2022

Unitary Approximate Message Passing for Sparse Bayesian Learning and Bilinear Recovery

Man Luo

UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Unitary Approximate Message Passing for Sparse Bayesian Learning and Bilinear Recovery

Man Luo

This thesis is presented as part of the requirements for the conferral of the degree:

Doctor of Philosophy

Supervisors:
A/Prof. Qinghua Guo

Co-supervisor:
Prof. Jiangtao Xi, Dr. Jun Tong

The University of Wollongong
School of Electrical, Computer and Telecommunications Engineering

01/21/2022

This research has been conducted with the support of an Australian Government Research Training Program Scholarship.

Declaration

I, *Man Luo*, declare that this thesis is submitted in partial fulfilment of the requirements for the conferral of the degree *Doctor of Philosophy*, from the University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualifications at any other academic institution.

Man Luo

June 16, 2022

Abstract

Over the past several years, the approximate message passing (AMP) algorithm has been applied to a broad range of problems, including compressed sensing (CS), robust regression, Bayesian estimation, etc. AMP was originally developed for compressed sensing based on the loopy belief propagation (BP). Compared to convex optimization based algorithms, AMP has low complexity and its performance can be rigorously characterized by a scalar state evolution (SE) in the case of a large independent and identically distributed (i.i.d.) (sub-) Gaussian matrix. AMP was then extended to solve general estimation problems with a generalized linear observation model. However, AMP performs poorly on a generic matrix such as non-zero mean, rank-deficient, correlated, or ill-conditioned matrix, resulting in divergence and degraded performance. It was discovered later that applying AMP to a unitary transform of the original model can remarkably enhance the robustness to difficult matrices. This variant is named unitary AMP (UAMP), or formally called UTAMP. In this thesis, leveraging UAMP, we propose UAMP-SBL for sparse signal recovery and Bi-UAMP for bilinear recovery, both of which inherit the low complexity and robustness of UAMP.

Sparse Bayesian learning (SBL) is a powerful tool for recovering a sparse signal from noisy measurements, which finds numerous applications in various areas. As a traditional implementation of SBL, e.g., Tipping's method, involves matrix inversion in each iteration, the computational complexity can be prohibitive for large scale problems. To circumvent this, AMP and its variants have been used as low-complexity solutions. Unfortunately, they will diverge for 'difficult' measurement matrices as previously mentioned. In this thesis, leveraging UAMP, a novel SBL algorithm called UAMP-SBL is proposed where UAMP is incorporated into the structured variational message passing (SVMP) to

handle the most computationally intensive part of message computations. It is shown that, compared to state-of-the-art AMP based SBL algorithms, the proposed UAMP-SBL is more robust and efficient, leading to remarkably better performance.

The bilinear recovery problem has many applications such as dictionary learning, self-calibration, compressed sensing with matrix uncertainty, etc. Compared to existing non-message passing alternates, several AMP based algorithms have been developed to solve bilinear problems. By using UAMP, a more robust and faster approximate inference algorithm for bilinear recovery is proposed in this thesis, which is called Bi-UAMP. With the lifting approach, the original bilinear problem is reformulated as a linear one. Then, variational inference (VI), expectation propagation (EP) and BP are combined with UAMP to implement the approximate inference algorithm Bi-UAMP, where UAMP is adopted for the most computationally intensive part. It is shown that, compared to state-of-the-art bilinear recovery algorithms, the proposed Bi-UAMP is much more robust and faster, and delivers significantly better performance.

Recently, UAMP has also been employed for many other applications such as inverse synthetic aperture radar (ISAR) imaging, low-complexity direction of arrival (DOA) estimation, iterative detection for orthogonal time frequency space modulation (OTFS), channel estimation for RIS-Aided MIMO communications, etc. Promising performance was achieved in all of the applications, and more applications of UAMP are expected in the future.

Acknowledgments

I would like to take this opportunity to express my thanks to all the people who helped me during my PhD studies in various aspects.

Firstly, I would like to express my deepest gratitude to my principle supervisor, A/Prof. Qinghua Guo, for his patience, encouragement and guidance throughout my PhD career. His insights for research always inspired me and help me to well address the issues I met. Without his invaluable suggestions and endless support, I could not have completed my thesis. My grateful thanks also go to my co-supervisors, Dr. Jun Tong and Prof. Jiangtao Xi for their helpful suggestions and encouragements.

My sincere thanks also goes to Prof. David Huang, my co-supervisor during the academic visiting to the University of Western Australia, who is a person with a wide range of knowledge and insights across a plenty variety of academic areas. It is a pleasant experience to visit Prof. Huang's group for I can always harvest great inspirations and constructive suggestions.

I wish to express my thanks go to my group-mates, Dawei Gao, Weikang Zhao, Yiwen Mao, Yuanyuan Zhang, Yuewen Zhu and other PhD colleagues.

Last but not the least, I would like to give a big hug to my beloved husband Dr. Yuanjun Zhao. He keeps supporting me to adventure in the complex world with all his patience, encouragement and love throughout my research period. I will place him like a seal over my heart for the rest of my life and been happy.

Author Publications

1. **M. Luo**, Q. Guo, M. Jin, Y. C. Eldar, D. Huang and X. Meng, "Unitary Approximate Message Passing for Sparse Bayesian Learning," *IEEE Transactions on Signal Processing*, vol. 69, pp. 6023-6039, 2021, doi: 10.1109/TSP.2021.3114985.

2. **M. Luo**, Q. Guo, D. Huang and J. Xi, "Sparse Bayesian Learning Based on Approximate Message Passing with Unitary Transformation," *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*, 2019, pp. 1-5, doi: 10.1109/VTS-APWCS.2019.8851644.

3. Y. Mao, **M. Luo**, D. Gao and Q. Guo, "Low complexity DOA estimation using AMP with unitary transformation and iterative refinement," *Digital Signal Processing*, vol. 106, p. 102800, 2020.

4. Z. Yuan, Q. Guo and **M. Luo**, "Approximate Message Passing With Unitary Transformation for Robust Bilinear Recovery," *IEEE Transactions on Signal Processing*, vol. 69, pp. 617-630, 2021, doi: 10.1109/TSP.2020.3044847.

List of Abbreviations

AMP	Approximate message passing
BAd-VAMP	bilinear adaptive VAMP
Bi-UAMP	bilinear UAMP
BP	Belief propagation
CS	Compressed sensing
EEG	Electroencephalography
EM	Expectation maximization
EP	Expectation Propagation
GAMP	Generalized approximate message passing
GGAMP	Gaussian generalized AMP
GSM	Gaussian scale mixture
KL	Kullback-Leibler
MAP	Maximum a posterior
MMSE	Minimum mean-squared error
MMV	Multiple measurement vector
MRI	Magnetic Resonance Imaging
MSE	Mean squared error
OAMP	Orthogonal approximate message passing
PC-VAMP	Perturbation considered vector approximate message passing
SBL	Sparse Bayesian learning
SE	State evolution
SMV	Single measurement vector
SNR	Signal to noise power ratio

SSR	Sparse signal reconstruction
SVD	Singular value decomposition
SVI	Structured Variational Inference
SVMP	Structured variational message passing
UAMP	AMP with unitary transformation
VAMP	Vector approximate message passing
VI	Variational Inference

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Background

The sparse signal recovery (SSR) problem has many applications such as the radar detection [1], the direction-of-arrival (DOA) estimation [2], the magnetic resonance imaging (MRI) [3] and the electroencephalography (EEG)/magnetoencephalography (MEG) source localization [4]. Among approaches used to recover sparse signals, sparse Bayesian learning has drawn much attention due to its ability to handle challenging problems, such as highly under-determined inverse problems and recovering signals with few sparsity [5].

For an SBL problem, the model can be written as:

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{w}, \tag{1.1}$$

where $\mathbf{y}$ is an observation vector with length M, $\mathbf{A}$ is a known matrix with size $M \times N$ ($M \leq N$) and $\mathbf{x}$ is a length-N sparse vector to be recovered. $\mathbf{w}$ denotes a Gaussian noise vector with mean zero and covariance matrix $\beta^{-1}\mathbf{I}$. The above problem can be extended to a multiple measurement vector (MMV) problem by replacing vectors $\mathbf{y}$, $\mathbf{x}$ and $\mathbf{w}$ with relevant matrices. SBL was first proposed in the context of machine learning in 2001 by Tipping [6] and it was adapted to be used for SSR in 2004 [7]. For the problem of sparse signal recovery, the traditional SBL algorithm can produce accurate estimates. However, the traditional implementation of SBL depends on computing multiple matrix inversions at each iteration. These matrix inversions in SBL are computationally expensive, which

limits the applicability of SBL algorithms to large scale problems. To address this issue, focusing on the expectation maximization (EM) based SBL algorithm in [6], the implementation of the E-step using AMP has been investigated. For instance, Gaussian generalized AMP (GGAMP) was used in [8] to implement the E-Step, where sufficient damping is used to enhance the robustness of the algorithm against a generic measurement matrix. This leads to the GGAMP-SBL algorithm with complexity significantly lower than that of SBL. However, the use of damping slows the convergence of the algorithm. Furthermore, we have observed that GGAMP-SBL still exhibits significant performance loss in the case of measurement matrices with relatively high correlation, condition number, or non-zero mean.

This thesis also aims to address the bilinear recovery problem, which is involved in many research areas like dictionary learning [9], self-calibration [10], compressed sensing with matrix uncertainty [11], etc. The bilinear recovery problem can be formulated that when known $\mathbf{A}_k$ are given, $\mathbf{b}$ and sparse $\mathbf{C}$ are to be estimated. The model is given by

$$\mathbf{Y} = \sum_{k=1}^{K} b_k \mathbf{A}_k \mathbf{C} + \mathbf{W}, \tag{1.2}$$

where $\{b_k\}$ and $\mathbf{C}$ are jointly recovered with known $\mathbf{A}_k$ from the noisy measurements $\mathbf{Y}$. Many algorithms have been developed to solve the bilinear problem. Some solve a convex relaxation of the original problem, while others adopt non-convex formulations via alternating methods [12], greedy methods, variational methods, message-passing methods, and other techniques [13]. Recently, several AMP based algorithms have been developed to address the bilinear problem, which show promising performance, compared to these non-message passing alternates [14]. The GAMP [15] was extended to bilinear GAMP (BiGAMP) [16] and then the parametric BiGAMP (P-BiGAMP) [17]. Lifted AMP was proposed in [18] by using the lifting approach [10, 19]. However, these AMP based algorithms are vulnerable to difficult $\mathbf{A}$ matrices, e.g., ill-conditioned, correlated, rank-deficient or non-zero mean matrices, as AMP can easily diverge in these cases [20]. Most recently, several algorithms based on vector AMP (VAMP) aimed to address this divergence issue [21]. Several VAMP based algorithms have been proposed, such as the lifted VAMP in [22], the bilinear adaptive VAMP (BAd-VAMP) in [14] and PC-VAMP in [23].

However, VAMP involves the calculations of two "extrinsic" precisions [14], which can be negative. These VAMP based algorithms are all badly affected by the VAMP-related instability issue, especially in the case of tough measurement matrices or some special priors.

1.2 Research Motivations

Unitary approximate message passing, called UAMP for convenience, was inspired by the work in [24], which can be regarded as the first application of UAMP to turbo equalization, where the normalized discrete Fourier transform matrix is used for unitary transformation. UAMP was proposed in 2015 [25]. It was discovered that the AMP algorithm can still work well for difficult $\mathbf{A}$ [20]. Instead of employing the original model (1.1), AMP is applied to a unitary transform of (1.1) in UAMP. As any matrix $\mathbf{A}$ has a singular value decomposition (SVD) $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}$, a unitary transformation with $\mathbf{U}^H$ can be performed, yielding

$$\mathbf{r} = \mathbf{\Phi}\mathbf{x} + \boldsymbol{\omega}, \tag{1.3}$$

where $\mathbf{r} = \mathbf{U}^H\mathbf{y}$, $\mathbf{\Phi} = \mathbf{U}^H\mathbf{A} = \mathbf{\Lambda}\mathbf{V}$ and $\mathbf{\Lambda}$ is an $M \times N$ ($M \leq N$) rectangular diagonal matrix. $\boldsymbol{\omega} = \mathbf{U}^H\mathbf{w}$ is still a zero-mean Gaussian noise vector with the same covariance matrix $\beta^{-1}\mathbf{I}$ and $\mathbf{U}^H$ is a unitary matrix. It is noted that in the case of a circulant matrix $\mathbf{A}$, e.g., in frequency domain equalization, the matrix for unitary transformation can be simply the normalized discrete Fourier transform matrix, which allows a more efficient implementation of the UAMP algorithm [24]. It is interesting that, with such a simple preprocessing, the robustness of AMP is remarkably enhanced, enabling it to handle difficult matrix $\mathbf{A}$.

UAMP is recently employed for inverse synthetic aperture radar (ISAR) imaging [26]. Real data experiments show its excellent capability of achieving high Doppler resolution with low complexity, where the measurement matrix can be highly correlated to achieve high Doppler resolution. The application of UAMP to low complexity direction of arrival estimation is also studied in [27]. UAMP has also been employed for iterative detection for orthogonal time frequency space modulation (OTFS) [28], which shows promising

performance.

VAMP and orthogonal AMP (OAMP) [29] are different to the UAMP algorithm and either of them consists of an LMMSE estimator and an MMSE denoiser. As the LMMSE estimator requires cubic complexity per iteration, which can be very high in applications with large-scale matrices, the LMMSE estimator is implemented with the aid of SVD. Two explicit expectation propagation (EP) operations are carried out between the iterations of the two modules. These two explicit EP operations lead to the computations of "extrinsic" precisions. The explicit computations of two "extrinsic" precisions are required. A problem is that the two gamma parameters can become negative in the iteration. However, these gamma parameters are essentially precisions and they should be positive. Although some remedies have been proposed, e.g., simply taking the absolute value of the calculated gamma, the treatments are heuristic without theoretical support. By contrast, there is no such problem in UAMP.

Promising performance and high robustness of UAMP in applications bring the motivation of designing efficient and robust sparse signal recovery algorithms in this thesis. The UAMP algorithm is applied for sparse Bayesian learning problems, resulting in a novel SBL algorithm called UAMP-SBL. Moreover, leveraging UAMP, an approximate inference algorithm for bilinear recovery is also proposed, which is called Bi-UAMP. Both of these algorithms can achieve more efficient signal recovery with significantly enhanced robustness, high performance and reduced complexity, compared to other state-of-the-art algorithms.

1.3 Research Contributions

In this thesis, leveraging UAMP algorithm, more efficient and robust algorithms for SBL and bilinear recovery have been proposed. To be more specific:

- In the first work, the empirical SE-based performance prediction for UAMP is investigated. The SE equation derived is with a high efficiency while the presentation is not complicated to understand. By using the empirical SE of UAMP, how to predict the performance of UAMP-SBL empirically is also investigated. The UAMP-

SBL is treated as UAMP with a special denoiser, enabling the use of UAMP-SE to predict the performance of UAMP-SBL.

- In the second work, a new SBL algorithm based on structured variational inference is proposed, leveraging AMP with a unitary transformation (UAMP-SBL). A Gamma distribution is employed as the hyperprior and the shape parameter of the Gamma distribution is tuned automatically during iterations. With these improvement, the proposed UAMP-SBL algorithm can approach the support-oracle bound closely in many cases with a generic measurement matrix. The impact of the shape parameter on SBL is also analyzed. Moreover, the proposed algorithm is extended from SMV problems to MMV problems. Complete comparison experiments are offered to clearly demonstrate the advantage of the proposed UAMP-SBL algorithm to other state-of-the-art ones.

- In the third work, a new approximate Bayesian inference algorithm is proposed for bilinear recovery and named as Bi-UAMP. This new approximate inference algorithm is designed by integrating the UAMP algorithm with BP, VI and EP to achieve efficient approximate inference. This proposed algorithm is also extended from SMV problems to MMV problems. From numerous experimental results, it is proved that the proposed Bi-UAMP is much more robust and faster than the other state-of-the-art algorithms.

1.4 Thesis Organization

The rest of this thesis is organized as follows, which is shown by the diagram in Figure 1.1. In Chapter 2, a literature review of current research is presented. Firstly AMP is introduced and various AMP-based techniques designed for solving the signal recovery problem are described. Then, the conventional SBL algorithm is explained in detail. Since the traditional implementation of SBL uses matrix inversions at each iteration, its complexity is too high for large-scale problems. AMP and its variants used for the low complexity implementation of SBL are reviewed. Finally, the bilinear recovery problem is presented. Since AMP based algorithms are vulnerable to difficult $\mathbf{A}$ matrices. To

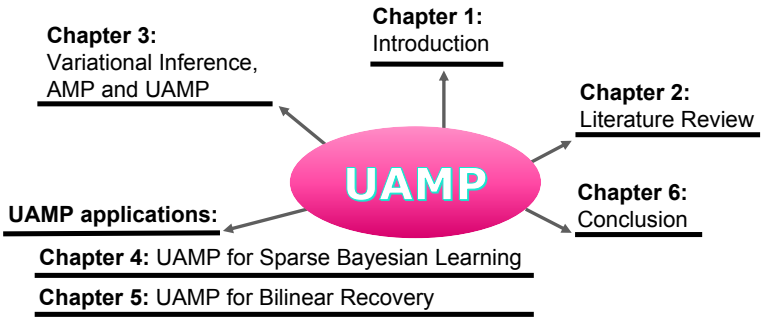

Figure 1.1: Summary diagram of the connections of sections in the thesis.

achieve robust bilinear recovery, conventional non-message passing based algorithms and message passing based algorithms are discussed. In Chapter 3, to provide a comprehensive insight of UAMP, a brief introduction of variational inference and AMP algorithm is given. Then, two versions of UAMP algorithm are described. The analysis of empirical state evolution used to predict the performance of UAMP are also detailed. In Chapter 4, a low-complexity and high-robust algorithm for sparse Bayesian learning is developed. Firstly, the SBL algorithm is briefly introduced. Then the UAMP-SBL algorithm is proposed, which is followed by the investigation of the SE-based performance prediction. After that, the impact of shape parameter is analyzed and the extended MMV settings are described. Numerical results are then given with corresponding performance discussion. In Chapter 5, the Bi-UAMP algorithm is designed and introduced. The extension for MMV problems and investigation of the algorithm properties are given. Experimental results and comparison discussion with other state-of-the-art message passing and non-message passing algorithms are also provided. The thesis is concluded in Chapter 6 and several inspiring future works are given too.

In this thesis, boldface lowercase and uppercase letters apply to represent column vectors and matrices, respectively. The superscript $(\cdot)^H$ represents the conjugate transpose for a complex matrix, and the transpose for a real matrix. The notation $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a Gaussian distribution of $\mathbf{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, and $\mathrm{Ga}(\boldsymbol{\gamma}|\epsilon, \eta)$ is a Gamma distribution with shape parameter ϵ and rate parameter η. Notation $\otimes$ represents the Kronecker product. The relation $f(x) = cg(x)$ for some positive constant c is written as $f(x) \propto g(x)$. The notation $\langle f(\mathbf{x})\rangle_{q(\mathbf{x})}$ denotes the expectation of $f(\mathbf{x})$ with respect to probability density function $q(\mathbf{x})$, and $\mathbb{E}[\cdot]$ is the expectation over all random variables

involved in the brackets. $Diag(\mathbf{a})$ is used to represent a diagonal matrix with elements of $\mathbf{a}$ on its diagonal, $Z_{m,n}$ is the (m,n)th element of $\mathbf{Z}$, and a_n is the nth element of vector $\mathbf{a}$. $(\mathbf{B})_D$ returns a diagonal matrix by forcing the off-diagonal elements of $\mathbf{B}$ to zero. The element-wise product and division of two vectors $\mathbf{a}$ and $\mathbf{b}$ are denoted by $\mathbf{a} \cdot \mathbf{b}$ and $\mathbf{a}./\mathbf{b}$, respectively. The superscript of $\mathbf{a}^t$ in an iterative algorithm denotes the tth iteration. The notation $\mathbf{a}^{\cdot -1}$ denotes the element-wise inverse operation to vector $\mathbf{a}$. $|\mathbf{A}|^2$ is used to denote element-wise magnitude squared operation for $\mathbf{A}$, and use $||\mathbf{a}||^2$ to denote the squared l_2 norm of $\mathbf{a}$. The notation $< \mathbf{a} >$ denotes the average operation for $\mathbf{a}$, i.e., the sum of the elements of $\mathbf{a}$ divided by its length. The notation $\int_{\mathbf{c} \vee c_n} f_{\mathbf{c}}(\mathbf{c})$ represents integral over all elements in $\mathbf{c}$ except c_n. $\mathbf{1}$ and $\mathbf{0}$ is used to denote an all-one vector and an all-zero vector with a proper length, respectively. Sometimes, a subscript n for $\mathbf{1}$, i.e., $\mathbf{1}_n$ is used to indicate its length n.

Chapter 2

Literature Review

2.1 Approximate Message Passing

Approximate message passing is an efficient approach to the estimation of signal vector $\mathbf{x}$ in the following model

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

where $\mathbf{y}$ is a measurement vector of length M and the measurement matrix $\mathbf{A}$ has a size of $M \times N$. $\mathbf{w}$ denotes a Gaussian noise vector with mean zero and covariance matrix $\beta^{-1}\mathbf{I}$, and β is the precision of the noise. It is assumed that the elements of $\mathbf{x}$ are i.i.d, i.e., $p(\mathbf{x}) = \prod_n p(x_n)$.

Originally, based on loopy belief propagation (BP) [30], AMP algorithms are proposed for the problem of compressed sensing. The advantages of AMP algorithms help to facilitate its utilise in the case of a large independent and identically distributed (i.i.d.) (sub-)Gaussian matrix $\mathbf{A}$ [31]. Firstly, AMP has a lower complexity than convex optimization based algorithms such as LASSO [32] and greedy algorithms such as iterative hard-thresholding [33]. Then, its per-iteration behavior is rigorously characterized via the state evolution (SE) and the SE equation converges to a fixed point which is unique. Thus, AMP is Bayes-optimal [34]. In [15] and [35], AMP was extended for addressing general estimation problems with a generalized linear observation model. A significant reduction in computational complexity can be achieved by implementing

the E-step with AMP in the expectation maximization based SBL method. Based on the original AMP, a generalized AMP (GAMP) algorithm was proposed in [15]. The GAMP was used to accommodate more general distribution $p(y_i|(\mathbf{Ax})_i)$ which may not be Gaussian (where y_i and $(\mathbf{Ax})_i$ denote the i-th element in $\mathbf{y}$ and $(\mathbf{Ax})$, respectively). The densities $p(\mathbf{x})$ and $p(\mathbf{y}|\mathbf{Ax})$ can be used to compute the maximum a posterior estimate $\hat{\mathbf{x}}_{MAP} = \arg\min_{\mathbf{x}} p(\mathbf{x}|\mathbf{y})$ in its max-sum mode, or approximate the minimum mean-squared error estimate $\hat{\mathbf{x}}_{MMSE} = \int \mathbf{x} p(\mathbf{x}|\mathbf{y}) d\mathbf{x} = \mathbb{E}(\mathbf{x}|\mathbf{y})$ in its sum-product mode [8].

However, AMP cannot work well for a generic matrix such as non-zero mean, rank-deficient, correlated, or ill-conditioned matrix $\mathbf{A}$ [20], resulting in divergence and poor performance. Many variants to AMP have been proposed to address the divergence issue and achieve better robustness to a generic $\mathbf{A}$, such as the damped AMP [20], swept AMP [36] and GAMP with adaptive damping [37]. The swept AMP replaces parallel variable updated in the GAMP algorithm with serial ones to enhance convergence. However, it is relatively slow and still diverges for certain matrix $\mathbf{A}$. The same issue also happens in the work of GAMP with adaptive damping.

More effective variants include UAMP [25] proposed in 2015, vector AMP [21] (VAMP) in 2016, orthogonal AMP (OAMP) [29] in 2016, memory AMP (MAMP) [38] in 2020, convolutional AMP (CAMP) [39] in 2020 and so on. In detail, OAMP was proposed in [29] for general unitarily-invariant matrices, including independent identically distributed Gaussian matrices and partial orthogonal matrices [29]. OAMP involves two local processors which are a linear estimator and a non-linear estimator under certain orthogonality constraints, i.e., the input and output estimation errors of each processor are orthogonal. OAMP is related to a variant of the expectation propagation algorithm [40]. In OAMP, the Onsager term in AMP vanishes as a result of the divergence-free constraint on non-linear estimator [29]. VAMP is equivalent to OAMP and is also regarded as an exact large system approximation of EP. Although they can solve the divergence problem, they only handle right-orthogonally invariant matrices $\mathbf{A}$. However, in most practical applications, matrices used are not belonged to the family of right-orthogonally invariant matrices, such as non-zero mean matrices. Then, VAMP based algorithms consist of an LMMSE estimator and an MMSE denoiser, and two explicit EP operations are carried out between the

iterations of the two modules. The two explicit EP operations cause the computations of "extrinsic" precisions. As the LMMSE estimator requires cubic complexity per iteration, which can be very prohibitive in applications with large-scale matrices, the LMMSE estimator is implemented with the aid of SVD. The explicit computations of two "extrinsic" precisions are required. A problem is that the two gamma parameters can become negative in the iteration, however, these gamma parameters are essentially precisions and they should be positive. Although some remedies have been proposed, e.g., simply taking the absolute value of the calculated gamma, the treatments are heuristic without theoretical support. In contrast, there is no such problem in UAMP. In particular, UAMP was derived based on a unitary transform of model (1.1), and it converges for any matrix $\mathbf{A}$ in the case of Gaussian prior [25]. This can be proved in a later chapter that Bi-UAMP performs significantly better and is much faster than BAd-VAMP and PC-VAMP for difficult matrices. In other words, UAMP can outperform VAMP in terms of bilinear recovery.

CAMP improves AMP by replacing the Onsager correction term by a convolution of all preceding messages [39]. However, CAMP does not work well for matrix $\mathbf{A}$ with high condition numbers [41], resulting in divergence. To solve the convergence issue in CAMP with high condition numbers [41], memory AMP has been proposed in [38]. In addition, CAMP has a relatively low convergence speed and for some matrices it may fail to converge. Thus, damping factor is used to improve the convergence. Unfortunately, the damping is performed on the non-linear estimator outputs, which breaks the asymptotic Gaussianity of estimation error [42].

Moreover, the CAMP suffers from the problem of a low convergence speed. What is worse, for matrices with high condition numbers, CAMP may fail to converge [38]. In addition of that, the heuristic damping used to improve the convergence of CAMP is performed on the a-posteriori outputs. This can break orthogonality and the asymptotic Gaussianity of estimation errors. As an improvement of CAMP, relaxation parameters and a damping vector are analytically optimized in MAMP based on state evolution. The damping vector used here can preserve the orthogonality and hence the convergence of MAMP is guaranteed and improved. However, CAMP and MAMP algorithms were motivated by SE analysis for right-rotationally invariant sensing matrices. Actually, it is

hard to find right-rotationally invariant matrices in practical engineering applications. For other matrices which are not right-rotationally invariant, e.g. non-zero mean matrix, it is easy for CAMP and MAMP to diverge. Compared with them, damping is not required in UAMP to converge on various difficult matrices. Thus, UAMP is more robust and faster than these state-of-the-art algorithms.

