internships at AT&T Lab, New Jersey in summer 2018 and SK Hynix Memory Solutions America in autumn 2019, and he will join Google as a machine learning engineer at San Francisco Bay Area. His published papers include [LP15], [LHP19] and [LCHP19]. He graduated with his Ph.D. in telecommunication, information theory and machine learning under the supervision of Professor Henry Pfister.