2.2 Sparse Bayesian Learning Algorithm

Consider recovering a length-N sparse vector $\mathbf{x}$ from measurements

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{w}, \tag{2.2}$$

where $\mathbf{y}$ is a measurement vector of length M, the measurement matrix $\mathbf{A}$ has size $M \times N$, and $\mathbf{x}$ is a length-N sparse vector to be recovered. $\mathbf{w}$ denotes a Gaussian noise vector with mean zero mean and covariance matrix $\beta^{-1}\mathbf{I}$.

To obtain the sparsest solution of the above equation, one can obtain a sparse estimation of $\mathbf{x}$ via Bayesian approaches. A popular method is the SBL. SBL was first proposed in 2001 [6], and it was adapted to be used for sparse signal recover in 2004 [7]. Hereby we give a detailed introduction of SBL. Essentially, SBL is a type II Bayesian approach. In SBL, the prior is a Gaussian scale mixture (GSM) [43, 44] on $\mathbf{x}$ and it is Gaussian conditioned on a precision vector γ. The prior of x_n is shown:

$$p(x_n) = \int \mathcal{N}(x_n|0, \gamma_n^{-1}) p(\gamma_n) d\gamma_n, \tag{2.3}$$

where the precision vector $\gamma = [\gamma_1, \gamma_2, ..., \gamma_N]^H$.

The EM-based SBL approach in [6] is used for sparse signal recovery in [7]. It has two steps in each iteration: the E-step and M-step. In the E-step, the a posterior probability $p(\mathbf{x}|\mathbf{y}, \hat{\boldsymbol{\gamma}})$ is computed, where $\hat{\boldsymbol{\gamma}} = [\hat{\gamma}_1, ..., \hat{\gamma}_N]^T$ is the learned precision vector in the last iteration and $\mathbf{y}$ is a measurement vector. The a posterior probability turns out to be Gaussian, i.e., $p(\mathbf{x}|\mathbf{y}, \hat{\boldsymbol{\gamma}}) = \mathcal{N}(\mathbf{x}|\hat{\mathbf{x}}, \mathbf{Z})$. The M-step is used to update the precisions. The EM-based SBL algorithm (which is called SBL hereafter) executes the following

iteration:

Repeat

$$\mathbf{Z} = \left(\beta \mathbf{A}^H \mathbf{A} + Diag(\hat{\boldsymbol{\gamma}})\right)^{-1} \tag{2.4}$$

$$\hat{\mathbf{x}} = \beta \mathbf{Z} \mathbf{A}^H \mathbf{y} \tag{2.5}$$

$$\hat{\gamma}_n = (2\epsilon + 1)/(2\eta + |\hat{x}_n|^2 + Z_{n,n}), n = 1, ..., N \tag{2.6}$$

Until terminated

where $Diag(\hat{\boldsymbol{\gamma}})$ is a diagonal matrix, with the elements of $\hat{\boldsymbol{\gamma}}$ as its diagonal, and $Z_{n,n}$ denotes the (n, n)th element of $\mathbf{Z}$.

Compared with other signal recovery algorithms, such as l_0-norm based optimization and l_1-norm based optimization, SBL can provide more information of sparse vector $\mathbf{x}$. The estimation of $\mathbf{x}$ is via its a posterior mean and the accuracy of the estimator is via its a posterior covariance matrix. However, the EM-based SBL has a high computational complexity of $O(\{M^2 N\})$. In detail, the E-step of the SBL algorithm requires a matrix inverse in (2.4) in each iteration. This results in cubic complexity in each iteration, which can be prohibitive for large scale problems. Due to the high complexity of the SBL, it is limited to imposing GSM priors. To address this issue, the implementation of the E-step using AMP has been explored in some works. GGAMP was used in [8] to implement the E-step, where sufficient damping is used to enhance the robustness of the algorithm against difficult measurement matrices. This leads to the GGAMP-SBL algorithm, of which the complexity is significantly lower than that of the conventional SBL algorithm. However, the use of damping slows the convergence of the algorithm. Furthermore, it is observed that GGAMP-SBL still exhibits significant performance loss in the case of measurement matrices with relatively high correlation, condition number, or non-zero mean. In this thesis, the challenging problem of high complexity of the SBL is successfully solved by adopting UAMP and a much more robust and efficient solution is proposed.

2.3 Bilinear Recovery

In this section, we consider the following bilinear problem

$$\mathbf{Y} = \sum_{k=1}^{K} b_k \mathbf{A}_k \mathbf{C} + \mathbf{W}, \tag{2.7}$$

where $\mathbf{Y}$ denotes measurements and matrices $\{\mathbf{A}_k\}$ are known. $\{b_k\}$ and $\mathbf{C}$ are to be recovered. $\mathbf{W}$ represents white Gaussian noise. When $\mathbf{Y}$, $\mathbf{C}$ and $\mathbf{W}$ are replaced with the corresponding vectors $\mathbf{y}$, $\mathbf{c}$ and $\mathbf{w}$, respectively, the above MMV problem is reduced to a SMV problem. Model (2.7) covers a variety of problems, e.g., compressed sensing with matrix uncertainty [11], joint channel estimation and detection [45], self-calibration and blind deconvolution [10], and structured dictionary learning [9]. Specifically when $\mathbf{c}$ is sparse, the problem is known as compressed sensing with matrix uncertainty [11]. Another example is that when $\mathbf{Y} = \mathbf{A}\mathbf{c} + \mathbf{W}$ with sparse $\mathbf{c}$ and structured $\mathbf{A} = \sum_k b_k \mathbf{A}_k$, the problem of estimating $\mathbf{c}$ and $\mathbf{A}$ is known as structured dictionary learning [9]. This model is also applicable to self-calibration and other circumstances.

There has been extensive research on this active field in the past few years, including the non-message passing methods and message passing methods. For non-message passing based algorithms, the performance of the Weighted and Structured Sparsity Cognizant Total Least Squares (WSS-TLS) from the award-winning paper [11] was significantly worse than the AMP approaches. Recently, motivated by the AMP algorithm, extension of the AMP to handle bilinear problem has been considered. The GAMP algorithm [15] was extended to bilinear GAMP (BiGAMP) [16] for solving a general bilinear problem. The parametric BiGAMP (P-BiGAMP) was then proposed in [17], which works with model (2.7) to jointly recover $\{b_k\}$ and $\mathbf{C}$. The BiGAMP algorithm is a special case of the P-BiGAMP algorithm. However, the evidence in [22] indicated that, like AMP, the P-BiGAMP algorithm is sensitive to deviations from the i.i.d. assumptions used in its derivation and analysis [17]. More recently, AMP methods for bilinear inference were proposed using the "lifting" approach [10, 19]. Lifted AMP was proposed in [18]. A rigorous analysis of "lifted AMP" was presented in [46]. However, lifted AMP inherits all the AMP-related convergence issues. That is, these AMP based algorithms are vulnerable

to difficult **A** matrices, e.g., ill-conditioned, correlated, rank-deficient or non-zero mean matrices as AMP can easily diverge in these cases [21].

The problem with AMP is that its behavior is understood only in the case of large or infinitely large, i.i.d. (sub) Gaussian A [34, 47]. Even the small deviation of the measurement matrices **A** from the i.i.d. zero mean Gaussian matrix can cause AMP to diverge. Most recently, to address this issue and accommodate a larger class of matrices **A**, many works have been done to extend VAMP [21] to deal with the bilinear recovery problem [14, 22]. The lifted VAMP was proposed in [22]. However, the computational complexity of lifted VAMP is high since the number of unknowns increases significantly, especially when the number of original variables is large [48]. To address this issue, the bilinear adaptive VAMP (BAd-VAMP) was proposed in [14], which also inherits the robustness of VAMP. It was shown that the BAd-VAMP algorithm is more robust and faster, and it can outperform lifted VAMP significantly [14]. Based on VAMP, PC-VAMP was proposed in [23] to achieve compressed sensing with structured matrix perturbation. In [48], BAd-VAMP was extended to incorporate arbitrary distributions on the output transform based on the framework in [35]. In BAd-VAMP, **c** is reconstructed by using the expectation-maximization algorithm. Then, **b** is reconstructed by implementing the VAMP algorithm to find the minimum mean-squared error estimate of **b**. However, due to the use of the EM algorithm, BAd-VAMP cannot apply to general priors on **b**. These variants to VAMP also inherit the instability of VAMP algorithm. VAMP involves the computations of two gamma parameters ("extrinsic" precision), these gamma parameters are essentially precisions and they should be positive. However, in some cases, they become negative in the iteration, especially in BAd-VAMP. This can also potentially lead to the instability of VAMP algorithm in the case of tough measurement matrix or some special priors.

Chapter 3

Variational Inference, AMP and UAMP Algorithms

3.1 Introduction

Consider recovering an unknown vector $\mathbf{x}$ from measurements

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{w} \tag{3.1}$$

where $\mathbf{y}$ is a measurement, $\mathbf{A}$ is a known $M \times N$ measurement matrix and $\mathbf{w}$ is a white Gaussian noise vector with distribution $\mathcal{N}(\mathbf{w}; \mathbf{0}, \beta^{-1}\mathbf{I})$. The unknown vector $\mathbf{x}$ which has a known prior density $p(x)$ and hence all the prior knowledge about $\mathbf{x}$ is assumed to be known. The approximate a posterior mean of $\mathbf{x}$ is used to serve as an estimate for $\mathbf{x}$ in the sense of minimum mean squared error. However, it is generally intractable for large scale problems. Thus, we resort to the variational inference (VI) which is one of the approximate inference techniques.

Variational techniques have been used for decades in quantum and statistical physics, where they are referred to as the mean field (MF) approximation [49]. Later, they found their way to the area of machine learning or statistical inference [50, 51]. The central idea of VI is to approximate the model posterior by a simpler distribution. To this end, one minimizes the Kullback-Leibler (KL) divergence between the posterior and the approximating distribution, which can be done in an iterative way. Instead of using fully fac-

torized trial functions where all variables are assumed to be independent (thereby likely resulting in poor approximations), more structured factorizations can be used. To develop a practical approximate inference algorithm, we follow the framework of structured variational inference (SVI). It is worth mentioning that SVI [52–54] has been formulated as nicely in terms of message passing in factor graphs, i.e., structured variational message passing (SVMP) [53].

The approximate message passing algorithm was developed based on the loopy BP for compressed sensing with model (3.1) [55]. AMP enjoys low complexity and its performance can be rigorously characterized by a scalar state evolution in the case of large i.i.d. (sub-)Gaussian $\mathbf{A}$ [31]. However, for a generic $\mathbf{A}$, the convergence of AMP cannot be guaranteed, e.g., AMP can easily diverge for non-zero mean, rank-deficient, correlated, or ill-conditioned matrix $\mathbf{A}$ [15].

In this chapter, a new variant of AMP based on a unitary transformation of the original model (hence the variant is called UTAMP or UAMP) is introduced, where the unitary matrix is available for any matrix A. Two versions of UAMP are represented. The difference between these two versions is two approximations, leading to matrix-vector products reducing from 4 to 2. This is very significant as the complexity of AMP-like algorithms is dominated by matrix-vector products. Interestingly, the two approximations also enhance the robustness of the algorithm.

In this chapter, how to employ the empirical SE to predict the performance for UAMP is introduced. The SE equation derived is with a high efficiency while the presentation is not complicated to understand. Since the UAMP-SBL is treated as UAMP with a special denoiser, it is also shown that the performance prediction of UAMP-SBL can be done by using the empirical SE of UAMP.

3.1.1 Chapter's Organization

The organization of the chapter is as follows. In section 3.2, a brief overview of the variational inference is given. In section 3.3, the definitions of the AMP algorithm are presented. In Section 3.4, the UAMP algorithm and its two versions are introduced. Empirical state evolution to predict the performance of UAMP algorithm is proposed.

3.2 Variational Inference

Variational inference, which has been widely used in Bayesian problems and especially for approximating posterior distributions, is defined as a type of machine learning algorithm for approximate inference [50, 52, 54].

Generally, when the posterior probability $p(\mathbf{x}|\mathbf{y})$ of a set of variables $\mathbf{x} = \{x_1, x_2, ..., x_m\}$ conditioned on $\mathbf{y}$ is to be computed, a set of observed variables $\mathbf{y} = \{y_1, y_2, ..., y_n\}$ are always given as known information. The posterior distribution is:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})} \tag{3.2}$$

To approximate the conditional distribution $p(\mathbf{x}|\mathbf{y})$, a surrogate distribution from a simpler family $q(\mathbf{x})$ is proposed. Inference on $q(\mathbf{x})$ will be much easier than on $p(\mathbf{x}|\mathbf{y})$ if a computationally friendly family of distribution is adopted.

In variational inference, by minimizing the KL divergence [56] to the true posterior function, a selected tractable trial distribution function can be optimized. The KL-divergence measures the differences between two probability distributions

$$\mathcal{KL}(P(x)||Q(x)) = -\int P(x) \log \frac{Q(x)}{P(x)} dx. \tag{3.3}$$

The KL-divergence is non-negative, while when the two distributions are identical it will be equal to zero.

Variational inference selects the surrogate model by minimizing the KL-divergence:

$$\tilde{q}(\mathbf{x}) = \arg\max_{q} \mathcal{KL}(q(\mathbf{x})||p(\mathbf{x}|\mathbf{y})), \tag{3.4}$$

where $\mathcal{KL}(q(\mathbf{x})||p(\mathbf{x}|\mathbf{y}))$ denotes the KL divergence from p to q, i.e.,

$$\mathcal{KL}(q(\mathbf{x})||p(\mathbf{x}|\mathbf{y})) = -\int q(\mathbf{x}) \log \frac{p(\mathbf{x}|\mathbf{y})}{q(\mathbf{x})} d\mathbf{x}. \tag{3.5}$$

Substitute $p(\mathbf{x}|\mathbf{y})$ into (3.5), the KL divergence is obtained as below:

$$\begin{aligned}
&\mathcal{KL}(q(\mathbf{x})||p(\mathbf{x}|\mathbf{y})) \\
&= \int q(\mathbf{x}) \log q(\mathbf{x}) d\mathbf{x} - \int q(\mathbf{x}) \log p(\mathbf{x}|\mathbf{y}) d\mathbf{x} \\
&= \mathbb{E}_{q(\mathbf{x})}[\log q(\mathbf{x})] - \mathbb{E}_{q(\mathbf{x})}[\log p(\mathbf{x}, \mathbf{y})] + \log p(\mathbf{y})
\end{aligned} \tag{3.6}$$

where $\log p(\mathbf{y})$ can be treated as a constant since it does not depend on q. This leads to an optimization problem over the evidence lower bound (ELBO).

$$ELBO(q) = \mathbb{E}_{q(\mathbf{x})}[\log p(\mathbf{x}, \mathbf{y})] - \mathbb{E}_{q(\mathbf{x})}[\log q(\mathbf{x})] \tag{3.7}$$

Minimizing the KL-divergence is equivalent to maximizing the ELBO. However, (3.6) is still difficult to compute directly for large scale problem. To overcome this, q needs to be constrained in a certain family of distribution. The mean field approximation has been used as one of the suitable family [57]. The surrogate distribution $q(\mathbf{x})$ is assumed to be a product of independent single variable factors in mean field approximation. Unfortunately, a poor approximation will be yielded when the target distribution is multi-modal or otherwise has more complicated correlations in mean field approximation.

Instead of using fully factorized trial functions where all variables are assumed to be independent, more structured factorizations can be used, leading to structured variational inference algorithms. With graphical models, SVI algorithms can be formulated as message passing [52–54], which is termed as structured variational message passing. In this thesis, SVMP is adopted and applied with the UAMP algorithm. It will be shown how UAMP can be used to handle the most computational intensive part of message computations, enabling the algorithm to achieve a low complexity and a high robustness.

3.3 AMP algorithm

The AMP algorithm is applied in compressed sensing to estimate an unknown vector $\mathbf{x}$ from linear measurement $\mathbf{y}$ obtained from (3.1) using separable denoiser η [55]. It means to act coordinate-wise when applied to a vector. Starting with $\mathbf{x}^0 = 0$, an all-zero vector,

AMP iterates as follows:

$$\mathbf{x}^{t+1} = \eta(\mathbf{A}^H\mathbf{z}^t + \mathbf{x}^t) \tag{3.8}$$

$$\mathbf{z}^t = \mathbf{y} - \mathbf{A}\mathbf{x}^t + \mathbf{z}^{t-1}\frac{1}{\delta}\left\langle \eta'(\mathbf{A}^H\mathbf{z}^{t-1} + \mathbf{x}^{t-1})\right\rangle, \tag{3.9}$$

where η' denotes the derivative of η and $\mathbf{A}^H$ denotes the transpose of $\mathbf{A}$. $\delta = \frac{M}{N}$. Under the assumption that $\mathbf{A}$ has i.i.d. sub-Gaussian entries, $\mathbf{x}$ has i.i.d. entries according to a probability distribution p_x. The last term of the equation (3.9) is referred to as the Onsager term [34].

In [34], the authors provide a heuristic deviation of AMP from the message passing algorithm which includes the following two iterative equations,

$$z^t_{a\to i} = y_a - \sum_{j\in[n]\setminus i} A_{aj}x^t_{j\to a}, \tag{3.10}$$

$$x^{t+1}_{i\to a} = \eta\left(\sum_{j\in[n]\setminus a} A_{aj}z^t_{j\to a}\right), \tag{3.11}$$

where subscript $a \to i$ in (3.10) represents the message from the factor node (contains information of observation) a to the variable node which contains information of signals. In (3.11), the subscript $i \to a$ represents the message from the variable node i to the factor node a. The subscript $[n] \setminus i$ denotes the set of $[n]$ but without element i. A direct calculation of MP based on (3.10) and (3.11) will not be practical especially for large dimension systems as it requires to update MN messages per iteration [55]. The computational complexity of MP is extremely high. On the other hand, it is easy to see that the right-hand side of (3.10) depends weakly on the index i and that the right-hand side of (3.11) depends weakly on a. A more careful analysis of this dependence leads to corrections of order one in the high-dimensional limit. Such corrections are however fully captured by the last term on the right hand side of (3.9), thus leading to the AMP algorithm [58].

The behavior of the AMP algorithms is predicted by a deterministic scalar recursion referred to as state evolution [30]. More specifically, with the initial condition $\tau_0^2 = \beta^{-1} +$

Algorithm 1 Vector Stepsize AMP

Initialize $\boldsymbol{\tau}_x^{(0)} > 0$ (with elements larger than 0) and $\mathbf{x}^{(0)}$. Set $\mathbf{s}^{(-1)} = \mathbf{0}$ and $t = 0$.

Repeat

1: $\boldsymbol{\tau}_p = |\mathbf{A}|^2 \boldsymbol{\tau}_x^t$
2: $\mathbf{p} = \mathbf{A}\mathbf{x}^{\mathbf{t}} - \boldsymbol{\tau}_p \cdot \mathbf{s}^{t-1}$
3: $\boldsymbol{\tau}_s = \mathbf{1}./(\boldsymbol{\tau}_p + \beta^{-1}\mathbf{1})$
4: $\mathbf{s}^t = \boldsymbol{\tau}_s \cdot (\mathbf{y} - \mathbf{p})$
5: $\mathbf{1}./\boldsymbol{\tau}_q = |\mathbf{A}^H|^2 \boldsymbol{\tau}_s$
6: $\mathbf{q} = \mathbf{x}^t + \boldsymbol{\tau}_q \cdot \mathbf{A}^H \mathbf{s}^t$
7: $\boldsymbol{\tau}_x^{t+1} = \boldsymbol{\tau}_q \cdot g_x'(\mathbf{q}, \boldsymbol{\tau}_q)$
8: $\mathbf{x}^{t+1} = g_x(\mathbf{q}, \boldsymbol{\tau}_q)$
9: $t = t + 1$

Until terminated

$\frac{N}{M} v_0^2$, the state of evolution includes two equations when $t \geq 1$:

$$\tau_t^2 = \beta^{-1} + \frac{N}{M} v_t^2 \tag{3.12}$$

$$v_{t+1}^2 = \mathbb{E}[(\eta(x + \tau_t z])^2] \tag{3.13}$$

where $x \sim p_x$ is independent of $z \sim \mathcal{N}(0, 1)$. The correctness of SE has been rigorously proved in [34]. The fixed points of state evolution describe the output of the corresponding AMP, when the latter is used for a sufficiently large number of iterations [30]. It is well known, within statistical mechanics, that the fixed point equations coincide with the equations obtained through a completely different non-rigorous approach, the replica method [59, 60].

3.4 UAMP algorithm

The UAMP algorithm, inspired by [24], was derived based on the vector stepsize AMP algorithm shown in **Algorithm** 1 and a unitary transform of model (3.1) [25]. In vector stepsize AMP and UAMP, the function $g_x(\mathbf{q}, \boldsymbol{\tau}_q)$ returns a column vector whose n-th element, denoted as $[g_x(\mathbf{q}, \boldsymbol{\tau}_q)]_n$, is given by

$$[g_x(\mathbf{q}, \boldsymbol{\tau}_q)]_n = \frac{\int x_n p(x_n) \mathcal{N}(x_n | q_n, \tau_{q_n}) dx_n}{\int p(x_n) \mathcal{N}(x_n | q_n, \tau_{q_n}) dx_n}. \tag{3.14}$$

Algorithm 2 UAMP Version 1

Unitary transform: $\mathbf{r} = \mathbf{U}^H\mathbf{y} = \mathbf{\Phi x} + \boldsymbol{\omega}$, where $\mathbf{\Phi} = \mathbf{U}^H\mathbf{A} = \mathbf{\Lambda V}$, and $\mathbf{U}$ is obtained from the SVD $\mathbf{A} = \mathbf{U\Lambda V}$.
Initialize $\boldsymbol{\tau}_x^{(0)} > 0$ and $\mathbf{x}^{(0)}$. Set $\mathbf{s}^{(-1)} = \mathbf{0}$ and $t = 0$.
Repeat

1: $\boldsymbol{\tau}_p = |\mathbf{\Phi}|^2\boldsymbol{\tau}_x^t$
2: $\mathbf{p} = \mathbf{\Phi}\boldsymbol{x}^t - \boldsymbol{\tau}_p \cdot \mathbf{s}^{t-1}$
3: $\boldsymbol{\tau}_s = \mathbf{1}./(\boldsymbol{\tau}_p + \beta^{-1}\mathbf{1})$
4: $\mathbf{s}^t = \boldsymbol{\tau}_s \cdot (\mathbf{r} - \mathbf{p})$
5: $\mathbf{1}./\boldsymbol{\tau}_q = |\mathbf{\Phi}^H|^2\boldsymbol{\tau}_s$
6: $\mathbf{q} = \mathbf{x}^t + \boldsymbol{\tau}_q \cdot (\mathbf{\Phi}^H\mathbf{s}^t)$
7: $\boldsymbol{\tau}_x^{t+1} = \boldsymbol{\tau}_q \cdot g_x'(\mathbf{q}, \boldsymbol{\tau}_q)$
8: $\mathbf{x}^{t+1} = g_x(\mathbf{q}, \boldsymbol{\tau}_q)$
9: $t = t + 1$

Until terminated

Equation (3.14) can be interpreted as the minimum mean square error (MMSE) estimation of x_n based on the following model

$$q_n = x_n + \varpi \tag{3.15}$$

where ϖ is a Gaussian noise with mean zero and variance τ_{q_n}.

The function $g_x'(\mathbf{q}, \boldsymbol{\tau}_q)$ returns a column vector and the n-th element is denoted by $[g_x'(\mathbf{q}, \boldsymbol{\tau}_q)]_n$, where the derivative is with respect to q_n. It is not hard to show that $\tau_{q_n}[g_x'(\mathbf{q}, \boldsymbol{\tau}_q)]_n$ is the a posterior variance of x_n with model (3.15). Note that $g_x(\mathbf{q}, \boldsymbol{\tau}_q)$ can also be changed for maximum a posterior estimation of $\mathbf{x}$.

The derivation of UAMP is briefly introduced in the following. As any matrix $\mathbf{A}$ can have its singular value decomposition (SVD) $\mathbf{A} = \mathbf{U\Lambda V}$, a unitary transformation with $\mathbf{U}^H$ to (3.1) can be performed, yielding

$$\mathbf{r} = \mathbf{\Lambda V x} + \boldsymbol{\omega} \tag{3.16}$$

where $\mathbf{r} = \mathbf{U}^H\mathbf{y}$, $\boldsymbol{\omega} = \mathbf{U}^H\mathbf{w}$ is still a zero-mean Gaussian noise vector with the same covariance matrix $\beta^{-1}\mathbf{I}$ and $\mathbf{\Lambda}$ is an $M \times N$ rectangular diagonal matrix. Then the vector stepsize AMP can be applided to equation (3.16) where the system matrix becomes a special matrix $\mathbf{\Lambda V}$. Applying the vector step size AMP leads to the second version of

Algorithm 3 UAMP version 2

Unitary transform: $\mathbf{r} = \mathbf{U}^H\mathbf{y} = \mathbf{\Phi x} + \boldsymbol{\omega}$, where $\mathbf{\Phi} = \mathbf{U}^H\mathbf{A} = \mathbf{\Lambda V}$, and $\mathbf{U}$ is obtained from the SVD $\mathbf{A} = \mathbf{U\Lambda V}$.
Define vector $\boldsymbol{\lambda} = \mathbf{\Lambda\Lambda}^H\mathbf{1}$.
Initialize $\tau_x^{(0)} > 0$ and $\mathbf{x}^{(0)}$. Set $\mathbf{s}^{(-1)} = \mathbf{0}$ and $t = 0$.
Repeat

1: $\boldsymbol{\tau}_p = \tau_x^t\boldsymbol{\lambda}$
2: $\mathbf{p} = \mathbf{\Phi x}^t - \boldsymbol{\tau}_p \cdot \mathbf{s}^{t-1}$
3: $\boldsymbol{\tau}_s = \mathbf{1}./(\boldsymbol{\tau}_p + \beta^{-1}\mathbf{1})$
4: $\mathbf{s}^t = \boldsymbol{\tau}_s \cdot (\mathbf{r} - \mathbf{p})$
5: $1/\tau_q = (1/N)\boldsymbol{\lambda}^T\boldsymbol{\tau}_s$
6: $\mathbf{q} = \mathbf{x}^t + \tau_q\mathbf{\Phi}^H\mathbf{s}^t$
7: $\tau_x^{t+1} = (\tau_q/N)\mathbf{1}^H g_x'(\mathbf{q}, \tau_q)$
8: $\boldsymbol{x}^{t+1} = g_x(\mathbf{q}, \tau_q)$
9: $t = t + 1$

Until terminated

UAMP, which is given as **Algorithm** 3. Unless specifically stated otherwise, UAMP refers to the second version.

It is not hard to verify that

$$|\mathbf{C}|^2\mathbf{d} = (\mathbf{C}Diag(\mathbf{d})\mathbf{C}^H)_D\mathbf{1}. \tag{3.17}$$

Now suppose we have a variance vector $\boldsymbol{\tau}_x^t$. According to Line 1 in the vector stepsize AMP and using (3.17), we have

$$\boldsymbol{\tau}_p = (\mathbf{\Lambda V}Diag(\boldsymbol{\tau}_x^t)\mathbf{V}^H\mathbf{\Lambda}^H)_D\mathbf{1}. \tag{3.18}$$

We can find that if $Diag(\boldsymbol{\tau}_x^t)$ is a scaled identity matrix, the computation of (3.18) can be significantly simplified. This motivates the replacement of $\boldsymbol{\tau}_x^t$ with $\tau_x^t\mathbf{1}$ where τ_x^t is the average of the elements of $\boldsymbol{\tau}_x^t$. Hence (3.18) is reduced to

$$\boldsymbol{\tau}_p = \tau_x^t\mathbf{\Lambda\Lambda}^H\mathbf{1} \tag{3.19}$$

which is Line 1 of **Algorithm** 3. Lines 2, 3 and 4 of **Algorithm** 3 can be obtained according to Lines 2, 3 and 4 of the vector stepsize AMP by simply replacing $\mathbf{A}$ with $\mathbf{\Lambda V}$. According to (3.17) again, Line 5 of the vector stepsize AMP with matrix $\mathbf{\Lambda V}$ can

be represented as

$$\mathbf{1}./\boldsymbol{\tau}_q = (\mathbf{V}^H\boldsymbol{\Lambda}^H Diag(\boldsymbol{\tau}_s)\boldsymbol{\Lambda}\mathbf{V})_D\mathbf{1}. \tag{3.20}$$

We then replace the diagonal matrix $\boldsymbol{\Lambda}^H\text{Diag}(\boldsymbol{\tau}_s)\boldsymbol{\Lambda}$ with a scaled identity matrix $\beta\mathbf{I}$ where β is the average of the diagonal elements of $\boldsymbol{\Lambda}^H Diag(\boldsymbol{\tau}_s)\boldsymbol{\Lambda}$, i.e.,

$$\beta = (1/N)\mathbf{1}^H\boldsymbol{\Lambda}\boldsymbol{\Lambda}^H\boldsymbol{\tau}_s. \tag{3.21}$$

Hence (3.20) is reduced to Line 5 of the **Algorithm** 3 algorithm. Line 6 can be obtained from Line 6 of the vector stepsize AMP by replacing $\mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}$ with $\boldsymbol{\Lambda}\mathbf{V}$. Compared with Line 7 in the vector stepsize AMP, an additional average operation is performed in Line 7 in **Algorithm** 3 to meet the requirement of a scalar τ_x^t in Line 1. We note that the average operation is not necessarily in Line 7 as we can also put the additional average operation in Line 1. Line 8 in **Algorithm** 3 is the same as Line 8 of the vector stepsize AMP except that τ_q is a scalar.

Remarks: It is worth pointing out that UAMP is not equivalent to the vector step size AMP due to the approximations made in the derivation. Interestingly, it is these approximations that make UAMP much more robust than AMP.

As discussed in [25], applying an average operation to the two vectors $\boldsymbol{\tau}_x$ in Line 7 and $|\boldsymbol{\Phi}^H|^2\boldsymbol{\tau}_s$ in Line 5 in **Algorithm** 2 leads to the UAMP shown in **Algorithm** 3. Specifically, due to the average operation in Line 7 of **Algorithm** 2, $\boldsymbol{\tau}_x^t$ in Line 1 turns into a scaled all-one vector $\tau_x^t\mathbf{1}$. With $\boldsymbol{\Phi} = \boldsymbol{\Lambda}\mathbf{V}$ and noting that $\mathbf{V}$ is a unitary matrix, it is not hard to show that

$$\begin{aligned} \boldsymbol{\tau}_p &= |\boldsymbol{\Phi}|^2(\tau_x^t\mathbf{1}) \\ &= \tau_x^t\boldsymbol{\lambda}, \end{aligned} \tag{3.22}$$

which is Line 1 of **Algorithm** 3. Performing the average operation to vector $|\boldsymbol{\Phi}^H|^2\boldsymbol{\tau}_s$, i.e.,

$$< |\boldsymbol{\Phi}^H|^2\boldsymbol{\tau}_s >= \frac{1}{N}\boldsymbol{\lambda}^T\boldsymbol{\tau}_s \tag{3.23}$$

leads to Line 5 of **Algorithm** 3. It is worth highlighting that the two average operations

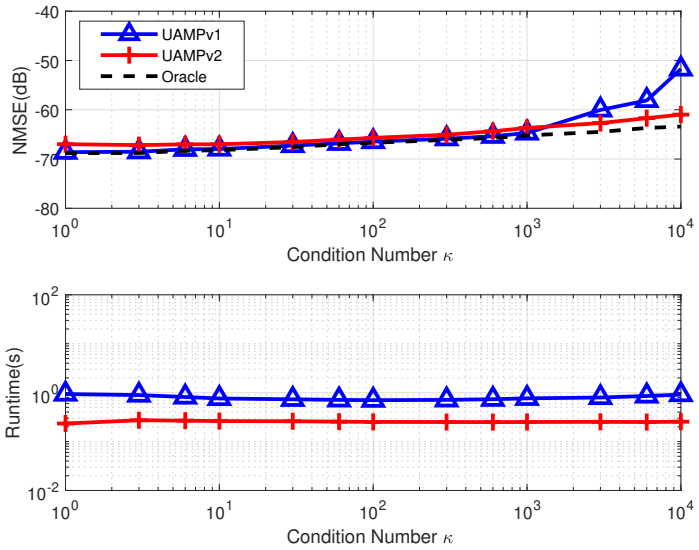

Figure 3.1: Performance and complexity comparisons of two versions of under ill-conditioned matrices in SNR = 60dB.

result in a significant reduction in computational complexity. Comparing **Algorithm** 2 with **Algorithm** 3, Line 1 and Line 5 of **Algorithm** 3 do not involve matrix-vector product operations, i.e., the number of matrix-vector products is reduced from 4 to 2 per iteration, which is a significant reduction as the complexity of AMP-like algorithms is dominated by matrix-vector products. Interestingly, the average operations also further enhance the stability of the algorithm from our finding. UAMP version 2 converges for any matrix $\mathbf{A}$ in the case of Gaussian priors [25]. In many cases, the noise precision β is unknown. The noise precision estimation can be incorporated into the UAMP algorithms as in [61].

The performance of the two versions of UAMP in SBL is compared. We compute the support-oracle bound on the achievable MSE based on the assumption that the support of $\mathbf{x}$ is known. The results are shown in Figure 3.1 in SNR = 60dB, where UAMP version 1 represents UAMP-SBL with UAMP version 1 and UAMP version 2 represents UAMP-SBL with UAMP version 2. It can be seen that when the condition number is small, UAMP version 1 performs slightly better than UAMP version 2, but UAMP version 2 performs better than UAMP version 1 when the conditional number is large (i.e., UAMP version 2 is more robust). Moreover, UAMP version 2 is also faster than UAMP version 1 due to the less number of matrix-vector products. UAMP-SBL with UAMP version 1 is also implemented, where no averaging operation is used, and UAMP-SBL with UAMP

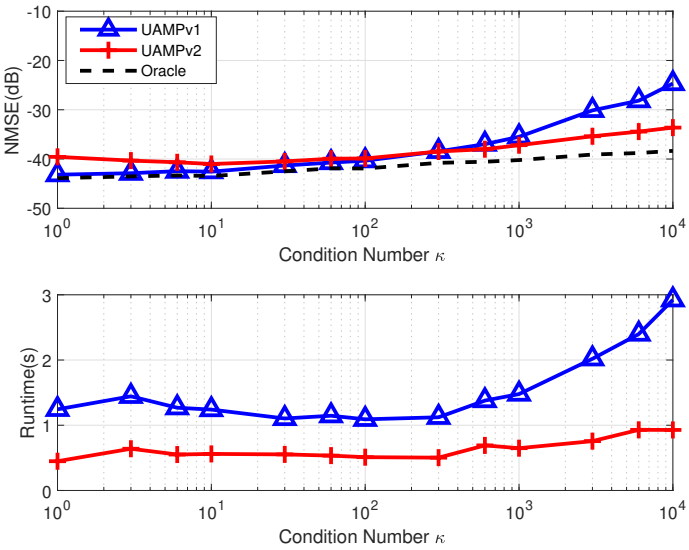

Figure 3.2: Performance and complexity comparisons of two versions under ill-conditioned matrices in SNR = 35dB.

version 2, where averaging operation is used. Their performance and the results shown in Figure 3.2 are also compared in SNR = 35dB. One is when the measurement matrix is not very tough, UAMP-SBL with UAMP version 1 performs better than UAMP-SBL with UAMP-version 2, which is due to the impact of averaging operation in UAMP version 2. The other is when the measurement matrices become tougher, UAMP-SBL with UAMP version 2 performs better than UAMP-SBL with UAMP version 1. This is because UAMP version 2 is more robust then UAMP version 1 based on our findings (It is shown in [25] that UAMP version 2 is guaranteed to converge for any matrix A in the case of Gaussian prior).

3.4.1 SE-Based Performance Prediction

As (U)AMP decouples the estimation of vector $\mathbf{x}$, in the tth iteration, the following pseudo observation model is

$$q_n^t = x_n + w_n^t, \tag{3.24}$$

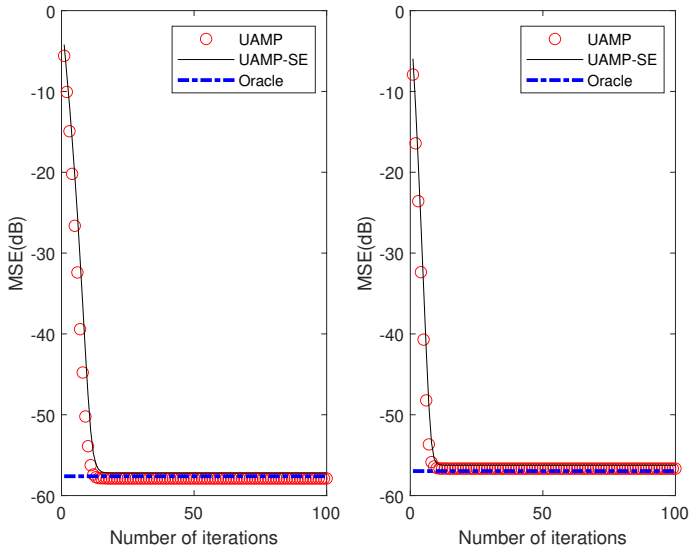

Figure 3.3: Performance of UAMP and its SE with a Bernoulli Gaussian prior for low-rank matrices (left) and non-zero mean matrices (right).

where q_n^t is the nth element of $\mathbf{q}$ in the tth iteration, and w_n^t denotes a Gaussian noise with mean 0. The variance of w_n^t denoted by τ^t is given by

$$\tau^t = \frac{N}{\mathbf{1}^H(\boldsymbol{\lambda}./(v_x^t\boldsymbol{\lambda} + \beta^{-1}\mathbf{1}))}, \tag{3.25}$$

which can be simply obtained based on Lines 1, 5 and 7 of UAMP version2. Here v_x^t is the average MSE of $\{x_n\}$ after denoising in the tth iteration. As it is difficult to obtain a closed form for the average MSE, the denoiser with the additive Gaussian noise model (3.24) by varying the variance of noise τ^t (or the SNR) is simulated, so that a "function" in terms of a table is obtained, with the variance of the noise as the input and the MSE as the output, i.e.,

$$v_x = \phi(\tau). \tag{3.26}$$

The performance of UAMP can be characterized by the following simple recursion

$$\tau^t = \frac{N}{\mathbf{1}^T(\boldsymbol{\lambda}./(v_x^t\boldsymbol{\lambda} + \beta^{-1}\mathbf{1}))} \tag{3.27}$$

$$v_x^{t+1} = \mathbb{E}\left[\left|g_x(x + \sqrt{\tau^t}z, \tau^t) - x\right|^2\right] \tag{3.28}$$

where β^{-1} is the noise variance, z is Gaussian with distribution $\mathcal{N}(z; 0, 1)$ and x has a prior

$p(x)$.

To demonstrate the SE of UAMP, the measurement matrix has a size of $M = 800$ and $N = 1000$ is assumed, the prior of the elements of $\mathbf{x}$ is Bernoulli Gaussian $p(x) = 0.9\delta(x) + 0.1\mathcal{N}(x; 0, 1)$, and the signal to noise ratio (SNR) is 50 dB. The non-zero mean matrices $\mathbf{A}$ with elements independently drawn from $\mathcal{N}(1, 1)$ is generated, and low rank matrices $\mathbf{A} = \mathbf{BC}$, where the size of $\mathbf{B}$ and $\mathbf{C}$ are 800×500 and 500×1000, respectively. Both $\mathbf{B}$ and $\mathbf{C}$ are i.i.d. Gaussian matrices with zero mean and unit variance. The mean squared error (MSE) of UAMP and its SE are shown in Figure 3.3. We compute the support-oracle bound on the achievable MSE based on the assumption that the support of $\mathbf{x}$ is known. It can be seen that in Figure 3.3 the SE matches well the simulation performance.

3.5 Conclusion

In this chapter, to facilitate comparisons with UAMP, a brief introduction of variational inference and AMP algorithm is given. The derivation of UAMP is provide which is more robust than AMP. Then, two versions of UAMP algorithm are described. Finally, the derivation and analysis of empirical state evolution used to predict the performance of UAMP are also detailed.

Chapter 4

UAMP for Sparse Bayesian Learning

4.1 Introduction

The problem of recovering a sparse signal $\mathbf{x}$ from noisy measurements $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{w}$, where $\mathbf{A}$ is a known measurement matrix [62] is considered. This problem finds numerous applications in various areas of signal processing, statistics and computer science [62], [63], [64], [65], [66], [67], [68]. One approach to recovering $\mathbf{x}$ is to use sparse Bayesian learning (SBL), where $\mathbf{x}$ is assumed to have a sparsity-promoting prior [6]. Conventional implementation of SBL involves matrix inversion in each iteration, resulting in prohibitive computational complexity for large scale problems.

The approximate message passing (AMP) algorithm [69], [30] has been proposed for low-complexity implementation of SBL [70, 71]. However, AMP does not work well for a generic matrix such as non-zero mean, rank-deficient, correlated, or ill-conditioned matrix $\mathbf{A}$ [20], resulting in divergence and poor performance. In [8], by incorporating damped Gaussian generalized AMP (GGAMP) to the EM-based SBL method, a GGAMP-SBL algorithm was proposed. Although the robustness of the approach is significantly improved, it comes at the cost of slowing the convergence. In addition, the algorithm still exhibits significant performance gap from the support-oracle bound when the measurement matrix has relatively high correlation, large condition number or non-zero mean.

To develop UAMP-SBL, we apply structured variational inference (SVI) [50], [52], [54]. In particular, the formulated problem is represented by a factor graph model, based

on which approximate inference is implemented in terms of structured variational message passing (SVMP) [52], [53], [54]. The use of SVMP allows the incorporation of UAMP to the message passing algorithm to handle the most computational intensive part of message computations with high robustness and low complexity. In UAMP-SBL, a Gamma distribution is used as the hyperprior for the precisions of the elements of $\mathbf{x}$. We propose to tune the shape parameter of the Gamma distribution automatically during iterations. We show by simulations that, in many cases with a generic measurement matrix, UAMP-SBL can still approach the support-oracle bound closely. In addition, the UAMP-SBL algorithm is extended from SMV problems to MMV [63], [72], [73]. Based on our preliminary results in [61], we present a new derivation of UAMP-SBL, extend it from SMV to MMV, and provide theoretical analyses and comprehensive comparisons. UAMP-SBL was applied to inverse synthetic aperture radar [74], where the measurement matrix can be highly correlated in order to achieve high Doppler resolution. Real data experiments in [74] demonstrate its superiority in terms of both recovery performance and speed.

4.1.1 Chapter's Organization

The organization of the chapter is as follows. In Section 4.2 the SBL algorithm is briefly reviewed. In Section 4.3, the UAMP algorithm is combined with the SBL algorithm to solve the SMV problem. the UAMP-SBL algorithm is introduce and the SE-based performance prediction for UAMP-SBL is investigate. In Section 4.4, the impact of the shape parameter is analyzed. In Section 4.5 UAMP-SBL is extended to the MMV setting.In Section 4.6 numerical results are presented to compare the performance and complexity of the proposed algorithms with the original SBL and with other AMP algorithms for the SMV case, and for the MMV case.

4.2 Sparse Bayesian Learning

Consider recovering a length-N sparse vector $\mathbf{x}$ from measurements

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{w}, \tag{4.1}$$

where $\mathbf{y}$ is a measurement vector of length M, the measurement matrix $\mathbf{A}$ has size $M \times N$, $\mathbf{w}$ denotes a Gaussian noise vector with mean zero and covariance matrix $\beta^{-1}\mathbf{I}$, and β is the precision of the noise. It is assumed that the elements in $\mathbf{x}$ are independent and the following two-layer sparsity-promoting prior is used

$$p(\mathbf{x}|\boldsymbol{\gamma}) = \prod_n p(x_n|\gamma_n) = \prod_n \mathcal{N}(x_n|0, \gamma_n^{-1}), \tag{4.2}$$

$$p(\boldsymbol{\gamma}) = \prod_n p(\gamma_n) = \prod_n \mathrm{Ga}(\gamma_n|\epsilon, \eta), \tag{4.3}$$

i.e., the prior of x_n is a Gaussian mixture

$$p(x_n) = \int \mathcal{N}(x_n|0, \gamma_n^{-1}) p(\gamma_n) d\gamma_n, \tag{4.4}$$

where the precision vector $\gamma = [\gamma_1, \gamma_2, ..., \gamma_N]^H$.

In the conventional SBL algorithm by Tipping [6], the precision vector $\boldsymbol{\gamma}$ is learned by maximizing the a posteriori probability

$$p(\boldsymbol{\gamma}|\mathbf{y}) \propto p(\mathbf{y}|\boldsymbol{\gamma}) p(\boldsymbol{\gamma}), \tag{4.5}$$

where the marginal likelihood function

$$p(\mathbf{y}|\boldsymbol{\gamma}) = \int p(\mathbf{y}|\mathbf{x}) p(\mathbf{x}|\boldsymbol{\gamma}) d\mathbf{x}. \tag{4.6}$$

It can be shown that [6]

$$\log p(\mathbf{y}|\boldsymbol{\gamma}) = -\frac{1}{2}\big(\log|\boldsymbol{\Sigma}| + \log|Diag(\boldsymbol{\gamma})| - \boldsymbol{\zeta}^H Diag(\boldsymbol{\gamma})\boldsymbol{\zeta}\big) + const, \tag{4.7}$$

where *const* represents terms independent of $\boldsymbol{\gamma}$, and

$$\boldsymbol{\Sigma} = \left(\beta \mathbf{A}^H \mathbf{A} + Diag(\boldsymbol{\gamma})\right)^{-1}, \tag{4.8}$$

$$\boldsymbol{\zeta} = \beta \boldsymbol{\Sigma} \mathbf{A}^H \mathbf{y}. \tag{4.9}$$

The posterior probability of $\mathbf{x}$

$$p(\mathbf{x}|\mathbf{y}, \boldsymbol{\gamma}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\zeta}, \boldsymbol{\Sigma}). \tag{4.10}$$

By taking the logarithm of $p(\boldsymbol{\gamma}|\mathbf{y})$ and ignoring the terms independent of $\boldsymbol{\gamma}$, the learning of $\boldsymbol{\gamma}$ is to maximize the following objective function [6]

$$\mathcal{L}(\boldsymbol{\gamma}) = \log p(\mathbf{y}|\boldsymbol{\gamma}) + \sum_{n=1}^{N} (\epsilon \log \gamma_n - \eta \gamma_n). \tag{4.11}$$

As the value of $\boldsymbol{\gamma}$ that maximizes $\mathcal{L}(\boldsymbol{\gamma})$ cannot be obtained in a closed form, iterative re-estimation is employed by taking advantage of (4.7), i.e., with a learned $\boldsymbol{\gamma}$ in the last iteration, compute $\boldsymbol{\Sigma}$ and $\boldsymbol{\zeta}$ with (4.8) and (4.9), then update $\boldsymbol{\gamma}$ by maximizing $\mathcal{L}(\boldsymbol{\gamma})$ with (4.7) used, which leads to a closed form to update γ_n

$$\gamma_n = (2\epsilon + 1)/(2\eta + |\zeta_n|^2 + \Sigma_{n,n}), n = 1, ..., N. \tag{4.12}$$

In summary, Tipping's SBL algorithm (which is called SBL hereafter) executes the following iteration [6]:

Repeat

$$\mathbf{Z} = \left(\beta \mathbf{A}^H \mathbf{A} + Diag(\hat{\boldsymbol{\gamma}})\right)^{-1} \tag{4.13}$$

$$\hat{\mathbf{x}} = \beta \mathbf{Z} \mathbf{A}^H \mathbf{y} \tag{4.14}$$

$$\hat{\gamma}_n = (2\epsilon + 1)/(2\eta + |\hat{x}_n|^2 + Z_{n,n}), n = 1, ..., N. \tag{4.15}$$

Until terminated

If the noise precision β is unknown, its estimation can be incorporated as well. The SBL algorithm can also be derived based on the EM algorithm [6], [8]. The SBL algorithm requires a matrix inverse in (4.13) in each iteration, resulting in cubic complexity per iteration.

4.3 Sparse Bayesian Learning Using UAMP

4.3.1 Problem Formulation and Approximate Inference

To enable the use of UAMP, the unitary transformed model $\mathbf{r} = \mathbf{\Phi x} + \boldsymbol{\omega}$ in (3.16) is employed. As in many applications the noise precision β is unknown, its estimation is also considered. The joint conditional distribution of $\mathbf{x}$, $\boldsymbol{\gamma}$ and β can be expressed as

$$p(\mathbf{x}, \boldsymbol{\gamma}, \beta|\mathbf{r}) \propto p(\mathbf{r}|\mathbf{x}, \beta)p(\mathbf{x}|\boldsymbol{\gamma})p(\boldsymbol{\gamma})p(\beta), \tag{4.16}$$

where $p(\mathbf{x}|\boldsymbol{\gamma})$ and $p(\boldsymbol{\gamma})$ are given by (4.2) and (4.3), respectively. It is assumed that an improper prior $p(\beta) \propto 1/\beta$ for the noise precision [6]. According to the transformed model (3.16), $p(\mathbf{r}|\mathbf{x}, \beta) = \mathcal{N}(\mathbf{r}|\mathbf{\Phi x}, \beta^{-1}\mathbf{I})$. Our aim is to find the marginal distribution $p(\mathbf{x}|\mathbf{r})$. The a posteriori mean is then used as an estimate of $\mathbf{x}$ in the sense of minimum mean squared error (MSE). However, exact inference is intractable due to the high dimensional integration involved, so approximate inference techniques is applied to.

Variational inference is a machine learning method for approximate inference [50], [52], [54]. In variational inference, a tractable trial distribution function is chosen and optimized by minimizing the Kullback-Leibler (KL) divergence between it and the true posterior function. Instead of using fully factorized trial functions where all variables are assumed to be independent (thereby likely resulting in poor approximations), more structured factorizations can be used, leading to SVI algorithms. With graphical models, SVI can be formulated as message-passing [52], [54], [53], which is termed SVMP. In this work, SVMP is adopted because it facilitates the incorporation of UAMP into SVMP. it will be shown how UAMP can be used to handle the most computational intensive part of message computations, how to achieve low complexity and high robustness. With

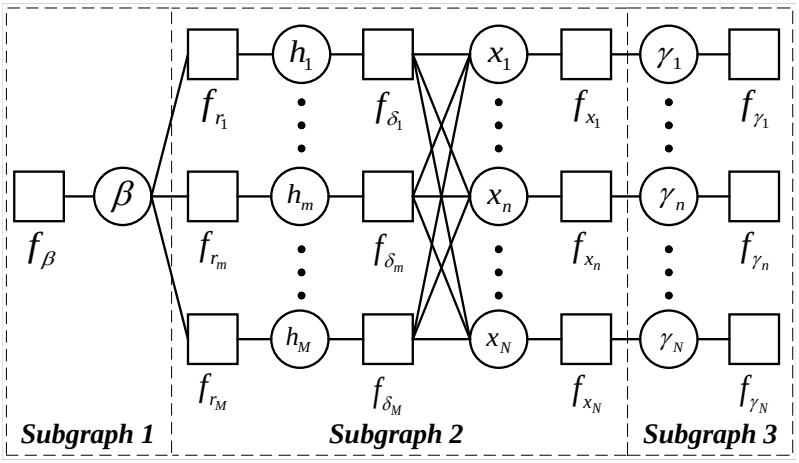

Figure 4.1: Factor graph of (4.17) for deriving UAMP-SBL.

SVMP, it also can be seen that an approximation to the marginal distribution $p(\mathbf{x}|\mathbf{r})$, where an approximation to $p(\boldsymbol{\gamma}|\mathbf{r})$ is also involved (the approximate inference for $\mathbf{x}$ and $\boldsymbol{\gamma}$ is performed alternately).

An auxiliary variable $\mathbf{h} = \boldsymbol{\Phi}\mathbf{x}$ to facilitate the incorporation of UAMP is introduced, which is crucial to an efficient realization of SBL. Then the conditional joint distribution is

$$\begin{aligned} & p(\mathbf{x}, \mathbf{h}, \boldsymbol{\gamma}, \beta|\mathbf{r}) \\ & \quad \propto p(\mathbf{r}|\mathbf{h}, \beta) p(\mathbf{h}|\mathbf{x}) p(\mathbf{x}|\boldsymbol{\gamma}) p(\boldsymbol{\gamma}|\epsilon) p(\beta) \\ & \quad = \prod_{m=1}^{M} \mathcal{N}(r_m|h_m, \beta^{-1}) \prod_{m=1}^{M} \delta(h_m - [\boldsymbol{\Phi}]_m \mathbf{x}) \\ & \qquad \prod_{n=1}^{N} \mathcal{N}(x_n|0, \gamma_n^{-1}) \prod_{n=1}^{N} \mathrm{Ga}(\gamma_n|\epsilon, \eta) p(\beta). \end{aligned} \tag{4.17}$$

To facilitate the derivation of the message passing algorithm, a factor graph representation of the factorization in (4.17) is shown in Figure4.1, where the local functions $f_\beta(\beta) \propto 1/\beta$, $f_{r_m}(r_m, h_m, \beta) = \mathcal{N}(r_m|h_m, \beta^{-1})$, $f_{\delta_m}(h_m, \mathbf{x}) = \delta(h_m - [\boldsymbol{\Phi}]_m \mathbf{x})$, $f_{x_n}(x_n, \gamma_n) = \mathcal{N}(x_n|0, \gamma_n^{-1})$, $f_{\gamma_n}(\gamma_n) = \mathrm{Ga}(\gamma_n|\epsilon, \eta)$ and $[\boldsymbol{\Phi}]_m$ is the mth row of matrix $\boldsymbol{\Phi}$.

Following SVI, the following structured trial function is defined by

$$\tilde{q}(\mathbf{x}, \mathbf{h}, \boldsymbol{\gamma}, \beta) = \tilde{q}(\beta)\tilde{q}(\mathbf{x}, \mathbf{h})\tilde{q}(\boldsymbol{\gamma}). \tag{4.18}$$

In terms of SVMP, the use of the above trial function corresponds to a partition of the

Algorithm 4 UAMP-SBL

Unitary transform: $\mathbf{r} = \mathbf{U}^H\mathbf{y} = \mathbf{\Phi}\mathbf{x} + \omega$, where $\mathbf{\Phi} = \mathbf{U}^H\mathbf{A} = \mathbf{\Lambda}\mathbf{V}$, and $\mathbf{A}$ has SVD $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}$.
Define vector $\boldsymbol{\lambda} = \mathbf{\Lambda}\mathbf{\Lambda}^H\mathbf{1}$.
Initialization: $\tau_x^{(0)} = 1$, $\hat{\mathbf{x}}^{(0)} = \mathbf{0}$, $\epsilon = 0.001$, $\hat{\boldsymbol{\gamma}} = \mathbf{1}$, $\hat{\beta} = 1$, $\mathbf{s} = \mathbf{0}$, and $t = 0$.
Do

1: $\boldsymbol{\tau}_p = \tau_x^t\boldsymbol{\lambda}$
2: $\mathbf{p} = \mathbf{\Phi}\hat{\boldsymbol{x}}^t - \boldsymbol{\tau}_p \cdot \mathbf{s}$
3: $\mathbf{v}_h = \boldsymbol{\tau}_p./(\mathbf{1} + \hat{\beta}\boldsymbol{\tau}_p)$
4: $\hat{\mathbf{h}} = (\hat{\beta}\boldsymbol{\tau}_p \cdot \mathbf{r} + \mathbf{p})./(\mathbf{1} + \hat{\beta}\boldsymbol{\tau}_p)$
5: $\hat{\beta} = M/(\|\mathbf{r} - \hat{\mathbf{h}}\|^2 + \mathbf{1}^H\mathbf{v}_h)$
6: $\boldsymbol{\tau}_s = \mathbf{1}./(\boldsymbol{\tau}_p + \hat{\beta}^{-1}\mathbf{1})$
7: $\mathbf{s} = \boldsymbol{\tau}_s \cdot (\mathbf{r} - \mathbf{p})$
8: $1/\tau_q = (1/N)\boldsymbol{\lambda}^H\boldsymbol{\tau}_s$
9: $\mathbf{q} = \hat{\mathbf{x}}^t + \tau_q\mathbf{\Phi}^H\mathbf{s}$
10: $\tau_x^{t+1} = (\tau_q/N)\mathbf{1}^H(\mathbf{1}./(\mathbf{1} + \tau_q\hat{\boldsymbol{\gamma}}))$
11: $\hat{\mathbf{x}}^{t+1} = \mathbf{q}./(\mathbf{1} + \tau_q\hat{\boldsymbol{\gamma}})$
12: $\hat{\gamma}_n = (2\epsilon + 1)/(|\hat{x}_n^{t+1}|^2 + \tau_x^{t+1}), n = 1, ..., N.$
13: $\epsilon = \frac{1}{2}\sqrt{\log(\frac{1}{N}\sum_n \hat{\gamma}_n) - \frac{1}{N}\sum_n \log\hat{\gamma}_n}$
14: $t = t + 1$

while $(\|\hat{\mathbf{x}}^{t+1} - \hat{\mathbf{x}}^t\|^2/\|\hat{\mathbf{x}}^{t+1}\|^2 > \delta_x$ and $t < t_{max})$

factor graph shown by the dotted boxes in Figure 4.1, where $\tilde{q}(\beta)$, $\tilde{q}(\mathbf{x}, \mathbf{h})$ and $\tilde{q}(\boldsymbol{\gamma})$ are associated with Subgraphs 1, 2 and 3, respectively.

As the KL divergence

$$\mathcal{KL}(\tilde{q}(\beta)\tilde{q}(\mathbf{x}, \mathbf{h})\tilde{q}(\boldsymbol{\gamma})\|p(\mathbf{x}, \mathbf{h}, \boldsymbol{\gamma}, \beta|\mathbf{r})), \tag{4.19}$$

is minimized, it is expected that

$$\tilde{q}(\mathbf{x}, \mathbf{h}) \approx p(\mathbf{x}, \mathbf{h}|\mathbf{r}), \tag{4.20}$$

$$\tilde{q}(\boldsymbol{\gamma}) \approx p(\boldsymbol{\gamma}|\mathbf{r}), \tag{4.21}$$

$$\tilde{q}(\beta) \approx p(\beta|\mathbf{r}). \tag{4.22}$$

Integrating out $\mathbf{h}$ in (4.20), which corresponds to running BP in Subgraph 2 (except the factor nodes connecting external variable nodes), $\tilde{q}(\mathbf{x}) \approx p(\mathbf{x}|\mathbf{r})$ is obtained. Running BP in Subgraph 2 involves the most intensive computations; fortunately it can be handled efficiently and with high robustness using UAMP. The derivation of UAMP-SBL is shown

in the following, and the algorithm is summarized in **Algorithm** 4.

Regarding the UAMP-SBL in **Algorithm** 4, the following remarks is given:

1. UAMPv2 which is shown in chapter 3 is employed in **Algorithm** 2. Similarly, UAMPv1 which is also shown in chapter 3 of **Algorithm** 3 can also be used. By comparing UAMPv1 and UAMPv2, the differences lie in Lines 1, 8, 9 and 10 as vectors $\boldsymbol{\tau}_x^t$ and $\boldsymbol{\tau}_q$ need to be used. The UAMP-SBL algorithms with two version of UAMP deliver comparable performance, but UAMP-SBL with UAMPv2 has lower complexity.

2. In SBL with Gamma hyperprior, the shape parameter ϵ and the rate parameter η are normally chosen to be very small values [6], and sometimes the value of the shape parameter ϵ is chosen empirically, e.g., $\epsilon = 1$ in [75]. In UAMP-SBL, to tune the shape parameter automatically (as shown in Line 13) with the following empirical rule is proposed by

$$\epsilon = \frac{1}{2}\sqrt{\log(\frac{1}{N}\sum_n \hat{\gamma}_n) - \frac{1}{N}\sum_n \log\hat{\gamma}_n}, \tag{4.23}$$

i.e., ϵ is learned iteratively with the iteration, starting from a small positive initial value. It is noted that, as the log function is concave, the parameter ϵ in (4.23) is guaranteed to be non-negative. In Section 4.4, it will be shown that the shape parameter ϵ in the SBL algorithms functions as a selective amplifier for $\{\gamma_n\}$, and a proper ϵ plays a significant role in promoting sparsity, leading to considerable performance improvement.

4.3.2 Derivation of UAMP-SBL with SVMP

We detail the forward and backward message passing in each subgraph of the factor graph in Figure 4.1 according to the principle of SVMP [50], [52], [53]. The notation $\mathcal{M}_{n_a\to n_b}(x)$ is used to denote a message passed from node n_a to node n_b, which is a function of x. Note that, if a forward message computation requires backward messages, we use the messages in previous iteration by default.

4.3.2.1 Message Computations in Subgraph 1

In this subgraph, we only need to compute the outgoing (forward) messages $\{\mathcal{M}_{\beta\to f_{r_m}}(\beta)\}$, which are input to Subgraph 2. The derivation of the message update rule is delayed in the message computations in Subgraph 2, and is given in (4.34).

4.3.2.2 Message Computations in Subgraph 2

According to SVMP, we need to run BP in this subgraph except at the factor nodes $\{f_{r_m}\}$ as they connect external variable nodes. Due to the involvement of $\mathbf{\Phi}$, this is the most computational intensive part, and we propose to use UAMP to handle it by integrating it to the message passing process.

According to the derivation of (U)AMP using loopy BP, UAMP provides the message from variable node h_m to function node f_{r_m}. Due to the Gaussian approximation in the the derivation of (U)AMP, the message is Gaussian, i.e.,

$$\mathcal{M}_{h_m\to f_{r_m}}(h_m) = \mathcal{M}_{f_{\delta_m}\to h_m}(h_m) = \mathcal{N}(h_m|p_m, \tau_{p_m}), \tag{4.24}$$

where the mean p_m and the variance τ_{p_m} are respectively the mth elements of $\boldsymbol{p}$ and $\boldsymbol{\tau}_p$ given in Line 2 and Line 1 of the UAMP algorithm (**Algorithm** 2), which are also Line 2 and Line 1 of the UAMP-SBL algorithm (**Algorithm** 4).

Following SVMP [53], the message $\mathcal{M}_{f_{r_m}\to\beta}(\beta)$ from factor node f_{r_m} to variable node β can be expressed as

$$\mathcal{M}_{f_{r_m}\to\beta}(\beta) \propto \exp\left\{\left\langle \log f_{r_m}(r_m|h_m, \beta^{-1})\right\rangle_{b(h_m)}\right\}, \tag{4.25}$$

where the belief of h_m is given as

$$b(h_m) \propto \mathcal{M}_{h_m\to f_{r_m}}(h_m)\mathcal{M}_{f_{r_m}\to h_m}(h_m). \tag{4.26}$$

Later we will see that $\mathcal{M}_{f_{r_m}\to h_m}(h_m) \propto \mathcal{N}(h_m|r_m, \hat{\beta}^{-1})$ where $\hat{\beta}^{-1}$ is an estimate of β^{-1} (in the last iteration), and its computation is delayed to (4.36). Hence $b(h_m)$ is Gaussian according to the property of the product of Gaussian functions, i.e., $b(h_m) = \mathcal{N}(h_m|\hat{h}_m, v_{h_m})$

with

$$v_{h_m} = (1/\tau_{p_m} + \hat{\beta})^{-1} \tag{4.27}$$

$$\hat{h}_m = v_{h_m}(\hat{\beta} r_m + p_m/\tau_{p_m}). \tag{4.28}$$

They can be rewritten in vector form as

$$\mathbf{v}_h = \boldsymbol{\tau}_p./(\mathbf{1} + \hat{\beta}\boldsymbol{\tau}_p) \tag{4.29}$$

$$\hat{\mathbf{h}} = (\hat{\beta}\boldsymbol{\tau}_p \cdot \boldsymbol{r} + \boldsymbol{p})./(\mathbf{1} + \hat{\beta}\boldsymbol{\tau}_p), \tag{4.30}$$

to avoid numerical problems as $\boldsymbol{\tau}_p$ may contain zero elements, which are Lines 3 and 4 of the UAMP-SBL algorithm. Then, from (4.25) and the Gaussianity of $b(h_m)$, the message $\mathcal{M}_{f_{r_m}\to\beta}(\beta)$ is

$$\mathcal{M}_{f_{r_m}\to\beta}(\beta) \propto \sqrt{\beta}\exp\left\{-\frac{\beta}{2}(|r_m - \hat{h}_m|^2 + v_{h_m})\right\}. \tag{4.31}$$

According to SVMP, the message from function node f_{r_m} to variable node h_m is

$$\begin{aligned}\mathcal{M}_{f_{r_m}\to h_m}(h_m) &\propto \exp\left\{\left\langle \log f_{r_m}(r_m|h_m,\beta^{-1})\right\rangle_{b(\beta)}\right\} \\ &\propto \mathcal{N}(h_m|r_m,\hat{\beta}^{-1}),\end{aligned} \tag{4.32}$$

where $\hat{\beta} = \langle\beta\rangle_{b(\beta)}$ with

$$\begin{aligned}b(\beta) &= \mathcal{M}_{\beta\to f_{r_m}}(\beta)\mathcal{M}_{f_{r_m}\to\beta}(\beta) \\ &= f_\beta(\beta)\prod_m \mathcal{M}_{f_{r_m}\to\beta}(\beta) \\ &\propto \beta^{\frac{M}{2}-1}\exp\left\{-\frac{\beta}{2}\sum_m\left(|r_m - \hat{h}_m|^2 + v_{h_m}\right)\right\},\end{aligned} \tag{4.33}$$

and

$$\mathcal{M}_{\beta\to f_{r_m}}(\beta) = f_\beta(\beta)\prod_{m'\neq m}\mathcal{M}_{f_{r_{m'}}\to\beta}(\beta). \tag{4.34}$$

It is noted that $b(\beta)$ is a Gamma distribution with the rate parameter

$$\frac{1}{2}\sum_m \left(|r_m - \hat{h}_m|^2 + v_{h_m}\right) \tag{4.35}$$

and the shape parameter $M/2$, so $\hat{\beta} = \langle\beta\rangle_{b(\beta)}$ can be computed as

$$\hat{\beta} = M/\sum_m \left(|r_m - \hat{h}_m|^2 + v_{h_m}\right), \tag{4.36}$$

which can be rewritten in vector form shown in Line 5 of the UAMP-SBL algorithm.

From (4.32), the Gaussian form of the message $\mathcal{M}_{f_{r_m}\to h_m}(h_m)$ suggests the following model

$$r_m = h_m + w_m, m = 1, ..., M, \tag{4.37}$$

where w_m is a Gaussian noise with mean 0 and variance $\hat{\beta}^{-1}$. This fits into the forward recursion of the UAMP algorithm as if the noise variance is known. Therefore, Lines 3 - 6 of the UAMP algorithm (**Algorithm** 2) can be executed, which are Lines 6 - 9 of the UAMP-SBL algorithm. According to the derivation of (U)AMP, UAMP produces the message $\mathcal{M}_{x_n\to f_{x_n}}(x_n) \propto \mathcal{N}(x_n|q_n, \tau_q)$ with mean q_n and variance τ_q, which are given in Lines 5 and 6 of the UAMP algorithm or Line 8 and Line 9 of the UAMP-SBL algorithm. We can see that the UAMP algorithm is integrated.

The function nodes $\{f_{x_n}\}$ connect the external variable node γ_n. According to SVMP, the outgoing message of Subgraph 2 $\mathcal{M}_{f_{x_n}\to\gamma_n}(\gamma_n)$ can be expressed as

$$\mathcal{M}_{f_{x_n}\to\gamma_n}(\gamma_n) \propto \exp\left\{\left\langle \log f_{x_n}(x_n|0, \gamma_n^{-1})\right\rangle_{b(x_n)}\right\}, \tag{4.38}$$

where the belief $b(x_n) \propto \mathcal{M}_{x_n\to f_{x_n}}(x_n)\mathcal{M}_{f_{x_n}\to x_n}(x_n)$.

The message $\mathcal{M}_{f_{x_n}\to x_n}(x_n) \propto \mathcal{N}(x_n|0, \hat{\gamma}_n^{-1})$ will be computed in (4.45), where $\hat{\gamma}_n = \langle\gamma_n\rangle_{b(\gamma_n)}$. Then $b(x_n)$ turns out to be Gaussian, i.e., $b(x_n) = \mathcal{N}(x_n|\hat{x}_n, \tau_{x_n})$ with

$$\tau_{x_n} = \left(1/\tau_q + \hat{\gamma}_n\right)^{-1} \tag{4.39}$$

$$\hat{x}_n = q_n/(1 + \tau_q\hat{\gamma}_n). \tag{4.40}$$

Performing the average operations to $\{\tau_{x_n}\}$ in (4.39) and arranging (4.40) in a vector form lead to Lines 10 and 11 of the UAMP-SBL algorithm. According to the above,

$$\mathcal{M}_{f_{x_n}\to\gamma_n}(\gamma_n) \propto \sqrt{\gamma_n}\exp\left\{-\frac{\gamma_n}{2}(|\hat{x}_n|^2+\tau_x)\right\}, \tag{4.41}$$

which is passed to Subgraph 3. This is the end of the message update in Subgraph 2.

4.3.2.3 Message Computations in Subgraph 3

The message $\mathcal{M}_{f_{\gamma_n}\to\gamma_n}(\gamma_n)$ from the factor node f_{γ_n} to the variable node γ_n is a predefined Gamma distribution with shape parameter ϵ and rate parameter η, i.e.,

$$\mathcal{M}_{f_{\gamma_n}\to\gamma_n}(\gamma_n) \propto {\gamma_n}^{\epsilon-1}\exp\{-\eta\gamma_n\}. \tag{4.42}$$

According to SVMP, the message

$$\mathcal{M}_{f_{x_n}\to x_n}(x_n) \propto \exp\left\{\left\langle \log f_x(x_n|0,\gamma_n^{-1})\right\rangle_{b(\gamma_n)}\right\}, \tag{4.43}$$

where the belief of γ_n

$$\begin{aligned} b(\gamma_n) &\propto \mathcal{M}_{f_{\gamma_n}\to\gamma_n}(\gamma_n)\mathcal{M}_{f_{x_n}\to\gamma_n}(\gamma_n) \\ &\propto {\gamma_n}^{\epsilon-\frac{1}{2}}\exp\left\{-\frac{\gamma_n}{2}(|\hat{x}_n|^2+\tau_x+2\eta)\right\}. \end{aligned} \tag{4.44}$$

Hence, the message

$$\mathcal{M}_{f_{x_n}\to x_n}(x_n) \propto \mathcal{N}(x_n|0,\hat{\gamma}_n^{-1}), \tag{4.45}$$

where

$$\hat{\gamma}_n = \langle\gamma_n\rangle_{b(\gamma_n)} = \frac{2\epsilon+1}{2\eta+|\hat{x}_n|^2+\tau_x}. \tag{4.46}$$

Here we set $\eta = 0$, and $\hat{\gamma}_n$ is reduced to

$$\frac{(2\epsilon+1)}{|\hat{x}_n|^2+\tau_x}, \tag{4.47}$$

which leads to Line 12 of the UAMP-SBL algorithm.

4.3.3 Computational Complexity

UAMP-SBL works well with a simple single loop iteration, which is in contrast to the double loop iterative algorithm GGAMP-SBL [8]. The complexity of UAMP-SBL (with UAMPv2) is dominated by two matrix-vector product operations in Line 2 and Line 9, i.e., $O(MN)$ per iteration. The algorithm typically converges fast and delivers outstanding performance as shown in Section 4.6. UAMP-SBL involves an SVD , but it only needs to be computed once and may be carried out off-line. The complexity of economic SVD is $O(\min\{M^2N, MN^2\})$. Note that for the runtime comparison in Section 4.6, off-line SVD computation is not been assumed, and the time consumed by SVD is counted for UAMP-SBL.

4.4 Impact of the Shape Parameter ϵ in SBL

In this section, the impact of the hyperparameter ϵ on the convergence of SBL is analyzed. the case of an identity matrix $\mathbf{A}$ is focused on. The same results for a general $\mathbf{A}$ are demonstrated numerically.

It is considerd that the conventional SBL algorithm (η is set to be zero) [6]. In the case of identity matrix $\mathbf{A}$, it reduces to

$$
\begin{aligned}
&\textit{Repeat}\\
&\quad Z_{n,n} = (\beta + \gamma_n^t)^{-1}\\
&\quad \hat{x}_n = \beta Z_{n,n} y_n \qquad (4.48)\\
&\quad \gamma_n^{t+1} = (2\epsilon + 1)/(|\hat{x}_n|^2 + Z_{n,n})\\
&\textit{Until terminated}
\end{aligned}
$$

Here note that in the above iteration $\gamma_n^{(0)} > 0$ is initialized. The iteration in terms of γ_n has

a closed form

$$\begin{aligned}\gamma_n^{t+1} &= \frac{2\epsilon+1}{(\beta(\beta+\gamma_n^t)^{-1}y_n)^2+(\beta+\gamma_n^t)^{-1}} \\ &= (2\epsilon+1)\frac{(\beta+\gamma_n^t)^2}{(\beta y_n)^2+\beta+\gamma_n^t} \\ &\triangleq g_\epsilon(\gamma_n^t).\end{aligned} \tag{4.49}$$

Next,the impact of ϵ on the convergence behavior and fixed point of the iteration (4.49) are investigated when $\epsilon = 0$ or ϵ takes a positive value.

For the iteration (4.49) with a small positive initial value $\gamma_n^{(0)}$, the following proposition and theorem is given.

Proposition 1: When $\epsilon = 0$, if $\beta y_n^2 > 1$, γ_n^t converges to a stable fixed point

$$\gamma_n' = \frac{\beta}{\beta y_n^2 - 1}; \tag{4.50}$$

if $\beta y_n^2 \leq 1$, γ_n^t goes to $+\infty$.

Proof. When $\epsilon = 0$, the iteration in terms of γ_n has a simplified closed form, i.e.,

$$\gamma_n^{t+1} = g_{\epsilon_0}(\gamma_n^t) = \frac{(\beta+\gamma_n^t)^2}{(\beta y_n)^2+\beta+\gamma_n^t}. \tag{4.51}$$

In order to find the fixed point, we need to solve the following equation

$$f(\gamma_n) = g_{\epsilon_0}(\gamma_n) - \gamma_n = 0, \tag{4.52}$$

which leads to the unique root

$$\gamma_n' = \frac{\beta}{\beta y_n^2 - 1}. \tag{4.53}$$

If $\beta y_n^2 > 1$, the root $\gamma_n' = \frac{\beta}{\beta y_n^2 - 1} > 0$.

Taking the derivative of $g_{\epsilon_0}(\gamma_n)$ in (4.51), we have

$$\frac{d}{d\gamma_n} g_{\epsilon_0}(\gamma_n) = 1 - \left(\frac{\beta^2 y_n^2}{\beta^2 y_n^2 + \beta + \gamma_n}\right)^2. \tag{4.54}$$

It is easy to verify that, when $\gamma_n > 0$,

$$0 < \frac{d}{d\gamma_n} g_{\epsilon_0}(\gamma_n) < 1. \tag{4.55}$$

Thus, the unique root

$$\gamma_n' = \frac{\beta}{\beta y_n^2 - 1} \tag{4.56}$$

is a stable fixed point of the iteration. when $\gamma_n > 0$,

As $0 < \frac{d}{d\gamma_n} g_{\epsilon_0}(\gamma_n) < 1$ with an initial value $\gamma_n^{(0)} > 0$, γ_n^t will converge to the stable fixed point γ_n' [76].

If $\beta y_n^2 \leq 1$, the root

$$\gamma_n' = \frac{\beta}{\beta y_n^2 - 1} < 0 \tag{4.57}$$

or $\gamma_n' = +\infty$, i.e., there is no cross-point between $y = g_{\epsilon_0}(\gamma_n)$ and $y = \gamma_n$ when $\gamma^n > 0$. As $g_{\epsilon_0}(0) = \frac{\beta^2}{(\beta y_n)^2 + \beta} > 0$, $y = g_{\epsilon_0}(\gamma_n)$ is above $y = \gamma_n$ for $\gamma_n > 0$. In addition, $y = g_{\epsilon_0}(\gamma_n)$ is an increasing function for $\gamma_n > 0$. Hence γ_n^t goes to $+\infty$ with the iteration.

□

Theorem 1: When $\epsilon > 0$, if $\beta y_n^2 > 1 + 4\epsilon + 4\sqrt{\epsilon^2 + \epsilon/2}$, γ_n^t converges to a stable fixed point

$$\gamma_{n(a)} = \frac{2\beta(1 + 2\epsilon)}{\beta y_n^2 - 4\epsilon - 1 + \sqrt{\beta^2 y_n^4 - 8\epsilon\beta y_n^2 - 2\beta y_n^2 + 1}}; \tag{4.58}$$

if $\beta y_n^2 < 1 + 4\epsilon + 4\sqrt{\epsilon^2 + \epsilon/2}$, γ_n^t goes to $+\infty$.

Proof. With $\epsilon > 0$, the derivative of $g_\epsilon(\gamma_n)$ is given as

$$\frac{dg_\epsilon(\gamma_n)}{d\gamma_n} = (2\epsilon + 1)\left(1 - \left(\frac{\beta u_n}{\beta u_n + \beta + \gamma_n}\right)^2\right), \tag{4.59}$$

where $u_n = \beta y_n^2$.

To find the fixed points of the iteration, we let

$$f(\gamma_n) = g_\epsilon(\gamma_n) - \gamma_n = 0, \tag{4.60}$$

leading to

$$2\epsilon\gamma_n^2 - \gamma_n\beta(\beta y_n^2 - 4\epsilon - 1) + \beta^2(1 + 2\epsilon) = 0. \tag{4.61}$$

The two roots of (4.61) are given by an alternative form for the quadratic formula is used, which can be deduced from the standard quadratic formula by Vieta's formulas.

$$\gamma_{n(a)} = \frac{2\beta(1 + 2\epsilon)}{u_n - 4\epsilon - 1 + \sqrt{u_n^2 - 8\epsilon u_n - 2u_n + 1}}, \tag{4.62}$$

and

$$\gamma_{n(b)} = \frac{2\beta(1 + 2\epsilon)}{u_n - 4\epsilon - 1 - \sqrt{u_n^2 - 8\epsilon u_n - 2u_n + 1}}. \tag{4.63}$$

If

$$u_n > 1 + 4\epsilon + 4\sqrt{\epsilon^2 + \epsilon/2}, \tag{4.64}$$

it is not hard to verify that

$$u_n - 4\epsilon - 1 - \sqrt{u_n^2 - 8\epsilon u_n - 2u_n + 1} > 0, \tag{4.65}$$

so both roots are positive. Hence they are two fixed points of the iteration. Next, we show that $\gamma_{n(a)}$ is a stable fixed point while $\gamma_{n(b)}$ is an unstable one.

Plugging the root $\gamma_{n(a)}$ into (4.59), we have

$$\frac{d}{d\gamma_n} g_\epsilon(\gamma_n)\bigg|_{\gamma_n = \gamma_{n(a)}} = (2\epsilon + 1)\left(1 - \left(\frac{\beta u_n}{\beta u_n + \beta + \gamma_{n(a)}}\right)^2\right). \tag{4.66}$$

It is clear that the derivative is larger than 0. Verifying that

$$\frac{d}{d\gamma_n} g_\epsilon(\gamma_n)|_{\gamma_n = \gamma_{n(a)}} < 1 \tag{4.67}$$

is equivalent to showing that

$$l(u_n) = (2\epsilon + 1)(\beta u_n)^2 - 2\epsilon(\beta u_n + \beta + \gamma_{n(a)})^2 \tag{4.68}$$

is larger than 0. Inserting (4.62) into (4.68),

$$\frac{4\epsilon l(u_n)}{\beta^2} = l_1(u_n) + ((4\epsilon + 1)u_n - 1)\sqrt{-l_1(u_n)}, \tag{4.69}$$

where

$$l_1(u_n) = -(u_n^2 - 8\epsilon u_n - 2u_n + 1) < 0. \tag{4.70}$$

Then

$$\frac{4\epsilon l(u_n)}{\beta^2} = \sqrt{-l_1(u_n)}(-\sqrt{-l_1(u_n)} + (4\epsilon u_n + u_n - 1)). \tag{4.71}$$

Because

$$(4\epsilon u_n + u_n - 1)^2 - (-l_1(u_n)) = 16\epsilon^2 u_n^2 + 8\epsilon u_n > 0, \tag{4.72}$$

the term in (4.71)

$$-\sqrt{-l_1(u_n)} + (4\epsilon u_n + u_n - 1) > 0, \tag{4.73}$$

and we have $l(u_n) > 0$. Therefore,

$$\frac{d}{d\gamma_n} g_\epsilon(\gamma_n)|_{\gamma_n = \gamma_{n(a)}} < 1, \tag{4.74}$$

i.e., $\gamma_{n(a)}$ is a stable fixed point. Similarly, it is not hard to show that $l(u_n) < 0$ (i.e., $\frac{d}{d\gamma_n} g_\epsilon(\gamma_n) > 1$) for $\gamma_n = \gamma_{n(b)}$, i.e., $\gamma_{n(b)}$ is an unstable fixed point.

Then we analyze the convergence behavior. As $\gamma_n > 0$, the derivative (4.59) is an increasing function and it is positive. In the above, it is already shown that

$$\frac{d}{d\gamma_n} g_\epsilon(\gamma_n)|_{\gamma_n = \gamma_{n(a)}} < 1. \tag{4.75}$$

Therefore, for $\gamma_n \in [0, \gamma_{n(a)}]$,

$$0 < \frac{d}{d\gamma_n} g_\epsilon(\gamma_n) < 1. \tag{4.76}$$

Thus, with an initial $\gamma_n^{(0)}$ with the range, γ_n^t converges to the stable fixed point $\gamma_{n(a)}$ [76].

Next we consider

$$u_n < 1 + 4\epsilon + 4\sqrt{\epsilon^2 + \epsilon/2}. \tag{4.77}$$

For $u_n \in (1 + 4\epsilon - 4\sqrt{\epsilon^2 + \epsilon/2}, 1 + 4\epsilon + 4\sqrt{\epsilon^2 + \epsilon/2})$, it can be verified that

$$u_n^2 - 8\epsilon u_n - 2u_n + 1 < 0, \tag{4.78}$$

leading to two complex roots $\gamma_{n(a)}$ and $\gamma_{n(b)}$. If

$$u_n \leq 1 + 4\epsilon - 4\sqrt{\epsilon^2 + \epsilon/2}, \tag{4.79}$$

it can be shown that

$$u_n^2 - 8\epsilon u_n - 2u_n + 1 \geq 0, \tag{4.80}$$

and

$$u_n^2 - 8\epsilon u_n - 2u_n + 1 < (u_n - 4\epsilon - 1)^2. \tag{4.81}$$

Thus

$$u_n - 4\epsilon - 1 < -4\sqrt{\epsilon^2 + \epsilon/2} < 0 \tag{4.82}$$

and

$$u_n - 4\epsilon - 1 \pm \sqrt{u_n^2 - 8\epsilon u_n - 2u_n + 1} < 0, \tag{4.83}$$

leading to negative $\gamma_{n(a)}$ and $\gamma_{n(b)}$. In summary, if

$$u_n < 1 + 4\epsilon + 4\sqrt{\epsilon^2 + \epsilon/2}, \tag{4.84}$$

the two roots are either complex or negative.

Hence, there is no cross-point between $y = g_\epsilon(\gamma_n)$ and $y = \gamma_n$ for $\gamma_n > 0$. As

$$g_\epsilon(0) = (2\epsilon + 1)\frac{\beta^2}{(\beta y_n)^2 + \beta} > 0, \tag{4.85}$$

$y = g_\epsilon(\gamma_n)$ is above $y = \gamma_n$. Meanwhile $g_\epsilon(\gamma_n^t)$ is an increasing function. Hence, γ_n^t goes to $+\infty$ with the iteration.

When

$$u_n = 1 + 4\epsilon + 4\sqrt{\epsilon^2 + \epsilon/2}, \tag{4.86}$$

there is single root

$$\gamma_n^* = \frac{2\beta(1+2\epsilon)}{u_n - 1 - 4\epsilon}. \tag{4.87}$$

Plugging γ_n^* into (4.59), we have

$$\frac{d}{d\gamma_n} g_\epsilon(\gamma_n)|_{\gamma_n=\gamma_n^*} = 1. \tag{4.88}$$

Thus γ_n^* is neutral fixed point [76]. Depending on the initial value $\gamma_n^{(0)}$, γ_n^t may converge to the fixed point γ_n^* or diverge.

□

Based on Proposition 1 and Theorem 1, the following remarks are made by:

1. If $\beta y_n^2 \leq 1$, for both $\epsilon = 0$ and $\epsilon > 0$, γ_n^t goes to $+\infty$. However, a positive ϵ accelerates the move of γ_n^t towards $+\infty$. This can be shown as follows. As $\beta > 0$ and $\beta y_n^2 \leq 1$, $(\beta y_n)^2 \leq \beta$ is obtained. Hence, from (4.49)

$$\begin{aligned} \gamma_n^{t+1} = g_\epsilon(\gamma_n^t) &\geq (2\epsilon+1)\frac{(\beta+\gamma_n^t)^2}{2\beta+\gamma_n^t} \\ &= (2\epsilon+1)\left(\gamma_n^t + \frac{\beta^2}{2\beta+\gamma_n^t}\right) \\ &> (2\epsilon+1)\gamma_n^t. \end{aligned} \tag{4.89}$$

   From (4.89), compared to $\epsilon = 0$, a positive value of ϵ moves γ_n^t towards infinity more quickly. Considering a fixed number of iterations, a positive value of ϵ can be significant because the precision can reach a large value much faster.

2. When $\epsilon = 0$, γ_n^t converges to a finite fixed point if $\beta y_n^2 > 1$. In contrast, when $\epsilon > 0$, γ_n^t goes to $+\infty$ if $\beta y_n^2 \in (1, 1 + 4\epsilon + 4\sqrt{\epsilon^2 + \epsilon/2})$. This is an additional range for γ_n^t to go to infinity. Hence, a positive ϵ is stronger in terms of promoting sparsity, compared to $\epsilon = 0$.

3. When $\epsilon > 0$, if $\beta y_n^2 = 1 + 4\epsilon + 4\sqrt{\epsilon^2 + \epsilon/2}$, γ_n^t may converge or diverge because the iteration has a unique neutral fixed point as shown in Theorem 1.

4. When $\beta y_n^2 > 1 + 4\epsilon + 4\sqrt{\epsilon^2 + \epsilon/2}$, γ_n^t always converges to a fixed point. Based on

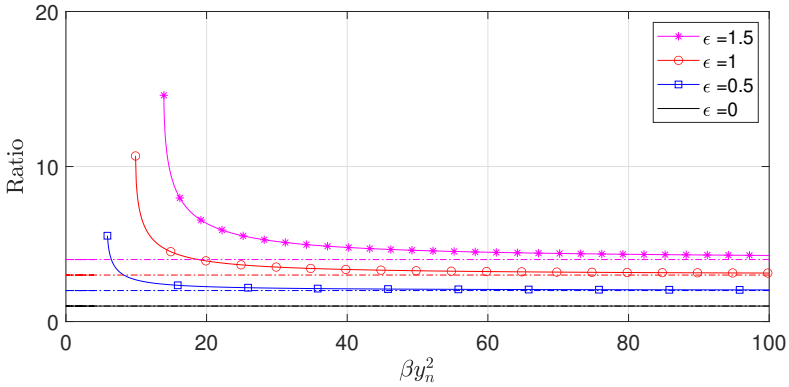

Figure 4.2: Ratio of precisions with different ϵ.

(4.50) and (4.58), the ratio of the precisions obtained with $\epsilon > 0$ and $\epsilon = 0$ is given by

$$\frac{\gamma_{n(a)}}{\gamma'_n} = \frac{2(1+2\epsilon)}{1 - \frac{4\epsilon}{\beta y_n^2 - 1} + \sqrt{\left(1 - \frac{4\epsilon}{\beta y_n^2 - 1}\right)^2 - \frac{8\epsilon(1+2\epsilon)}{(\beta y_n^2 - 1)^2}}}. \tag{4.90}$$

The ratio is a function of βy_n^2, and

$$\gamma_{n(a)}/\gamma'_n \approx 1 + 2\epsilon, \tag{4.91}$$

if βy_n^2 is relatively large.

The ratios of the precisions versus βy_n^2 are shown in Figure 4.2, where they are not shown for $\beta y_n^2 < 1 + 4\epsilon + 4\sqrt{\epsilon^2 + \epsilon/2}$ as they are infinity when $1 < \beta y_n^2 < 1 + 4\epsilon + 4\sqrt{\epsilon^2 + \epsilon/2}$, and undefined when $\beta y_n^2 \leq 1$ (see the above remarks). It can be seen that the precision obtained with $\epsilon = 0$ is amplified depending on the value of βy_n^2. The smaller the value of βy_n^2, the larger the amplification for the corresponding precision (in the case of $\beta y_n^2 \leq 1$, the ratios are undefined. However, considering a fixed number of iterations, the ratios can be large as γ_n^t with a positive ϵ goes to infinity much quicker). Note that $y_n = x_n + w_n$ and β is the noise precision. Hence, if βy_n^2 is a small value, it is highly likely that the corresponding x_n is zero, hence the precision γ_n should go to infinity. If βy_n^2 is a large value, it is highly likely that the corresponding x_n is non-zero, hence γ_n should be a finite value. It can be seen that a positive ϵ tends to a sparser solution, and a proper value of ϵ leads to much better recovery performance, compared to $\epsilon = 0$.

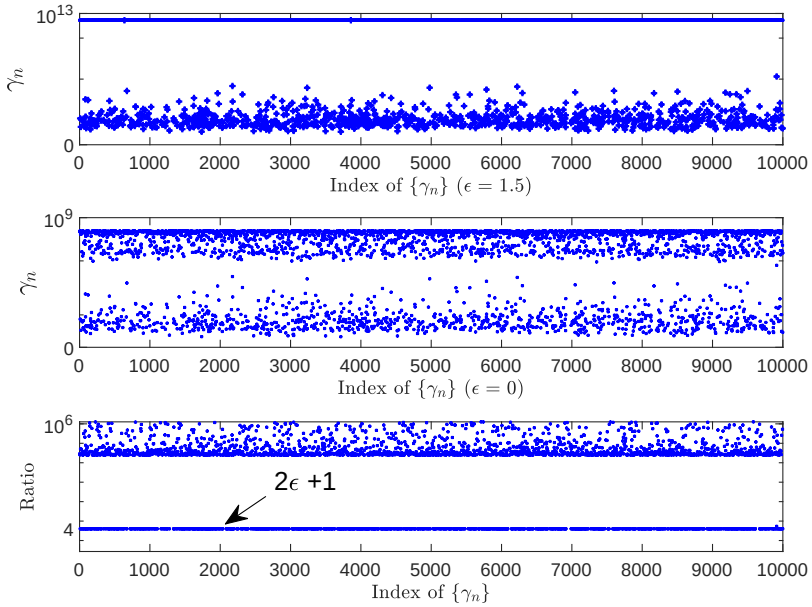

Figure 4.3: Precisions and their ratios (**A** is an identity matrix).

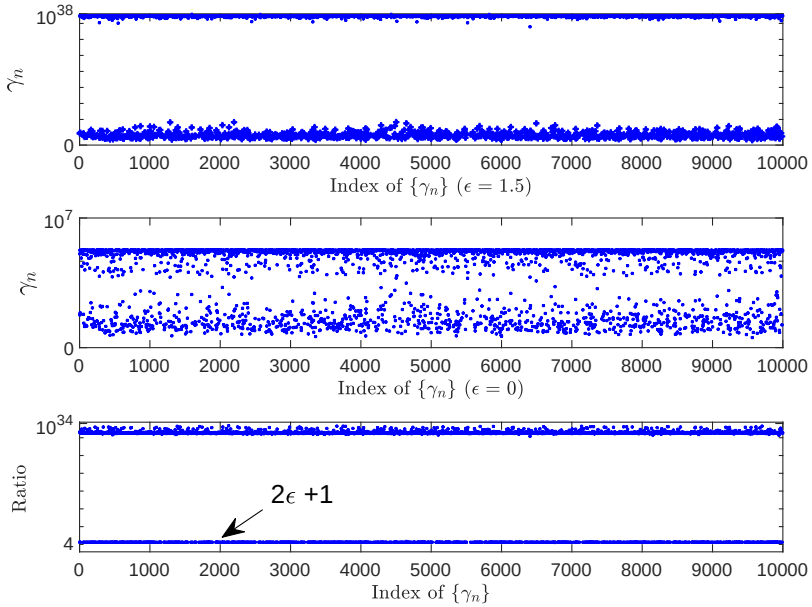

Figure 4.4: Precisions and their ratios (**A** is i.i.d Gaussian).

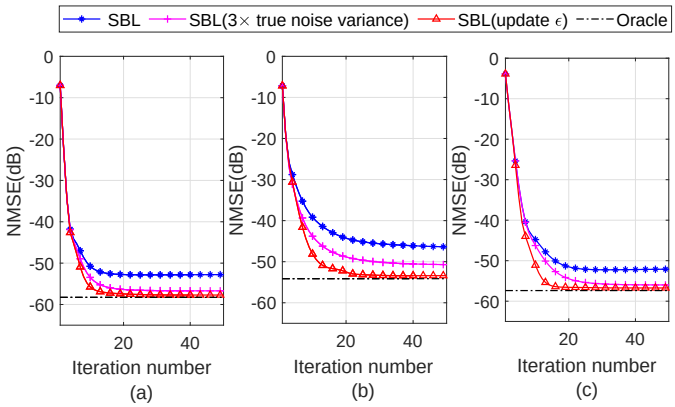

Figure 4.5: Performance of the conventional SBL. (a) Gaussian matrix; (b) correlated matrix with $c = 0.3$; (c) low-rank matrix with $R/N = 0.6$.

The precisions of the elements of the sparse vector obtained by the SBL algorithm with $\epsilon = 1.5$ and $\epsilon = 0$ are shown in Figure 4.3, where $\mathbf{A}$ is an identity matrix with size 10000×10000, the sparsity rate of the signal is 0.1, and SNR = 50dB. It can be seen that the precisions with $\epsilon = 1.5$ are separated into two groups more clearly, and the ratios for the small precisions are roughly 4 (i.e., $1 + 2\epsilon$), while other precisions are amplified significantly. Although the above analysis is for an identity matrix $\mathbf{A}$, it is interesting that the same results are observed for a general matrix $\mathbf{A}$ as demonstrated numerically in Figure 4.4, where $\mathbf{A}$ is an i.i.d Gaussian matrix with size 5000×10000, the non-zero shape parameter $\epsilon = 1.5$, and the sparsity rate and the SNR are the same as the case of identity matrix. (Similar observations are observed for other matrices). It can be seen that the small precisions are also roughly amplified by 4 times while others are amplified significantly, leading to two well-separated groups.

It is noted that the value of ϵ should be determined properly. If the matrix $\mathbf{A}$ and the sparsity rate of $\mathbf{x}$ are given, a proper value for ϵ through trial and error is can be found. However, this is inconvenient, and the sparsity rate of the signal may not be available. The empirical equation (4.23) to determine the value of ϵ is found. Next, its effectiveness with the SBL algorithm is examined.

Plugging the shape parameter update rule (4.23) to the conventional SBL algorithm

leads to the following iterative algorithm (assuming the noise precision β is known):

Repeat

$$\mathbf{Z} = \left(\beta \mathbf{A}^H \mathbf{A} + Diag(\hat{\boldsymbol{\gamma}})\right)^{-1}$$

$$\hat{\mathbf{x}} = \beta \mathbf{Z} \mathbf{A}^H \mathbf{y}$$

$$\hat{\gamma}_n = (2\epsilon + 1)/(|\hat{x}_n|^2 + Z_{n,n}), n = 1, ..., N$$

$$\epsilon = \frac{1}{2} \sqrt{\log(\frac{1}{N} \sum_n \hat{\gamma}_n) - \frac{1}{N} \sum_n \log \hat{\gamma}_n}$$

Until terminated

To demonstrate the effectiveness of the shape parameter update rule (4.23), the performance of the conventional SBL algorithm with and without shape parameter update is compared. The results are shown in Figure 4.5, where the SNR is 50dB, the size of the measurement matrix is 800×1000, and the sparsity rate $\rho = 0.1$. The performances of SBL at lower SNR are provided in Section 4.6.

In this figure, the support-oracle bound is also shown for reference. The matrices in (a), (b), and (c) are respectively i.i.d. Gaussian, correlated and low-rank matrices (refer to Section 4.6 for their generations). It can be seen that there is a clear gap between the performance of the conventional SBL and the bounds, and with shape parameter updated with our rule, the SBL algorithm attains the bound. It is worth mentioning the empirical finding in [8], i.e., replacing the noise variance β^{-1} with $3\beta^{-1}$ can lead to better performance of GGAMP-SBL [8]. It is used for the conventional SBL algorithm and the performance is also included in Figure 4.5. It can be seen that it also leads to substantial performance improvement, but its performance is inferior to that of SBL with updated ϵ using (4.23). Moreover, in many cases, the noise variance is unknown, and it may be hard to determine its value accurately. In contrast, our empirical update of ϵ does not require any additional information.

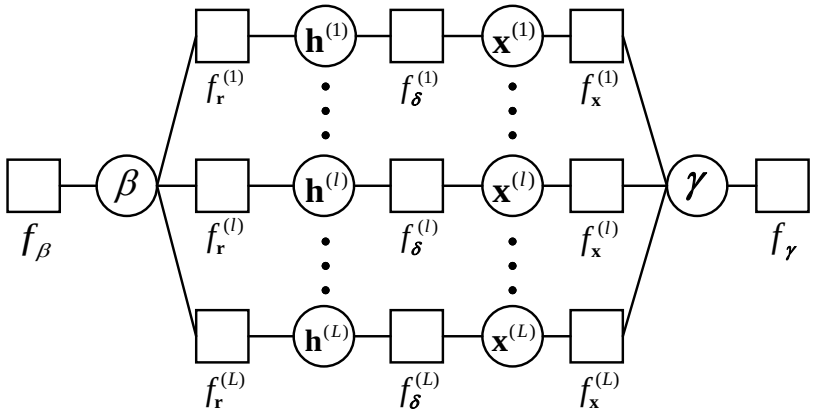

Figure 4.6: Factor graph representation of (4.94).

4.5 Extension to MMV

In this section, the MMV setting is extended by UAMP-SBL, where the relation among the sparse vectors is exploited, e.g., common support and temporal correlation.

4.5.1 UAMP-SBL for MMV

The objective on an MMV problem is to recover a collection of length-N sparse vectors $\mathbf{X} = \left[\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(L)}\right]$ from L noisy length-M measurement vectors $\mathbf{Y} = \left[\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, ..., \mathbf{y}^{(L)}\right]$ with the following model

$$\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{W}, \tag{4.92}$$

where it is assumed that the L vectors $\{\mathbf{x}^{(l)}\}$ share a common support (i.e., joint sparsity), $\mathbf{A}$ is a known measurement matrix with size $M \times N$, and $\mathbf{W}$ denotes an i.i.d. Gaussian noise matrix with the elements having mean zero and precision β.

With the SVD $\mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}$, a unitary transformation with $\mathbf{U}^H$ to (4.92) can be performed, i.e.,

$$\mathbf{R} = \boldsymbol{\Phi}\mathbf{X} + \boldsymbol{\Omega}, \tag{4.93}$$

where $\mathbf{R} = \mathbf{U}^H\mathbf{Y} = \left[\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, ..., \mathbf{r}^{(L)}\right]$, $\boldsymbol{\Phi} = \boldsymbol{U}^H\mathbf{A} = \boldsymbol{\Lambda}\mathbf{V}$ and $\boldsymbol{\Omega} = \mathbf{U}^H\mathbf{W}$ is still white and Gaussian with mean zero and precision β. Define $\mathbf{h}^{(l)} = \boldsymbol{\Phi}\boldsymbol{x}^{(l)}$ and $\mathbf{H} = [\mathbf{h}^{(1)}, ..., \mathbf{h}^{(L)}]$.

Algorithm 5 UAMP-SBL for MMV

Unitary transform: $\mathbf{R} = \mathbf{U}^H\mathbf{Y} = \mathbf{\Phi X} + \mathbf{W}$, where $\mathbf{\Phi} = \boldsymbol{U}^H\mathbf{A} = \mathbf{\Lambda}\boldsymbol{V}$, and $\mathbf{A}$ has SVD $\boldsymbol{A} = \boldsymbol{U}\mathbf{\Lambda}\boldsymbol{V}$.
Define vector $\boldsymbol{\lambda} = \mathbf{\Lambda}\mathbf{\Lambda}^H\mathbf{1}$.
Initialization: $\forall l$; $\tau_x^{l(0)} = 1$, $\hat{\mathbf{x}}^{l(0)} = \mathbf{0}$, $\epsilon' = 0.001$, $\hat{\boldsymbol{\gamma}} = \mathbf{1}$, $\hat{\beta} = 1$, $\mathbf{s}^l = \mathbf{0}$, and $t = 0$.
Do

1: $\forall l$; ${\boldsymbol{\tau}_p}^l = \tau_x^{l(t)}\boldsymbol{\lambda}$
2: $\forall l$; $\mathbf{p}^l = \mathbf{\Phi}\hat{\mathbf{x}}^{l(t)} - \boldsymbol{\tau}_p^l \cdot \mathbf{s}^l$
3: $\forall l$; $\mathbf{v}_h^l = \boldsymbol{\tau}_p^l./(\mathbf{1} + \hat{\beta}\boldsymbol{\tau}_p^l)$
4: $\forall l$; $\hat{\mathbf{h}}^l = (\hat{\beta}\boldsymbol{\tau}_p^l \cdot \mathbf{r}^l + \mathbf{p}^l)./(\mathbf{1} + \hat{\beta}\boldsymbol{\tau}_p^l)$
5: $\hat{\beta} = LM/(\sum_l(||\mathbf{r}^l - \hat{\mathbf{h}}^l||^2 + \mathbf{1}^H\mathbf{v}_h^l))$;
6: $\forall l$; $\boldsymbol{\tau}_s^l = \mathbf{1}./(\boldsymbol{\tau}_p^l + \hat{\beta}^{-1}\mathbf{1})$
7: $\forall l$; $\mathbf{s}^l = \boldsymbol{\tau}_s^l \cdot (\mathbf{r}^l - \mathbf{p}^l)$
8: $\forall l$; $1/\tau_q^l = (1/N)\boldsymbol{\lambda}^H\boldsymbol{\tau}_s^l$
9: $\forall l$; $\mathbf{q}^l = \hat{\mathbf{x}}^{l(t)} + \tau_q^l(\mathbf{\Phi}^H\mathbf{s}^l)$
10: $\forall l$; $\tau_x^{l(t+1)} = (\tau_q^l/N)\mathbf{1}^H(\mathbf{1}./(\mathbf{1} + \tau_q^l\hat{\boldsymbol{\gamma}}))$
11: $\forall l$; $\hat{\mathbf{x}}^{l(t+1)} = \mathbf{q}^l./(\mathbf{1} + \tau_q^l\hat{\boldsymbol{\gamma}})$
12: $\hat{\gamma}_n = \frac{2\epsilon'+1}{(1/L)\sum_{l=1}^L(|\hat{x}_n^{l(t+1)}|^2 + \tau_x^{l(t+1)})}, n = 1, ..., N.$
13: $\epsilon' = \frac{1}{2}\sqrt{\log(\frac{1}{N}\sum_n \hat{\gamma}_n) - \frac{1}{N}\sum_n \log\hat{\gamma}_n}$
14: $t = t + 1$
while $\frac{1}{L}\sum_{l=1}^L(||\hat{\mathbf{x}}^{l(t+1)} - \hat{\mathbf{x}}^{l(t)}||^2/||\hat{\mathbf{x}}^{l(t+1)}||^2) > \delta_x$ and $t < t_{max}$)

Then the following joint distribution is given by

$$
\begin{aligned}
&p(\mathbf{X}, \mathbf{H}, \boldsymbol{\gamma}, \beta|\mathbf{R}) \\
&\propto \prod_{l=1}^{L} p(\mathbf{r}^{(l)}|\mathbf{h}^{(l)}, \beta)p(\mathbf{h}^{(l)}|\mathbf{x}^{(l)})p(\mathbf{x}^{(l)}|\boldsymbol{\gamma})p(\boldsymbol{\gamma})p(\beta) \\
&= \prod_{l=1}^{L}\prod_{m=1}^{M} \mathcal{N}(r_m^{(l)}|h_m^{(l)}, \beta^{-1})\delta(h_m^{(l)} - [\mathbf{\Phi}]_m\mathbf{x}^{(l)}) \\
&\times \prod_{l=1}^{L}\prod_{n=1}^{N} \mathcal{N}(x_n^{(l)}|0, \gamma_n^{-1}) \prod_{n=1}^{N} \mathrm{Ga}(\gamma_n|\epsilon, \eta)p(\beta).
\end{aligned} \tag{4.94}
$$

Define factors $f_{\mathbf{r}}^{(l)}(\mathbf{r}^{(l)}, \mathbf{h}^{(l)}, \beta) = \prod_m \mathcal{N}(r_m^{(l)}|h_m^{(l)}, \beta)$, $f_{\boldsymbol{\delta}}^{(l)}(\mathbf{h}^{(l)}, \mathbf{x}^{(l)}) = \prod_m \delta(h_m^{(l)} - [\mathbf{\Phi}]_m\mathbf{x}^{(l)})$, $f_\beta(\beta) \propto 1/\beta$, $f_{\mathbf{x}}^{(l)}(\mathbf{x}^{(l)}, \boldsymbol{\gamma}) = \prod_n \mathcal{N}(x_n^{(l)}|0, \gamma_n^{-1})$, and $f_{\boldsymbol{\gamma}}(\boldsymbol{\gamma}, \epsilon) = \prod_n \mathrm{Ga}(\gamma_n|\epsilon, \eta)$ denotes the hyperprior of the hyperparameters $\{\gamma_n\}$. The factor graph representation of (4.94) is shown in Figure 4.6. The vector variable node $\boldsymbol{\gamma}$ is used in the factor graph to make it neat. It is noted that each entry $x_n^{(l)}$ of $\boldsymbol{x}^{(l)}$ is connected to γ_n through the function node between them., based on which the message passing algorithm can be derived. The message updates related to $\mathbf{x}^{(l)}$ and $\mathbf{h}^{(l)}$ are the same as those for the SMV case and can be computed in parallel. The difference lies in the computations of $\hat{\beta}$ and $\hat{\boldsymbol{\gamma}}$, and the relevant derivations

are shown in the following. The UAMP-SBL for MMV is summarized in **Algorithm** 5, where UAMPv2 is employed. The complexity of the algorithm is $O(MNL)$ per iteration.

The belief $b(\beta)$ can be represented as

$$\begin{aligned} b(\beta) &\propto f_\beta(\beta)\prod_{l,m}\mathcal{M}_{f_{r_m}^{(l)}\to\beta}(\beta) \\ &\propto 1/\beta\prod_{l,m}\mathcal{N}(h_m^{(l)}|r_m^{(l)},\hat{\beta}^{-1}). \end{aligned} \tag{4.95}$$

Then according to the equation

$$\hat{\beta} = \langle\beta\rangle_{b(\beta)}, \tag{4.96}$$

we have

$$\hat{\beta} = ML/\sum_{m,l}\left(|r_m^{(l)} - \hat{h}_m^{(l)}|^2 + v_{h_m}^{(l)}\right). \tag{4.97}$$

According to the factor graph in Figure 4.6, the belief $b(\gamma_n)$ can be updated as

$$\begin{aligned} b(\gamma_n^{(l)}) &\propto \mathcal{M}_{f^{(l)}_{\gamma_n\to\gamma_n^{(l)}}}(\gamma_n^{(l)})\mathcal{M}_{f^{(l)}_{x_n\to\gamma_n^{(l)}}}(\gamma_n^{(l)}) \\ &= (\gamma_n^{(l)})^{\epsilon-1+\frac{1}{2}}\exp\left\{-\frac{\gamma_n^{(l)}}{2}(2\eta + (|\hat{x}_n^{(l)}|^2 + \tau_x^{(l)}))\right\}. \end{aligned} \tag{4.98}$$

Here, we still set $\eta = 0$ and the expectation of γ_n leads to

$$\hat{\gamma}_n = \frac{2\epsilon' + 1}{(1/L)\sum_{l=1}^{L}(|\hat{x}_n^{(l)}|^2 + \tau_x^{(l)})}, \tag{4.99}$$

where $\epsilon' = \epsilon/L$. By comparing (4.99) with (4.46), the update of ϵ' can be expressed as

$$\epsilon' = \frac{1}{2}\sqrt{\log(\frac{1}{N}\sum_n\hat{\gamma}_n) - \frac{1}{N}\sum_n\log\hat{\gamma}_n}. \tag{4.100}$$

4.5.2 UAMP-TSBL

With the assumption of a common sparsity profile shared by all sparse vectors, it is further considered exploiting the temporal correlation that exists between the non-zero elements. The messages update related to $\mathbf{h}^{(l)}$, ϵ and β are the same as those for the

Algorithm 6 UAMP-TSBL

Unitary transform: $\mathbf{R} = \mathbf{U}^H\mathbf{Y} = \boldsymbol{\Phi}\mathbf{X} + \mathbf{W}$, where $\boldsymbol{\Phi} = \boldsymbol{U}^H\mathbf{A} = \boldsymbol{\Lambda}\boldsymbol{V}$, and $\mathbf{A}$ has SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}$.

Define vector $\boldsymbol{\lambda} = \boldsymbol{\Lambda}\boldsymbol{\Lambda}^H\mathbf{1}$.

Initialization: $\forall l$: $\tau_x^{l(0)} = 1$, $\hat{\mathbf{x}}^{l(0)} = \mathbf{0}$, $\mathbf{q}^l = \mathbf{0}$,$\boldsymbol{\tau}_q^{l(0)} = \mathbf{1}$, $\boldsymbol{\xi}^{l(0)} = \mathbf{0}$, $\boldsymbol{\psi}^{l(0)} = \mathbf{1}$, $\boldsymbol{\theta}^{l(0)} = \mathbf{0}$, $\boldsymbol{\phi}^{l(0)} = \mathbf{1}$, $\mathbf{s}^{l(-1)} = \mathbf{0}$, $\epsilon' = 0.001$, $\hat{\boldsymbol{\gamma}}^{(0)} = \mathbf{1}$, $\hat{\beta} = 1$, and $t = 0$.

Do

$\boldsymbol{\xi}^1 = \mathbf{0}$

$\boldsymbol{\psi}^1 = \mathbf{1}./\hat{\boldsymbol{\gamma}}^{(t)}$

for $l = 2, ..., L$

  $\boldsymbol{\xi}^l = \alpha\left(\frac{\mathbf{q}^{l-1}}{\boldsymbol{\tau}_q^{l-1}} + \frac{\boldsymbol{\xi}^{l-1}}{\boldsymbol{\psi}^{l-1}}\right)\cdot\left(\frac{\boldsymbol{\tau}_q^{l-1}\cdot\boldsymbol{\psi}^{l-1}}{\boldsymbol{\tau}_q^{l-1}+\boldsymbol{\psi}^{l-1}}\right)$

  $\boldsymbol{\psi}^l = \alpha^2\left(\frac{\boldsymbol{\tau}_q^{l-1}\cdot\boldsymbol{\psi}^{l-1}}{\boldsymbol{\tau}_q^{l-1}+\boldsymbol{\psi}^{l-1}}\right) + (1-\alpha^2)/\hat{\boldsymbol{\gamma}}^{(t)}$

end

for $l = 1, ..., L$

  ${\boldsymbol{\tau}_p}^l = \tau_x^{l(t)}\boldsymbol{\lambda}$

  $\mathbf{p}^l = \boldsymbol{\Phi}\hat{\mathbf{x}}^{l(t)} - \boldsymbol{\tau}_p^l\cdot\mathbf{s}^{l(t-1)}$

  $\mathbf{v}_h^l = \boldsymbol{\tau}_p^l./(\mathbf{1} + \hat{\beta}\boldsymbol{\tau}_p^l)$

  $\hat{\mathbf{h}}^l = (\hat{\beta}\boldsymbol{\tau}_p^l\cdot\mathbf{r}^l + \mathbf{p}^l)./(\mathbf{1} + \hat{\beta}\boldsymbol{\tau}_p^l)$

end

$\hat{\beta} = LM/(\sum_l(\|\mathbf{r}^l - \hat{\mathbf{h}}^l\|^2 + \mathbf{1}^H\mathbf{v}_h^l))$

for $l = 1, ..., L$

  $\boldsymbol{\tau}_s^l = \mathbf{1}./(\boldsymbol{\tau}_p^l + \hat{\beta}^{-1}\mathbf{1})$

  $\mathbf{s}^{l(t)} = \boldsymbol{\tau}_s^l\cdot(\mathbf{r}^l - \mathbf{p}^l)$

  $1/\tau_q^l = (1/N)\boldsymbol{\lambda}^H\boldsymbol{\tau}_s^l$

  $\mathbf{q}^l = \hat{\mathbf{x}}^{l(t)} + \tau_q^l(\boldsymbol{\Phi}^H\mathbf{s}^{l(t)})$

  $\tau_x^{l(t+1)} = (1/N)\mathbf{1}^H(\mathbf{1}./(\mathbf{1}./\boldsymbol{\tau}_q^l + \mathbf{1}./\boldsymbol{\phi}^l + \mathbf{1}./\boldsymbol{\psi}^l))$

  $\hat{\mathbf{x}}^{l(t+1)} = \tau_x^{l(t+1)}(\mathbf{q}^l./\boldsymbol{\tau}_q^l + \boldsymbol{\theta}^l./\boldsymbol{\phi}^l + \boldsymbol{\xi}^l./\boldsymbol{\psi}^l)$

end

$\boldsymbol{\theta}^{L-1} = \frac{1}{\alpha}\mathbf{q}^L$

$\boldsymbol{\phi}^{L-1} = \frac{1}{\alpha^2}\left(\boldsymbol{\tau}_q^L + (1-\alpha^2)/\hat{\boldsymbol{\gamma}}^{(t)}\right)$

for $l = L-2, ..., 1$

  $\boldsymbol{\theta}^l = \frac{1}{\alpha}\left(\frac{\mathbf{q}^{l+1}}{\boldsymbol{\tau}_q^{l+1}} + \frac{\boldsymbol{\theta}^{l+1}}{\boldsymbol{\phi}^{l+1}}\right)\cdot\left(\frac{\boldsymbol{\tau}_q^{l+1}\boldsymbol{\phi}^{l+1}}{\boldsymbol{\tau}_q^{l+1}+\boldsymbol{\phi}^{l+1}}\right)$

  $\boldsymbol{\phi}^l = \frac{1}{\alpha^2}\left(\frac{\boldsymbol{\tau}_q^{l+1}\boldsymbol{\phi}^{l+1}}{\boldsymbol{\tau}_q^{l+1}+\boldsymbol{\phi}^{l+1}} + (1-\alpha^2)/\hat{\boldsymbol{\gamma}}^{(t)}\right)$

end

$\hat{\boldsymbol{\gamma}}^{(t+1)} = L(2\epsilon'+1)/[|\hat{\mathbf{x}}^{1(t+1)}|^2 + \tau_x^{1(t+1)}\mathbf{1}$
$+\frac{1}{1-\alpha^2}\sum_{l=2}^{L}(|\hat{\mathbf{x}}^{l(t+1)}|^2 + \tau_x^{l(t+1)}\mathbf{1})$
$+\frac{\alpha^2}{1-\alpha^2}\sum_{l=1}^{L-1}(|\hat{\mathbf{x}}^{l(t+1)}|^2 + \tau_x^{l(t+1)}\mathbf{1}) - \frac{2\alpha}{1-\alpha^2}\sum_{l=2}^{L}(\hat{\mathbf{x}}^{l(t+1)}\cdot\hat{\mathbf{x}}^{(l-1)(t+1)})]$

$\epsilon' = \frac{1}{2}\sqrt{\log(\frac{1}{N}\sum_n\hat{\gamma}_n^{(t+1)}) - \frac{1}{N}\sum_n\log\hat{\gamma}_n^{(t+1)}}$

$t = t + 1$

while $\frac{1}{L}\sum_{l=1}^{L}(\|\hat{\mathbf{x}}^{l(t+1)} - \hat{\mathbf{x}}^{l(t)}\|^2/\|\hat{\mathbf{x}}^{l(t+1)}\|^2) > \delta_x$ and $t < t_{max}$

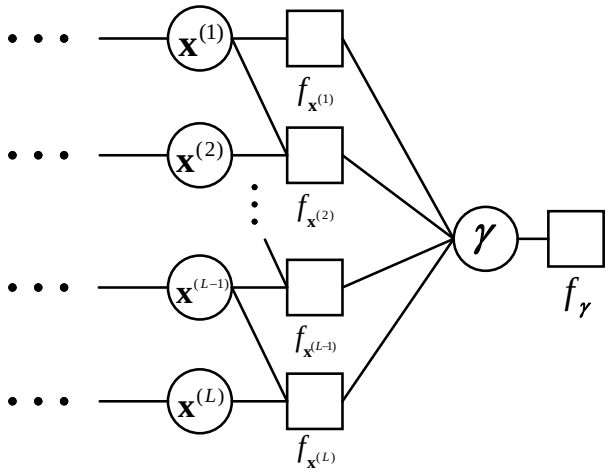

Figure 4.7: The additional factor graph for deriving UAMP-TSBL.

MMV case, where no temporal correlation between non-zero elements is assumed. As the correlation is considered, the differences from the UAMP-SBL MMV algorithm lie in the computations of $\hat{\gamma}_n$ and $\mathbf{x}^{(l)}$.

As in [8], an AR(1) process is use [77] to model the correlation between $x_n^{(l)}$ and $x_n^{(l-1)}$, i.e.,

$$\begin{aligned} x_n^{(l)} &= \alpha x_n^{(l-1)} + \sqrt{1-\alpha^2}\vartheta_n^{(l)} \\ p(x_n^{(l)}|x_n^{(l-1)}) &= \mathcal{N}(x_n^{(l)}|\alpha x_n^{(l-1)}, (1-\alpha^2)\gamma_n^{-1}), l > 1 \\ p(x_n^{(1)}) &= \mathcal{N}(x_n^{(1)}|0, \gamma_n^{-1}), \end{aligned} \tag{4.101}$$

where $\alpha \in (-1, 1)$ is the temporal correlation coefficient and $\vartheta_n^{(l)} \sim \mathcal{N}(0, \gamma_n^{-1})$. Due to the temporal correlation, the conditional prior distribution for the vector $\mathbf{x}^{(l)}$ changes. The factors $\{f_{x_n^{(l)}}(x_n^{(l)}, \gamma_n)\}$ is redefined, i.e., $f_{x_n^{(l)}}(x_n^{(l)}, \gamma_n) = p(x_n^{(l)}|x_n^{(l-1)})$ for $l > 1$ and $f_{x_n^{(1)}}(x_n^{(1)}, \gamma_n) = p(x_n^{(1)})$. Thus, each $x_n^{(l)}$ is connected to the factor nodes $f_{x_n}^{(l)}(x_n^{(l)}|\gamma_n)$, $f_{x_n}^{(l+1)}(x_n^{(l+1)}|\gamma_n)$ and $\{f_{\delta_m}^{(l)}(h_m^{(l)}|\mathbf{x}^{(l)}), \forall m\}$. The factor graph characterizing the temporal correlation is shown in Figure 4.7. The remaining part of the graph is omitted as it is the same as that of the MMV case without temporal correlation. The derivation of the extra message passing for the UAMP-TSBL algorithm is shown in the following, and the algorithm is summarized in **Algorithm** 6. UAMP-TSBL is an extension of the UAMP-SBL algorithm for MMV (**Algorithm** 5). The complexity of the UAMP-TSBL algorithm is also dominated by matrix-vector multiplications, and it is $O(MNL)$ per iteration.

We only derive the message passing for the graph shown in Figure 4.7. The message

$\mathcal{M}_{f_{x_n}^{(l)}\to x_n^{(l)}}(x_n^{(l)})$ is computed by the BP rule with the product of messages $\{\mathcal{M}_{f_{\delta m}^{(l-1)}\to x_n^{(l-1)}}(x_n^{(l-1)}), \forall m\}$ defined in UAMP and message $\{\mathcal{M}_{f_{\delta_m}^{(l-1)}\to x_n^{(l-1)}}(x_n^{(l-1)})\}$, i.e.,

$$\begin{aligned}
&\mathcal{M}_{f_{x_n}^{(l)}\to x_n^{(l)}}(x_n^{(l)}) \\
&= \left\langle f_{x_n^{(l)}}(x^{(l)}) \right\rangle_{\mathcal{M}_{f_{x_n}^{(l-1)}\to x_n^{(l-1)}} \prod_m \mathcal{M}_{f_{\delta_m}^{(l-1)}\to x_n^{(l-1)}}} \\
&\propto \mathcal{N}(x_n^{(l)}|\xi_n^{(l)}, \psi_n^{(l)}),
\end{aligned} \tag{4.102}$$

which leads to Lines 1 to 6 of the UAMP-TSBL algorithm. Similarly, the message $\mathcal{M}_{f_{x_n}^{(l+1)}\to x_n^{(l)}}(x_n^{(l)})$ from factor node $f_{x_n}^{(l+1)}$ to variable node $x_n^{(l)}$ is also updated by the BP rule

$$\begin{aligned}
&\mathcal{M}_{f_{x_n}^{(l+1)}\to x_n^{(l)}}(x_n^{(l)}) \\
&= \left\langle f_{x_n^{(l+1)}}(x^{(l+1)}) \right\rangle_{\mathcal{M}_{f_{x_n}^{(l+2)}\to x_n^{(l+1)}} \prod_m \mathcal{M}_{f_{\delta_m}^{(l+1)}\to x_n^{(l+1)}}} \\
&\propto \mathcal{N}(x_n^{(l)}|\theta_n^{(l)}, \phi_n^{(l)}),
\end{aligned} \tag{4.103}$$

leading to Lines 22 to 27 of the UAMP-TSBL algorithm. We compute the belief of variable $x_n^{(l)}$ by

$$\begin{aligned}
b(x_n^{(l)}) &\propto \mathcal{M}_{f_{x_n}^{(l)}\to x_n^{(l)}} \mathcal{M}_{f_{x_n}^{(l+1)}\to x_n^{(l)}} \prod_m \mathcal{M}_{f_{\delta m}^{(l)}\to x_n^{(l)}} \\
&\propto \mathcal{N}(x_n^{(l)}|\hat{x}_n^{(l)}, \tau_x^{(l)})
\end{aligned} \tag{4.104}$$

leading to Lines 19 to 20 of the UAMP-TSBL algorithm. With the beliefs $b(x_n^{(l)})$ and $b(x_n^{(l-1)})$, the message $\mathcal{M}_{f_{x_n}^{(l)}\to\gamma_n}(\gamma_n)$ can be obtained as

$$\mathcal{M}_{f_{x_n}^{(l)}\to\gamma_n}(\gamma_n) = \exp\left\{\left\langle f_{x_n}^{(l)}(x_n^{(l)}|\gamma_n)\right\rangle_{b(x_n^{(l)})b(x_n^{(l-1)})}\right\}. \tag{4.105}$$

Then, with the message $\mathcal{M}_{f_{\gamma_n}\to\gamma_n}(\gamma_n)$ in (4.42), the belief $b(\gamma_n)$

$$b(\gamma_n) \propto \mathcal{M}_{f_{\gamma_n}\to\gamma_n}(\gamma_n)\mathcal{M}_{f_{x_n}\to\gamma_n}(\gamma_n). \tag{4.106}$$

Then the update of $\hat{\gamma}_n$ can be expressed as

$$\begin{aligned}\hat{\gamma}_n = L(2\epsilon' + 1)/(|\hat{\mathbf{x}}_n^{(1)}|^2 + \tau_x^{(1)} + \frac{1}{\alpha^2}\sum_{l=2}^{L}(|\hat{\mathbf{x}}_n^{(l)}|^2 + \tau_x^{(l)}) \\ + \frac{\alpha^2}{1-\alpha^2}\sum_{l=1}^{L-1}(|\hat{\mathbf{x}}_n^{(l)}|^2 + \tau_x^{(l)}) - \frac{2\alpha}{1-\alpha^2}\sum_{l=2}^{L}(\hat{\mathbf{x}}_n^{(l)}\hat{\mathbf{x}}_n^{(l-1)})).\end{aligned} \tag{4.107}$$

4.6 Numerical results

In this section, the proposed UAMP-(T)SBL algorithms is compared with the conventional SBL and state-of-the-art AMP-based SBL algorithms. The performance of various algorithms using normalized MSE is evaluated, and is defined as

$$\text{NMSE} \triangleq \frac{1}{K}\sum_{k=1}^{K}||\hat{\mathbf{x}}_k - \mathbf{x}_k||^2/||\mathbf{x}_k||^2, \tag{4.108}$$

$$\text{NMSE} \triangleq \frac{1}{KL}\sum_{k=1}^{J}\sum_{l=1}^{L}||\hat{\mathbf{x}}_k^{(l)} - \mathbf{x}_k^{(l)}||^2/||\mathbf{x}_k^{(l)}||^2 \tag{4.109}$$

for the SMV and MMV cases respectively, where $\hat{\mathbf{x}}_k$ ($\hat{\mathbf{x}}_k^{(l)}$) is the estimate of $\mathbf{x}_k$ ($\mathbf{x}_k^{(l)}$), and K is the number of trials. Since different algorithms have different computational complexity per iteration and they require a different number of iterations to converge, as in [8], the runtime of the algorithms to indicate their relative computational complexity is measured. It is noted that the time consumed by the SVD in UAMP-SBL is counted for the runtime.

To test the robustness and performance of the algorithms, the following measurement matrices is used:

1. Ill-conditioned Matrix: Matrix $\mathbf{A}$ is constructed based on the SVD $\mathbf{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}$ where $\boldsymbol{\Lambda}$ is a singular value matrix with $\Lambda_{i,i}/\Lambda_{i+1,i+1} = \kappa^{1/(M-1)}$ for $i = 1, 2, ..., M-1$ (i.e., the condition number of the matrix is κ).

2. Correlated Matrix: The correlated matrix $\mathbf{A}$ is constructed using $\mathbf{A} = \mathbf{C}_L^{1/2}\mathbf{G}\mathbf{C}_R^{1/2}$, where $\mathbf{G}$ is an i.i.d. Gaussian matrix with mean zero and unit variance, and $\mathbf{C}_L$ is an $M \times M$ matrix with the (m, n)th element given by $c^{|m-n|}$ where $c \in [0, 1]$. Matrix $\mathbf{C}_R$ is generated in the same way but with a size of $N \times N$. The parameter c controls

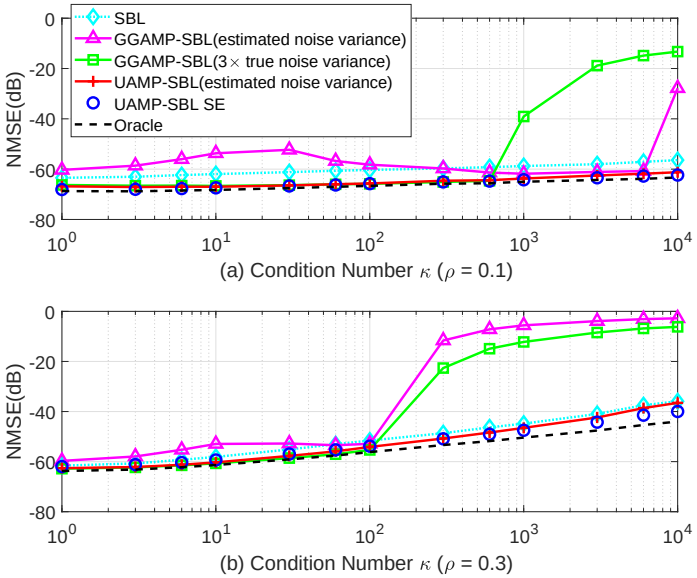

Figure 4.8: Performance comparison (ill-conditioned matrices).

the correlation of matrix **A**.

3. Non-zero Mean Matrix: The elements of matrix **A** are drawn from a non-zero mean Gaussian distribution, i.e., $a_{m,n} \sim \mathcal{N}(a_{m,n}|\mu, 1)$. The mean μ measures the derivation from the i.i.d. zero-mean Gaussian matrix.

4. Low Rank Matrix: The measurement matrix $\mathbf{A} = \mathbf{BC}$, where the size of **B** and **C** are $M \times R$ and $R \times N$, respectively, and $R < M$. Both **B** and **C** are i.i.d. Gaussian matrices with mean zero and unit variance. The rank ratio R/N is used to measure the deviation of matrix **A** from the i.i.d. Gaussian matrix.

4.6.1 Numerical Results for SMV

In this section, UAMP-SBL against the conventional SBL [6] and the state-of-the-art AMP based SBL algorithm GGAMP-SBL [8] with estimated noise variance and 3 times of the true noise variance are compared. The vector **x** is drawn from a Bernoulli-Gaussian distribution with a non-zero probability ρ. The SNR is defined as SNR $\triangleq E\,\|\mathbf{Ax}\|^2 / E\,\|\mathbf{w}\|^2$. As a performance benchmark, the support-oracle MMSE bound [8] is also included. $M = 800$, $N = 1000$ and the SNR is set to be 60dB, unless it is specified are set. For UAMP-SBL the maximum iteration number $t_{max} = 300$ (note that there is no inner iter-

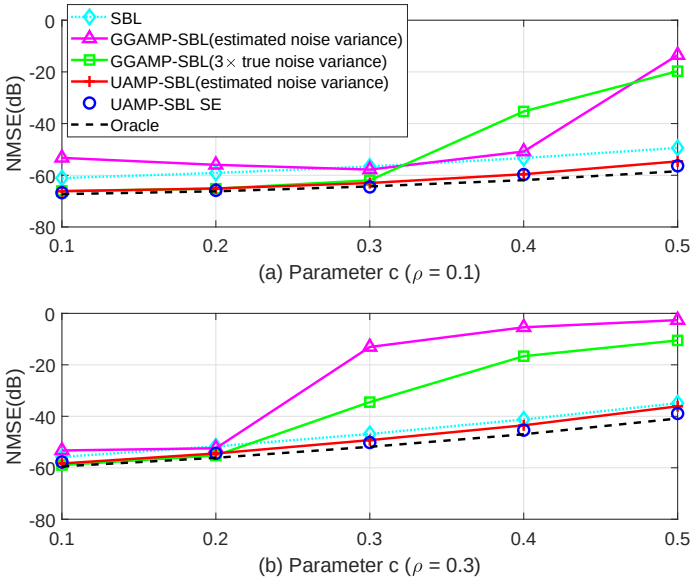

Figure 4.9: Performance comparison (correlated matrices).

ation in UAMP-SBL) are set. GGAMP-SBL is a double loop algorithm, the maximum numbers of E-step and outer iteration are set to be 50 and 1000 respectively. The damping factor for GGAMP-SBL is 0.2 to enhance its robustness against tough measurement matrices. It is noted that the damping factor can be increased to reduce the runtime of GGAMP-SBL but at the cost of reduced robustness.

In Figure 4.8, the performance of various algorithms in terms of NMSE versus the condition number is shown in (a) for a sparsity rate of $\rho = 0.1$ and (b) for a sparsity rate of $\rho = 0.3$. It can be seen from Figure 4.8(a) that UAMP-SBL delivers the best performance (even better than the conventional SBL algorithm), which closely approaches the support-oracle bound. With a larger sparsity rate in Figure 4.8(b), UAMP-SBL still exhibits excellent performance and it performs slightly better than SBL and significantly better than GGAMP-SBL when the condition number is relatively large. In addition, the simulation performance of UAMP-SBL matches well with the performance predicted with SE.

Figure 4.9 shows the performance of various algorithms versus a range of correlation parameter c from 0.1 to 0.5, where the sparsity rate $\rho = 0.1$ in (a) and $\rho = 0.3$ in (b). From this figure, it can be seen that, UAMP-SBL still delivers exceptional performance, which is better than SBL and significantly better than GGAMP-SBL when the correlation

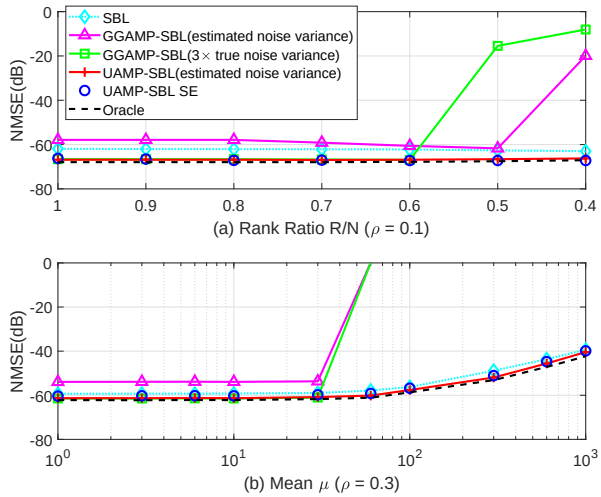

Figure 4.10: Performance comparison: (a)low rank matrices; (b) non-zero mean matrices.

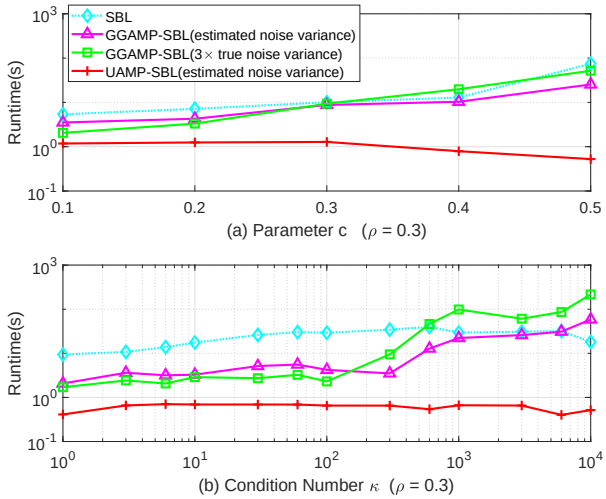

Figure 4.11: Runtime of algorithms under ill-conditioned matrices with $\rho = 0.3$.

parameter c is relatively large. The gap between UAMP-SBL and GGAMP-SBL becomes more notable with a higher sparsity rate. The performance of UAMP-SBL matches well with SE again.

In Figure 4.10, it is examined by the performance of the algorithms versus rank ratio in (a), where the sparsity rate $\rho = 0.1$, and versus non-zero mean in (b), where the sparsity rate $\rho = 0.3$. It can be seen that UAMP-SBL still delivers performance which closely matches the support-oracle bound, and is slightly better than that of SBL. It also can be seen that GGAMP-SBL diverges when the mean μ is relatively large. The performance of UAMP-SBL matches well with SE as well.

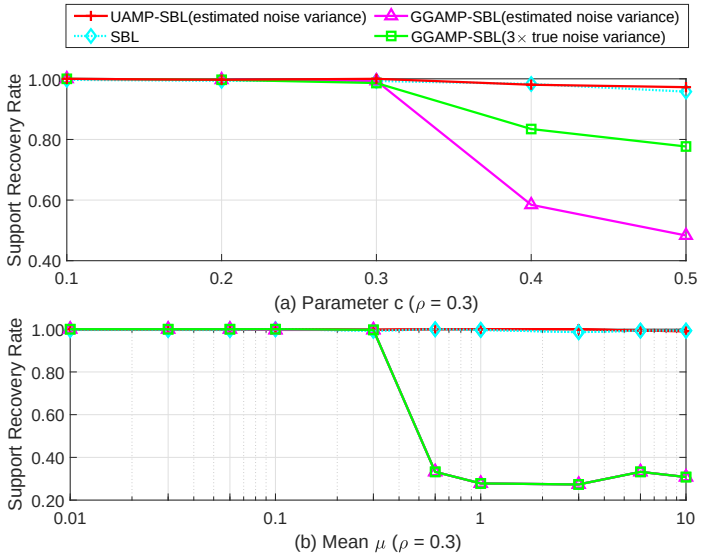

Figure 4.12: Support recovery rate comparison: (a) low rank matrices; (b) non-zero mean matrices.

The average runtime of various algorithms is shown in Figure 4.11, where the sparsity rate $\rho = 0.3$, and the measurement matrice are correlated in (a) and ill-conditioned in (b). It can be seen that UAMP-SBL is much faster than GGAMP-SBL and SBL. SBL is normally the slowest as it has the highest complexity due to the matrix inverse in each iteration. It is noted that, for GGAMP-SBL, the damping factor to be relatively small value 0.2 is set to enable it to achieve better performance and robustness. If the damping factor is increased, GGAMP-SBL could become faster but at the cost of offsetting its performance and robustness.

In summary, when the deviation of the measurement matrices from the i.i.d. zero-mean Gaussian matrix is small, GGAMP-SBL (with 3× true noise variance) and UAMP-SBL deliver similar performance, and both of them can achieve the support-oracle bound. However, when the deviation is relatively large, UAMP-SBL can significantly outperform GGAMP-SBL (UAMP-SBL can still approach the support oracle bound closely in many cases), which demonstrates that UAMP-SBL is much more robust. In addition, UAMP-SBL is also much faster. Meanwhile, the simulation performance of UAMP-SBL matches well with SE.

In Figure 4.12, the support recovery rate of the algorithms versus correlation parameter c for correlation matrices in (a) and mean value μ for non-zero mean matrices in (b), where the sparse rate $\rho = 0.3$ is evaluated. The support recovery rate is defined as the

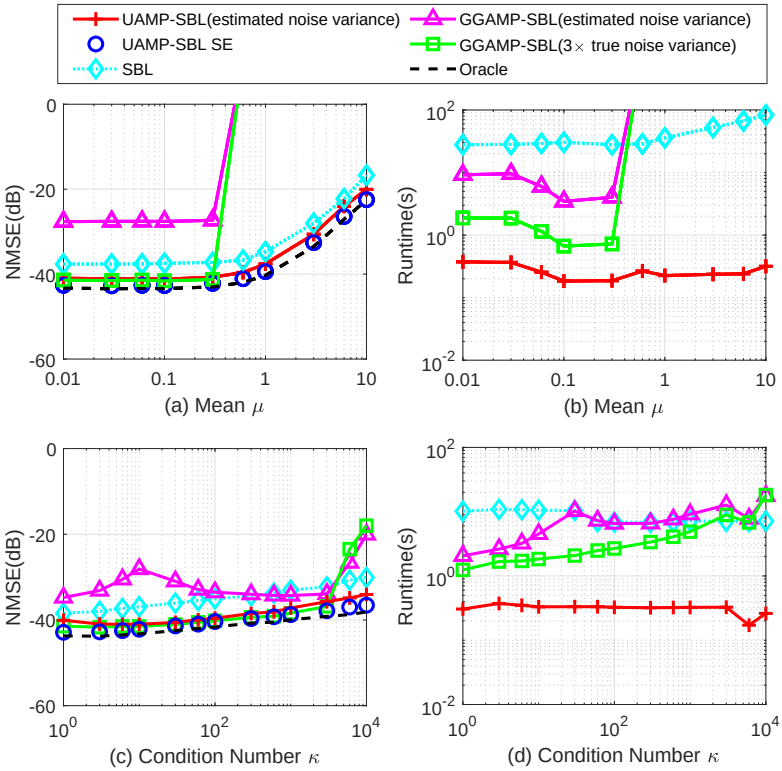

Figure 4.13: Performance and runtime comparisons of various algorithms where SNR = 35dB.

percentage of successful trials in the total trials [78]. In the noiseless case, a successful trial is recorded if the indexes of estimated non-zero signal elements are the same as the true indexes. In the noisy case, as the true sparse vector cannot be recovered exactly, the recovery is regarded to be successful if the indexes of the estimated elements with the $\mathcal{K}$ largest absolute values are the same as the true indexes of non-zero elements in the sparse vector $\mathbf{x}$, where $\mathcal{K}$ is the number of non-zero elements in $\mathbf{x}$. From the results, It can seen that UAMP-SBL and SBL deliver similar performance and they can significantly outperform GGAMP-SBL when c or μ is relatively large.

The performance of various algorithms at SNR = 35dB, and the NMSE performance and runtime of the algorithms are shown in Figure 4.13, where (a) and (b) are for non-zero mean matrices, and (c) and (d) are for ill-conditioned matrices are compared. The sparsity rate $\rho = 0.1$. Again, it can be seen that, compared to GGAMP-SBL, UAMP-SBL delivers better performance with considerably much smaller runtime when the mean or condition number of the matrices are relatively large. The performance of various algorithms versus SNR can be shown in Figure 4.14, where the matrices are highly ill-conditioned with a

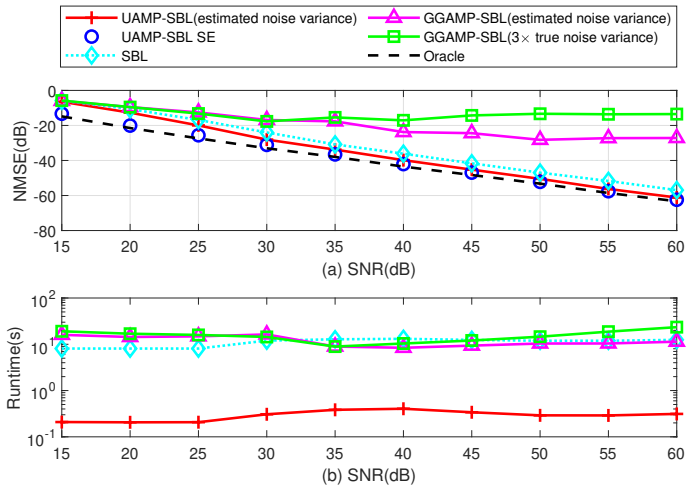

Figure 4.14: Performance and runtime comparisons of various algorithms for highly ill-conditioned matrices.

condition number $\kappa = 10^4$. It can be seen that UAMP-SBL performs better and is faster than SBL and GGAMP-SBL.

The key difference between AMP and UAMP is that a unitary transformation is performed in UAMP, which makes UAMP much more robust against a generic measurement matrix. Inspired by this, the impact of the unitary transformation on the GGAMP-SBL algorithm is tested, where the unitary transformation to the original model is perform and then GGAMP-SBL is carried out.this algorithm is called UT-GGAMP-SBL, and is compared with UAMP-SBL in the case of correlated matrices. The performance and the corresponding runtime are shown in Figure 4.15, where the hyper-parameter ϵ of UT-GGAMP-SBL is not updated in (a) and (b) while updated in (c) and (d). It can be seen that, thanks to the unitary transformation, the stability of GGAMP-SBL can be significantly improved as expected. Figure 4.15 (a) shows that UT-GGAMP-SBL with 3 times true noise variance achieves almost the same performance as UAMP-SBL, however, UT-GGAMP-SBL requires the knowledge of noise variance and it is significantly slower than UAMP-SBL. Figure 4.15 (c) shows that updating ϵ is not helpful for UT-GGAMP-SBL. UT-GGAMP-SBL with estimated noise variance simply diverges (so its performance is not shown). UT-GGAMP-SBL with 3 times true noise variance is inferior to UAMP-SBL when c is relatively large. Again, UAMP-SBL is much faster.

In Figure 4.16, UAMP-SBL with VAMP-EM in [79] is compared. In VAMP-EM, Bernoulli-Gaussian priors are employed and the parameters are learned using EM. The

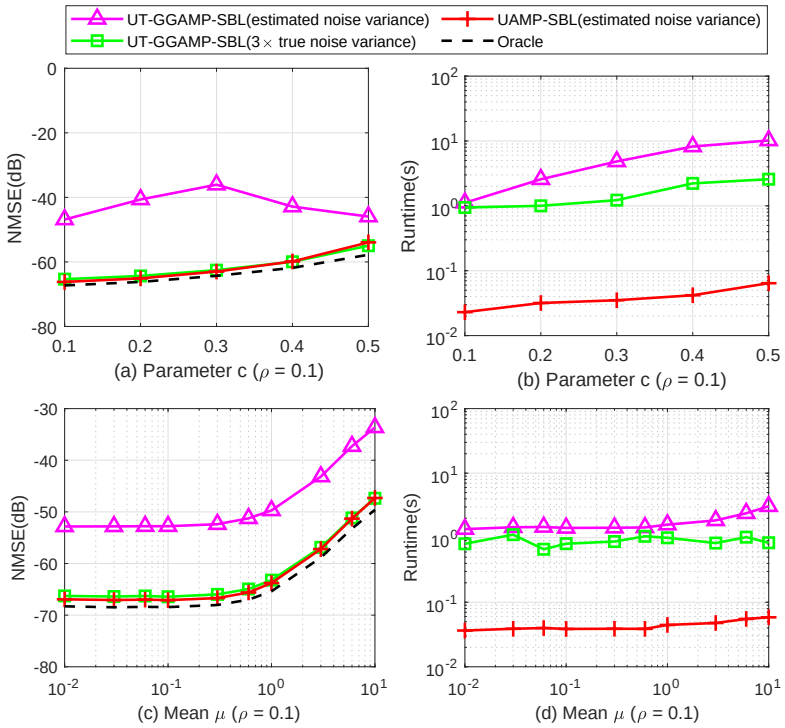

Figure 4.15: Performance and runtime comparisons of UAMP-SBL and UT-GGAMP-SBL.

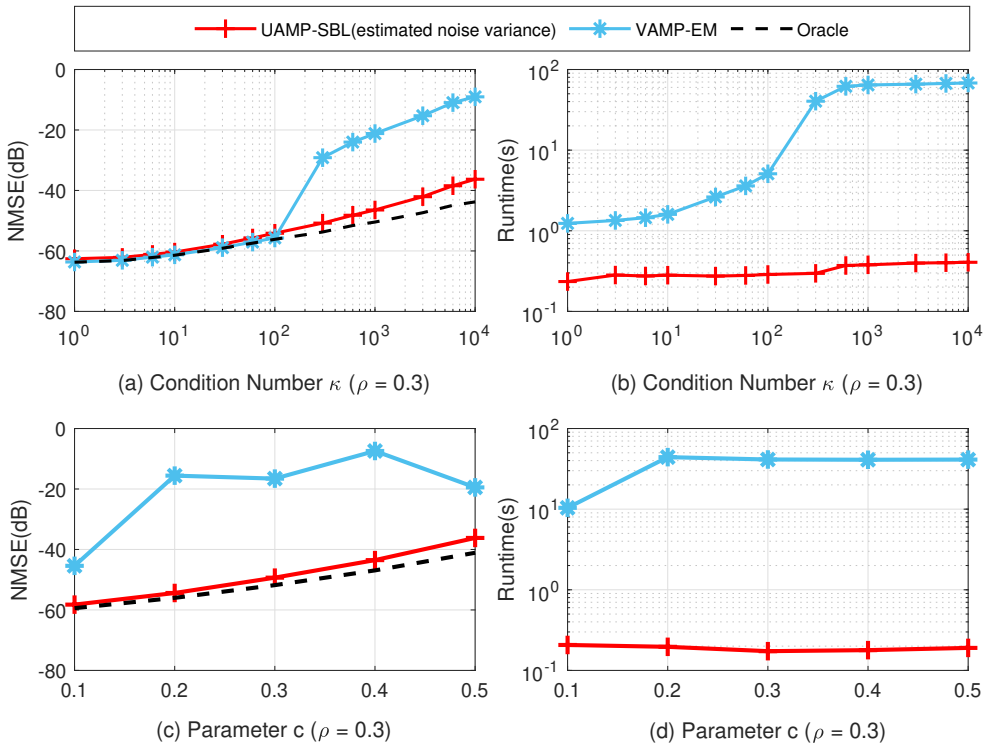

Figure 4.16: Performance and runtime comparisons of UAMP-SBL and VAMP-EM.

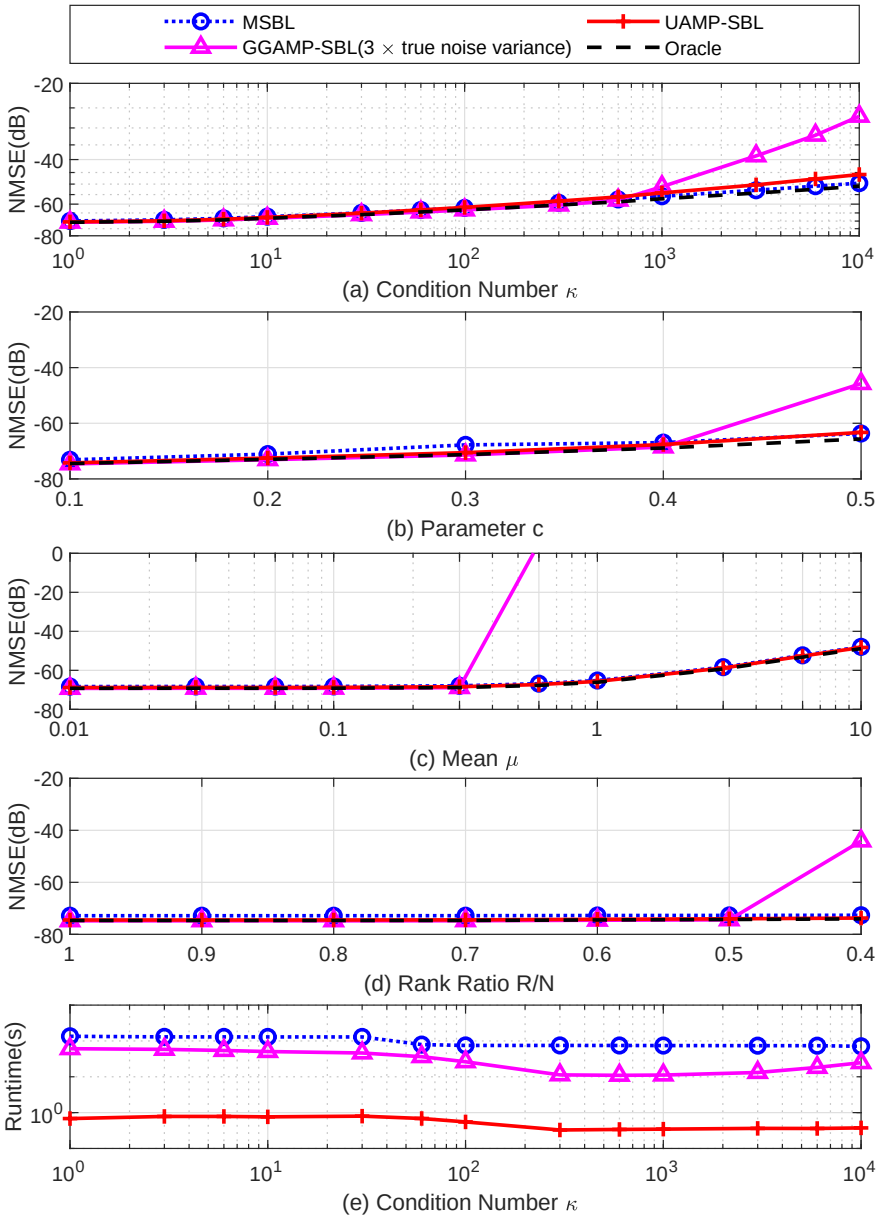

Figure 4.17: Performance comparison of various algorithms in the case of MMV.

NMSE performance and runtime are shown in Figure 4.16, where (a) and (b) are for ill-conditioned matrices, and (c) and (d) are for correlated matrices. The sparsity rate $\rho = 0.3$. It can seen that, compared to VAMP-EM, UAMP-SBL can deliver better performance while runs faster.

4.6.2 Numerical Results for MMV

The elements of the sparse vectors $\left\{\mathbf{x}^{(l)}, l = 1 : L\right\}$ are drawn from a Bernoulli-Gaussian distribution, and the vectors share a common support. The number of measurement vec-

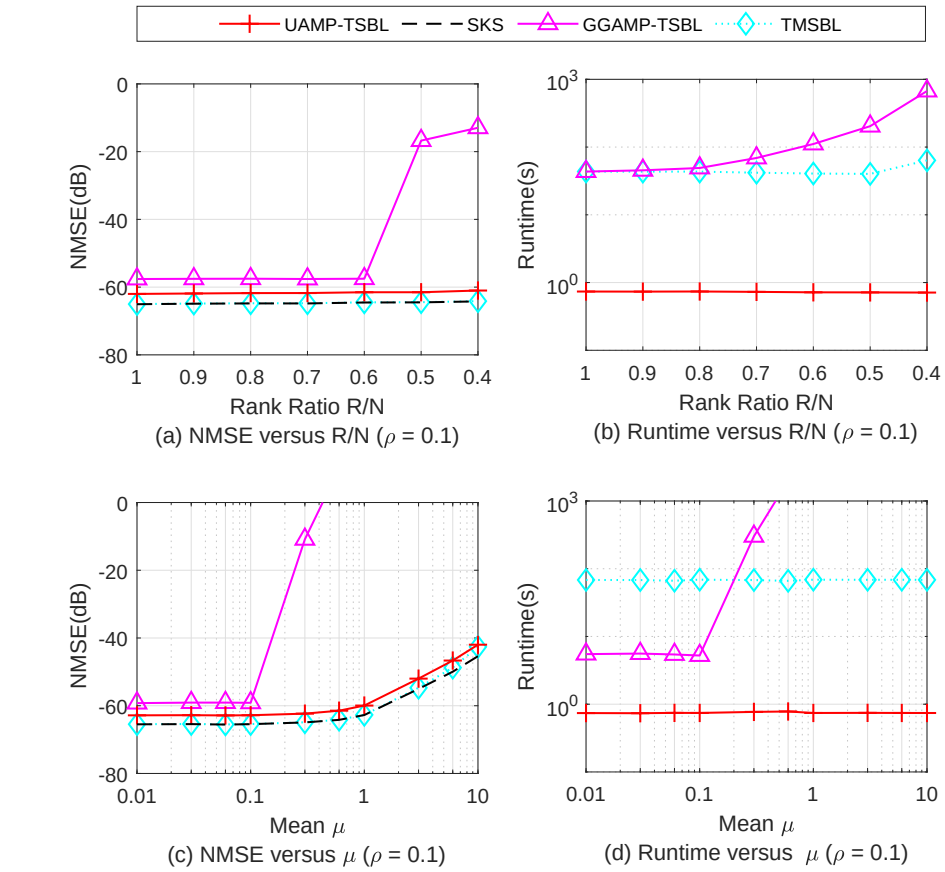

Figure 4.18: Performance comparison of various algorithms in the case of MMV with temporal correlation.

tors is 5. The performance of the algorithms with ill-conditioned, correlated, non-zero mean and low-rank measurement matrices is shown in Figure 4.17 (a)-(d), respectively. In this figure, the performance of the direct extension of the conventional SBL algorithm to the MMV model (MSBL) [80] and support-oracle bound is also included.

It can be seen from this figure that, when the deviation of the measurement matrices from the i.i.d. zero-mean Gaussian matrix is small, GGAMP-SBL (with 3× true noise variance) and UAMP-SBL deliver similar performance, and both of them can approach the bound closely. MSBL works slightly worse than GGAMP-SBL and UAMP-SBL. However, when the deviation is relatively large, MSBL delivers slightly better performance but at high complexity. In most cases, UAMP-SBL and MSBL almost have the same performance, and can significantly outperform GGAMP-SBL. As an example, It can be shown the average runtime of different algorithms in the case of ill-conditioned matrices in Figure 4.17(e), where UAMP-SBL converges significantly faster than GGAMP-

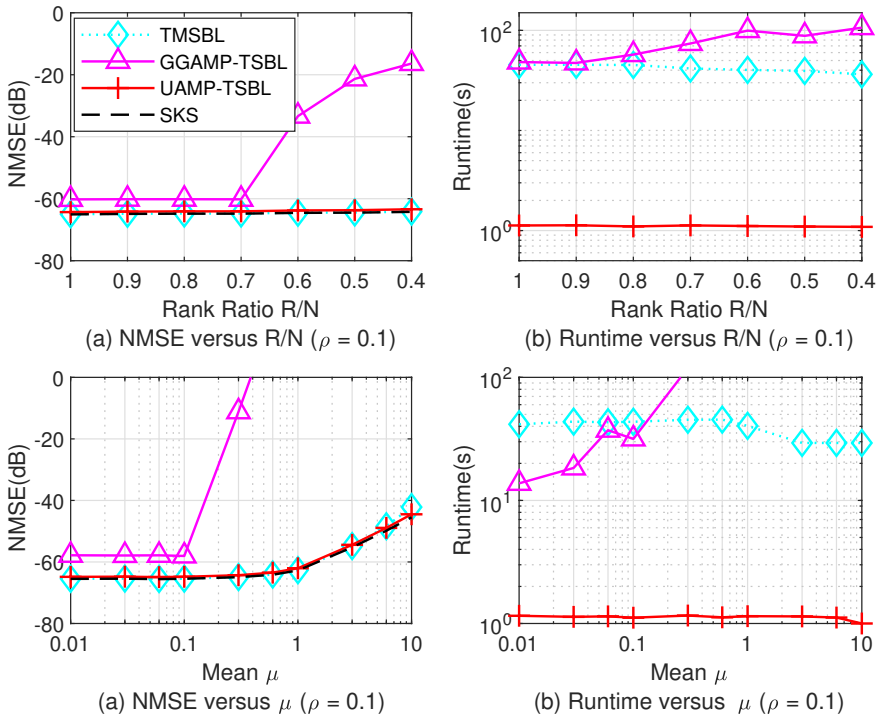

Figure 4.19: Performance comparison of various algorithms in the case of MMV with temporal correlation.

SBL and MSBL.

Furthermore, a numerical study is presented to illustrate the performance of UAMP-SBL when incorporating the temporal correlation. Besides the temporally correlated SBL (TMSBL) [78] and GGAMP-SBL, the recovery performance with a lower bound: the achievable NMSE by a support-aware Kalman smoother (SKS) [81] with the knowledge of the support of the sparse vectors and the true values of β, α and $\boldsymbol{\gamma}$ is also compared. The SKS is implemented in a more efficient way by incorporating UAMP. As examples, low rank and non-zero mean measurement matrices are used to test their performance. The sparsity rate $\rho = 0.1$, SNR = 50dB and the temporal correlation coefficient $\alpha = 0.8$ in Figure 4.18 and the temporal correlation coefficient $\alpha = 0.6$ in Figure 4.19. It can be seen from Figure 4.18 and Figure 4.19 that, UAMP-TSBL can approach the bound closely and outperform other algorithms significantly when the rank ratio is relatively low and the mean is relatively high. In addition, UAMP-TSBL is much faster.

4.7 Conclusion

In this paper, leveraging UAMP, UAMP-SBL for sparse signal recovery with the framework of structured variational inference is proposed, which inherits the low complexity and robustness of UAMP against a generic measurement matrix. It is demonstrated that, compared to the state-of-the-art AMP based SBL algorithm, UAMP-SBL can achieve much better performance in terms of robustness, speed and recovery accuracy.

Chapter 5

UAMP for Bilinear Recovery

5.1 Introduction

The problem of bilinear recovery with model $\mathbf{Y} = \sum_{k=1}^{K} b_k \mathbf{A}_k \mathbf{C} + \mathbf{W}$, where $\{b_k\}$ and $\mathbf{C}$ are jointly recovered with known $\mathbf{A}_k$ from the noisy measurements $\mathbf{Y}$ is considered. When $\mathbf{Y}$, $\mathbf{C}$ and $\mathbf{W}$ are replaced with the corresponding vectors $\mathbf{y}$, $\mathbf{c}$ and $\mathbf{w}$, respectively, the above multiple measurement vector (MMV) problem is reduced to a single measurement vector (SMV) problem. Model (2.7) covers a variety of problems, e.g., compressed sensing (CS) with matrix uncertainty [11], joint channel estimation and detection [45], self-calibration and blind deconvolution [10], and structured dictionary learning [9].

Recently, several approximate message passing (AMP) [69] [82] based algorithms have been developed to solve the bilinear problem, which show promising performance, compared to existing non-message passing alternates [14]. The generalized AMP (GAMP) [15] was extended to bilinear GAMP (BiGAMP) [16] for solving a general bilinear problem, i.e., recover both $\mathbf{A}$ and $\mathbf{X}$ from observation $\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{W}$. The parametric BiGAMP (P-BiGAMP) is then proposed in [17], which works with model (2.7) to jointly recover $\{b_k\}$ and $\mathbf{C}$. Lifted AMP was proposed in [18] by using the lifting approach [19], [10]. However, these AMP based algorithms are vulnerable to difficult $\mathbf{A}$ matrices, e.g., ill-conditioned, correlated, rank-deficient or non-zero mean matrices as AMP can easily diverge in these cases [20].

In this work, leveraging UAMP, we propose a more robust and faster approximate in-

ference algorithm for bilinear recovery, which is called Bi-UAMP. By using the lifting approach, the original bilinear problem is reformulated as a linear one. Then, the structured variational inference (VI) [50], [52], [53], expectation propagation (EP) [40] and belief propagation (BP) [83], [84] are combined with UAMP, where UAMP is employed to handle the most computational intensive part, leading to the fast and robust approximate inference algorithm Bi-UAMP. It is shown that Bi-UAMP performs significantly better and is much faster than state-of-the-art bilinear recovery algorithms for difficult matrices.

5.1.1 Chapter's Organization

The organization of the chapter is as follows. In Section 5.2, the Bi-UAMP algorithm is introduced. Bi-UAMP is designed for SMV problems. In Section 5.3 Bi-UAMP is then extended for MMV problems and its properties are investigated. Numerical examples and comparisons with state-of-the-art message passing and non-message passing algorithms are provided in Section 5.4, and conclusions are drawn in Section 5.5.

5.2 Bilinear UAMP

5.2.1 Problem Formulation

Different from [11], a Bayesian treatment of the bilinear recovery problem is considered

$$\mathbf{y} = \sum_{k=1}^{K} b_k \mathbf{A}_k \mathbf{c} + \mathbf{w}, \tag{5.1}$$

where $\mathbf{b} \triangleq [b_1, ..., b_K]^T$, $\mathbf{c}$ and β (the precision of the noise) are random variables with priors $p(\mathbf{b})$, $p(\mathbf{c})$ and $p(\beta)$, respectively. It is noted that, in the case of no a priori information available, $p(\mathbf{b})$, $p(\mathbf{c})$ and $p(\beta)$ can be simply chosen as non-informative priors. This also differs from the development of BAd-VAMP in [14], where both $\mathbf{b}$ and β are treated as unknown deterministic variables, and their values are estimated following the framework of expectation maximization (EM). However, a Bayesian treatment of $\mathbf{b}$ is more advantageous. In the case of a priori information available for $\mathbf{b}$, a Bayesian method

enables the use of the a priori information, which may be very helpful to improve the recovery performance. If no a priori information is known, a non-informative prior can be simply used. Moreover, in the context of iterative inference considered in this paper, the Bayesian treatment of $\mathbf{b}$ is also different from that of the EM method in that only a point estimate of $\mathbf{b}$ is involved in the iteration of the EM method, while a distribution of $\mathbf{b}$ is involved in the iterative process of the method with Bayesian treatment. Even in the case of non-informative priors for the method with Bayesian treatment, they are still different in this way normally.. Here, for simplicity, the SMV problem is taken as example, but the extension of our discussion to the case of MMV is straightforward.

The joint conditional distribution of $\mathbf{b}$, $\mathbf{c}$ and β can be expressed as

$$p(\mathbf{b}, \mathbf{c}, \beta|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{b}, \mathbf{c}, \beta)p(\mathbf{b})p(\mathbf{c})p(\beta). \tag{5.2}$$

It is aimed at finding the a posterior distributions $p(\mathbf{b}|\mathbf{y})$ and $p(\mathbf{c}|\mathbf{y})$, and therefore their a posterior means that can be used as their estimates, i.e., $\hat{\mathbf{b}} = \mathbb{E}(\mathbf{b}|\mathbf{y})$ and $\hat{\mathbf{c}} = \mathbb{E}(\mathbf{c}|\mathbf{y})$. However, this is often intractable because high dimensional integration is required to compute the a posteriori distributions $p(\mathbf{b}|\mathbf{y})$ and $p(\mathbf{c}|\mathbf{y})$. As a result, the approximate Bayesian inference techniques is used.

5.2.2 Problem and Model Reformulation for Efficient UAMP-Based Approximate Inference

Similar to the lifting approach, $\mathbf{A} \triangleq [\mathbf{A}_1, ..., \mathbf{A}_K]_{M\times NK}$ are defined, then the original bilinear model can be reformulated as

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{w} \tag{5.3}$$

with the auxiliary variable

$$\mathbf{x} = \mathbf{b} \otimes \mathbf{c} = \begin{pmatrix} b_1\mathbf{c} \\ \vdots \\ b_K\mathbf{c} \end{pmatrix}_{NK\times 1}, \tag{5.4}$$

where $\mathbf{x}$ can be indexed as

$$\mathbf{x} = \left[x_{1,1}, ...x_{N,1}, ..., x_{n,k}, ...x_{N,K}\right]^T \tag{5.5}$$

with

$$x_{n,k} = c_n b_k. \tag{5.6}$$

With an SVD for matrix $\mathbf{A}$, i.e., $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}$, performing unitary transformation yields $\mathbf{r} = \mathbf{\Phi}\mathbf{x}+\boldsymbol{\omega}$, where $\mathbf{r} = \mathbf{U}^H\mathbf{y}$, $\mathbf{\Phi} = \mathbf{\Lambda}\mathbf{V}$ has a size of $M\times NK$, and $\boldsymbol{\omega} = \mathbf{U}^H\mathbf{w}$ is still white and Gaussian with the same precision β. It is noted that performing the unitary transformation here is purely to facilitate the use of UAMP. As $\mathbf{U}^H$ is a unitary matrix, the transformation will not result in any loss. So the resultant algorithms will work with the transformed observation $\mathbf{r}$, instead of $\mathbf{y}$. Then define a new auxiliary variable $\mathbf{z} = \mathbf{\Phi}\mathbf{x}$ as in [85], [48], [35] and [86]. Later, it will be seen that the introduction of the auxiliary variables $\mathbf{x}$ and $\mathbf{z}$ facilitates the integration of UAMP into the approximate Bayesian inference algorithm, which is crucial to achieving efficient and robust inference.

Table 5.1: Distributions and factors in (5.7)

Factor	Distribution	Function
$f_{\mathbf{r}}$	$p\left(\mathbf{r}\vert\mathbf{z},\beta\right)$	$\mathcal{N}\left(\mathbf{z};\mathbf{r},\beta^{-1}I\right)$
$f_{\mathbf{z}}$	$p\left(\mathbf{z}\vert\mathbf{x}\right)$	$\delta\left(\mathbf{z}-\mathbf{\Phi}\mathbf{x}\right)$
$f_{\mathbf{x}}$	$p(\mathbf{x}\vert\mathbf{c},\mathbf{b})$	$\delta\left(\mathbf{x}-\mathbf{b}\otimes\mathbf{c}\right)$
$f_{x_{n,k}}$	$p\left(x_{n,k}\vert b_k, c_n\right)$	$\delta\left(x_{n,k}-b_k c_n\right)$
$f_{\mathbf{c}}$	$p(\mathbf{c})$	prior of $\mathbf{c}$, e.g., prior promoting sparsity
$f_{\mathbf{b}}$	$p(\mathbf{b})$	prior of $\mathbf{b}$
f_{β}	$p(\beta)$	$\propto \beta^{-1}$

With the two latent variables $\mathbf{x}$ and $\mathbf{z}$, the following joint conditional distribution of $\mathbf{c}, \mathbf{b}, \mathbf{x}, \mathbf{z}, \beta$ and its factorization are

$$\begin{aligned} & p(\mathbf{c},\mathbf{b},\mathbf{x},\mathbf{z},\beta|\mathbf{r}) \\ & \quad \propto p(\mathbf{r}|\mathbf{z},\beta)p(\mathbf{z}|\mathbf{x})p(\mathbf{x}|\mathbf{b},\mathbf{c})p(\mathbf{c})p(\mathbf{b})p(\beta) \\ & \quad \triangleq f_{\mathbf{r}}(\mathbf{z},\beta)f_{\mathbf{z}}(\mathbf{z},\mathbf{x})f_{\mathbf{x}}(\mathbf{x},\mathbf{b},\mathbf{c})f_{\mathbf{c}}(\mathbf{c})f_{\mathbf{b}}(\mathbf{b})f_{\beta}(\beta). \end{aligned} \tag{5.7}$$

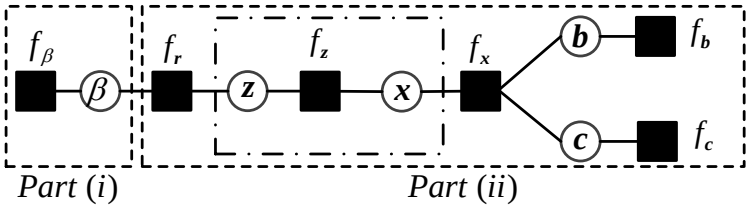

Figure 5.1: Factor graph representation of (5.7).

Hence our aim is to find the a posteriori distributions $p(\mathbf{c}|\mathbf{r})$ and $p(\mathbf{b}|\mathbf{r})$ and their estimates in terms of the a posteriori means, i.e., $\hat{\mathbf{c}} = \mathbb{E}(\mathbf{c}|\mathbf{r})$ and $\hat{\mathbf{b}} = \mathbb{E}(\mathbf{b}|\mathbf{r})$. It seems that, due to the involvement of two extra latent variables $\mathbf{x}$ and $\mathbf{z}$, the use of (5.7) could be more complicated than that of (5.2), but it enables efficient approximate inference by incorporating UAMP, as detailed later. The probability functions and the corresponding factors (to facilitate the factor graph representation) are listed in Table 1, and a factor graph representation of (5.7) is depicted in Figure 5.1.

The framework of structured variational inference (SVI) [50] is followed, which can be formulated nicely as message passing with graphical models [52], [53], [54], [87]. The trial function for the joint conditional distribution function $p(\mathbf{c}, \mathbf{b}, \mathbf{x}, \mathbf{z}, \beta|\mathbf{r})$ in (5.7) is chosen as

$$\tilde{q}(\mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{z}, \beta) = \tilde{q}(\beta)\tilde{q}(\mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{z}). \tag{5.8}$$

The employment of this trial function corresponds to a partition of the factor graph in Figure 5.1 [53], i.e., $\tilde{q}(\beta)$ and $\tilde{q}(\mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{z})$ are associated respectively with the subgraphs denoted by Part (i) and Part (ii), where the variable node β is external to Part (ii). With SVI, the variational lower bound

$$\begin{aligned} &\mathcal{L}(\tilde{q}(\mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{z}, \beta)) = \\ &\mathbb{E}[\log(p(\mathbf{c}, \mathbf{b}, \mathbf{x}, \mathbf{z}, \beta|\mathbf{r}))] - \mathbb{E}[\log(\tilde{q}(\mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{z}, \beta))] \end{aligned} \tag{5.9}$$

is maximized with respect to the trial function, so that the following Kullback-Leibler divergence

$$\mathcal{KL}(\tilde{q}(\beta)\tilde{q}(\mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{z})||p(\mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{z}, \beta|\mathbf{r})), \tag{5.10}$$

is minimized, which leads to the approximation (by integrating out β)

$$\tilde{q}(\mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{z}) \approx p(\mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{z}|\mathbf{r}). \tag{5.11}$$

From the above, by integrating out $\mathbf{c}, \mathbf{x}$ and $\mathbf{z}$, it is expected that the marginal $\tilde{q}(\mathbf{b}) \approx p(\mathbf{b}|\mathbf{r})$, and similarly, by integrating out $\mathbf{b}, \mathbf{x}$ and $\mathbf{z}$, $\tilde{q}(\mathbf{c}) \approx p(\mathbf{c}|\mathbf{r})$. In terms of structured variation message passing [53], the computation of $\tilde{q}(\mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{z})$ corresponds to BP in the subgraph shown in Part (ii) of the factor graph in Figure 5.1, except the function node $f_{\mathbf{r}}$ because it connects an external variable node β [53]. It is noted that the BP message passing between $\mathbf{z}$, f_z and $\mathbf{x}$ (i.e., BP in the dash-dotted box in Figure 5.1) can be difficult and computational intensive. Fortunately, AMP, derived based on loopy BP (which in this case is actually UAMP as the unitary transformation has already been performed previously) is an excellent replacement to accomplish the BP message passing for the dash-dotted box efficiently. In addition, there are difficulties with the priors $p(\mathbf{b})$ and $p(\mathbf{c})$ (corresponding to the factors $f_{\mathbf{b}}$ and $f_{\mathbf{c}}$ in Figure 5.1) as they may not be friendly, resulting in intractable BP messages. This can be handled with EP, which has been widely used in the literature to solve similar problems. At the variable node $\mathbf{c}$ (or $\mathbf{b}$), an approximate marginal about $\mathbf{c}$ (or $\mathbf{b}$) through an iterative process with moment matching [40] is obtained, thereby an approximation to the a posteriori mean $\mathbb{E}(\mathbf{c}|\mathbf{r})$ (or $\mathbb{E}(\mathbf{b}|\mathbf{r})$), which can be served as our estimate.

It is noted that, all inference methods mentioned above including VI, EP, and UAMP involve an iterative process (but with a different hierarchy), and the multiple iterative processes can be simply combined as a single one. In terms of message passing, this is to carry out a forward message passing process and a backward message passing process in Figure 5.1 as an iteration. Thanks to the incorporation of UAMP to handle the BP in the dashed-dotted box in Figure 5.1, this leads to an efficient and robust approximate inference algorithm with details elaborated in next section.

5.2.3 Derivation of the Message Passing Algorithm

In this section, the forward and backward message passing in Figure 5.1 according to the principle of structured variational message passing [50], [52], [53] and EP are represented in detail. Throughout this thesis, the notation $\mathcal{M}_{n_a \to n_b}(h)$ is used to denote a message passed from node n_a to node n_b, which is a function of h.

5.2.3.1 Message Computations at Nodes x, $f_{\mathbf{z}}$, z and $f_{\mathbf{r}}$

Treat $\mathbf{x}$, $f_{\mathbf{z}}$ and $\mathbf{z}$ as a module, shown by the dash-dotted box in Figure 5.1. In the backward direction, with the incoming messages from the factor nodes $f_{\mathbf{x}}$ as the input, the module needs to output the message $\mathcal{M}_{\mathbf{z} \to f_{\mathbf{r}}}(\mathbf{z})$. In the forward direction, with the incoming messages from the factor node $f_{\mathbf{r}}$ as input, the module needs to output the message $\mathcal{M}_{\mathbf{x} \to f_{\mathbf{x}}}(\mathbf{x})$. This is the most computational intensive part of the approximate inference method, and it can be efficiently handled with UAMP as mentioned earlier. Considering the structure of $\mathbf{x}$ shown in (5.4), the length-NK vector $\mathbf{x}$ is divided into K length-N vectors $\{\mathbf{x}_k, k = 1, ..., K\}$, i.e.,

$$\mathbf{x} = \left[\mathbf{x}_1^T, ..., \mathbf{x}_K^T\right]^T. \tag{5.12}$$

Due to this, the UAMP algorithms in Section II cannot be applied directly, but the derivation still follows that of the UAMP algorithms exactly.

Note that the size of matrix $\mathbf{\Phi}$ is $M \times NK$. It is partitioned into K sub-matrices $\{\mathbf{\Phi}_k, k = 1, ..., K\}$, each with a size of $M \times N$, i.e.,

$$\mathbf{\Phi} = [\mathbf{\Phi}_1, ..., \mathbf{\Phi}_K]. \tag{5.13}$$

Then K vectors $\{\boldsymbol{\phi}_k, k = 1, ..., K\}$, each with a length of M, is defined, i.e.,

$$\boldsymbol{\phi}_k = |\mathbf{\Phi}_k|^2 \mathbf{1}_M. \tag{5.14}$$

With the above definitions, the following model is

$$\mathbf{r} = \sum_{k=1}^{K} \mathbf{\Phi}_k \mathbf{x}_k + \boldsymbol{\omega}. \tag{5.15}$$

The backward message passing is firstly investigated. Assume that the incoming message from factor node f_x is available, which is the mean and variance of $\mathbf{x}_k$. Following UAMP, it is assumed that the elements of $\mathbf{x}_k$ have a common variance $v_{\mathbf{x}_k}$, and the computation of $v_{\mathbf{x}_k}$ will be detailed later. The mean of $\mathbf{x}$ is denoted by $\hat{\mathbf{x}}$. Then two vectors $\boldsymbol{\nu}_{\mathbf{p}}$ and $\mathbf{p}$ are calculated as

$$\boldsymbol{\nu}_{\mathbf{p}} = \sum_{k=1}^{K} \boldsymbol{\phi}_k v_{\mathbf{x}_k} \tag{5.16}$$

$$\mathbf{p} = \sum_{k=1}^{K} \boldsymbol{\Phi}_k \hat{\mathbf{x}}_k - \boldsymbol{\nu}_{\mathbf{p}} \cdot \mathbf{s}, \tag{5.17}$$

where $\mathbf{s}$ is a vector, which is computed in the last iteration. UAMP also allows a loopy BP derivation that is the same as AMP, except that the derivation is based on the unitary transformed model. According to the BP derivation of (U)AMP,

$$\mathcal{M}_{\mathbf{z}\to f_{\mathbf{r}}}(\mathbf{z}) = \mathcal{M}_{f_{\mathbf{z}}\to\mathbf{z}}(\mathbf{z}) = \mathcal{N}\left(\mathbf{z}; \mathbf{p}, D(\boldsymbol{\nu}_p)\right). \tag{5.18}$$

It is noted that the factor node $f_{\mathbf{r}}$ connects the external variable node β. According to the rules of the structured variational message passing [53], the message $\mathcal{M}_{f_{\mathbf{r}}\to\beta}(\beta)$ can be computed as

$$\mathcal{M}_{f_{\mathbf{r}}\to\beta}(\beta) \propto \exp\left\{\int_{\mathbf{z}} \mathfrak{b}(\mathbf{z}) \log f_{\mathbf{r}}\right\} \tag{5.19}$$

where $\mathfrak{b}(\mathbf{z})$ is the the approximate marginal of $\mathbf{z}$, i.e.,

$$\begin{aligned} \mathfrak{b}(\mathbf{z}) &\propto \mathcal{M}_{f_{\mathbf{r}}\to\mathbf{z}}(\mathbf{z}) \mathcal{M}_{\mathbf{z}\to f_{\mathbf{r}}}(\mathbf{z}) \\ &= \mathcal{N}(\mathbf{z}; \hat{\mathbf{z}}, D(\boldsymbol{\nu}_{\mathbf{z}})) \end{aligned} \tag{5.20}$$

with

$$\boldsymbol{\nu}_{\mathbf{z}} = \mathbf{1}./\left(\mathbf{1}./\boldsymbol{\nu}_{\mathbf{p}} + \hat{\beta}\mathbf{1}_M\right) \tag{5.21}$$

$$\hat{\mathbf{z}} = \boldsymbol{\nu}_{\mathbf{z}} \cdot \left(\mathbf{p}./\boldsymbol{\nu}_{\mathbf{p}} + \hat{\beta}\mathbf{r}\right) \tag{5.22}$$

where $\hat{\beta}$ is the approximate a posteriori mean of the noise precision β in the last iteration. Note that there may be zero elements in $\boldsymbol{\nu}_{\mathbf{p}}$. $\hat{\beta}$ is initialized to 1 according to [8]. To avoid

the potential numerical problem, the above equations can be rewritten as

$$\boldsymbol{\nu}_{\mathbf{z}} = \boldsymbol{\nu}_p./(\mathbf{1} + \hat{\beta}\boldsymbol{\nu}_p) \tag{5.23}$$

$$\hat{\mathbf{z}} = (\hat{\beta}\boldsymbol{\nu}_p \cdot \boldsymbol{r} + \boldsymbol{p})./(\mathbf{1} + \hat{\beta}\boldsymbol{\nu}_p). \tag{5.24}$$

It is noted that in the above derivation, the message $\mathcal{M}_{f_{\mathbf{r}}\to\mathbf{z}}(\mathbf{z})$ is required, which turns out to be Gaussian, i.e., $\mathcal{M}_{f_{\mathbf{r}}\to z}(\mathbf{z}) = \mathcal{N}(\mathbf{z}, \mathbf{r}, \hat{\beta}^{-1})$, and its derivation is delayed to (5.26). Then, it is not hard to show that the message

$$\mathcal{M}_{f_{\mathbf{r}}\to\beta}(\beta) \propto \beta^M \exp\{-\beta(\|\mathbf{r} - \hat{\mathbf{z}}\|^2 + \mathbf{1}^T\boldsymbol{\nu}_{\mathbf{z}})\}. \tag{5.25}$$

This is the end of the backward message passing.

Next, the forward message passing is investigated. According to the rules of the structured variational message passing and noting that $f_{\mathbf{r}}$ connects the external variable node β,

$$\begin{aligned} \mathcal{M}_{f_{\mathbf{r}}\to\mathbf{z}}(\mathbf{z}) &\propto \exp\left\{\int_\beta \mathfrak{b}(\beta)\log f_{\mathbf{r}}\right\} \\ &\propto \mathcal{N}(\mathbf{z}; \mathbf{r}, \hat{\beta}^{-1}) \end{aligned} \tag{5.26}$$

with

$$\begin{aligned} \mathfrak{b}(\beta) &\propto \mathcal{M}_{f_{\mathbf{r}}\to\beta}(\beta) f_\beta \\ &\propto \beta^{M-1}\exp\{-\beta(\|\mathbf{r} - \hat{\mathbf{z}}\|^2 + \mathbf{1}^T\boldsymbol{\nu}_{\mathbf{z}})\}, \end{aligned} \tag{5.27}$$

and

$$\hat{\beta} = \int_\beta \beta\mathfrak{b}(\beta) = \frac{M}{\|\mathbf{r} - \hat{\mathbf{z}}\|^2 + \mathbf{1}^T\boldsymbol{\nu}_{\mathbf{z}}}, \tag{5.28}$$

where the use of the notation $\hat{\beta}$ is slightly abused as it is not distinguished from the last iteration. The result for $\hat{\beta}$ coincides with the result in [88] and [89].

The message $\mathcal{M}_{f_{\mathbf{r}}\to\mathbf{z}}(\mathbf{z})$ is input to the dash-dotted box in Figure 5.1. The Gaussian form

of the message suggests the following model

$$\mathbf{r} = \mathbf{z} + \mathbf{w}', \tag{5.29}$$

where the noise $\mathbf{w}'$ is Gaussian with mean zero and precision $\hat{\beta}$. This allows seamless connection with the forward recursion of UAMP. According to UAMP, the intermediate vectors $\boldsymbol{\nu}_{\mathbf{s}}$ and $\mathbf{s}$ are updated by

$$\boldsymbol{\nu}_{\mathbf{s}} = \mathbf{1}./(\boldsymbol{\nu}_{\mathbf{p}} + \hat{\beta}^{-1}\mathbf{1}) \tag{5.30}$$

$$\mathbf{s} = \boldsymbol{\nu}_{\mathbf{s}} \cdot (\mathbf{r} - \mathbf{p}). \tag{5.31}$$

Then calculate vectors $\nu_{\mathbf{q}_k}$ and $\hat{\mathbf{q}}_k$ for $k = 0, ..., K$ with

$$\nu_{\mathbf{q}_k} = 1/\left\langle |\boldsymbol{\Phi}_k^H|^2 \boldsymbol{\nu}_{\mathbf{s}} \right\rangle \tag{5.32}$$

$$\mathbf{q}_k = \hat{\mathbf{x}}_k + \nu_{\mathbf{q}_k} \boldsymbol{\Phi}_k^H \mathbf{s}. \tag{5.33}$$

The messages $\mathbf{q}_k$ and $\nu_{\mathbf{q}_k}$ are the mean and variance of $\mathbf{x}_k$. According to the BP derivation of (U)AMP,

$$\mathcal{M}_{\mathbf{x}\to f_{\mathbf{x}}}(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mathbf{q}, D(\boldsymbol{\nu}_{\mathbf{q}})) \tag{5.34}$$

with

$$\mathbf{q} = [\mathbf{q}_1^T, ..., \mathbf{q}_K^T]^T \tag{5.35}$$

$$\boldsymbol{\nu}_{\mathbf{q}} = [\nu_{\mathbf{q}_1}, ..., \nu_{\mathbf{q}_K}]^T \otimes \mathbf{1}_N, \tag{5.36}$$

which is the output of the dash-dotted box in Figure 5.1. This is the end of the forward message passing.

5.2.4 Message Computations at Nodes $f_{\mathbf{x}}$, b and c

It is noted that the function $f_{\mathbf{x}}(\mathbf{x}, \mathbf{c}, \mathbf{b})$ can be further factorized, i.e.,

$$f_{\mathbf{x}}(\mathbf{x}, \mathbf{c}, \mathbf{b}) = \prod\nolimits_{n,k} f_{x_{n,k}}(b_k, c_n), \tag{5.37}$$

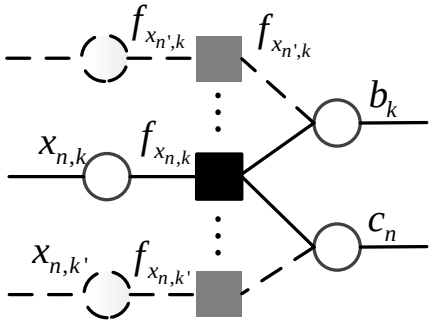

Figure 5.2: Factor graph representation for $f_{x_{n,k}}(c_n, b_k)$.

and the factor $f_{x_{n,k}}(c_n, b_k)$ is shown in Figure 5.2 with solid lines, which will be used to derive the forward and backward message computations.

The forward message passing is firstly investigated , where the message $\mathcal{M}_{\mathbf{x}\to f_{\mathbf{x}}}(\mathbf{x})$ is available from the dash-dotted box . The nth entry of $\mathbf{q}_k$ is denoted by $q_{n,k}$, then $\mathcal{M}_{x_{n,k}\to f_{x_{n,k}}}(x_{n,k}) = \mathcal{N}(x_{n,k}; q_{n,k}, \nu_{\mathbf{q}_k})$ and the factor $f_{x_{n,k}} = \delta(x_{n,k} - b_k c_n)$ are given.

To compute the message $\mathcal{M}_{f_{x_{n,k}}\to c_n}(c_n)$ with BP at factor node $f_{x_{n,k}}$, $x_{n,k}$ and b_k is need to integrate out. However, due to the multiplication of b_k and c_n, the message will be intractable even if the incoming message $\mathcal{M}_{b_k\to f_{x_{n,k}}}(b_k)$ is Gaussian. To solve this, BP is applied firstly and the variable $x_{n,k}$ is eliminated to get an intermediate function node $\tilde{f}_{x_{n,k}}(c_n, b_k)$, i.e.,

$$\begin{aligned}\tilde{f}_{x_{n,k}}(c_n, b_k) &= \int_{x_{n,k}} \mathcal{M}_{x_{n,k}\to f_{x_{n,k}}}(x_{n,k}) \cdot f_{x_{n,k}} \\ &= \mathcal{N}\left(c_n b_k; q_{n,k}, \nu_{\mathbf{q}_k}\right). \end{aligned} \tag{5.38}$$

This turns the function node $f_{x_{n,k}}$ with the hard constraint $\delta(x_{n,k} - b_k c_n)$ to a 'soft' function node, enabling the use of variational inference to handle c_n and b_k. With the intermediate local function $\tilde{f}_{x_{n,k}}(b_k, c_n)$, the outgoing message from $f_{x_{n,k}}$ is calculated to c_n as

$$\begin{aligned}\mathcal{M}_{f_{x_{n,k}}\to c_n}(c_n) &= \exp\left\{\int_{b_k} \mathfrak{b}(b_k) \log \tilde{f}_{x_{n,k}}\right\} \\ &= \mathcal{N}\left(c_n; \vec{c}_{n,k}, \vec{\nu}_{c_{n,k}}\right) \end{aligned} \tag{5.39}$$

where

$$\vec{c}_{n,k} = \frac{q_{n,k}\hat{b}_k^*}{|\hat{b}_k|^2 + \nu_{b_k}}, \tag{5.40}$$

$$\vec{v}_{c_{n,k}} = \frac{\nu_{\mathbf{q}_k}}{|\hat{b}_k|^2 + \nu_{b_k}}, \tag{5.41}$$

with $\hat{b}_k$ and ν_{b_k} being the approximate a posteriori mean and variance of b_k, which are computed in (5.58) and (5.59). It is noted that, in the case of $b_1 = 1$, $\hat{b}_1 = 1$ and $\nu_{b_1} = 0$ are set. With BP and referring to Figure 5.2, the message $\mathcal{M}_{c_n \to f_\mathbf{c}}(c_n)$ can be represented as

$$\mathcal{M}_{c_n \to f_\mathbf{c}}(c_n) = \mathcal{N}(c_n; \vec{c}_n, \vec{v}_{c_n}) \tag{5.42}$$

with

$$\vec{v}_{c_n} = 1/\sum_{k=1}^{K} \frac{1}{\vec{v}_{c_{n,k}}} \tag{5.43}$$

$$\vec{c}_n = \vec{v}_{c_n} \sum_{k=1}^{K} \frac{\vec{c}_{n,k}}{\vec{v}_{c_{n,k}}}. \tag{5.44}$$

So, the marginal of c_n ($n = 1, ..., N$) can be expressed as

$$\mathfrak{b}(c_n) = \int_{\mathbf{c} \vee c_n} \prod_n \mathcal{M}_{c_n \to f_\mathbf{c}}(c_n) f_\mathbf{c}. \tag{5.45}$$

As mentioned earlier, according to EP, the marginal is projected to be Gaussian through moment matching, i.e.,

$$\mathfrak{b}'(c_n) = \mathcal{N}(c_n; \hat{c}_n, \nu_{c_n}) \tag{5.46}$$

with

$$\hat{c}_n = \mathbb{E}\Big[c_n | \{\vec{v}_{c_n}, \vec{c}_n\}, f_\mathbf{c}\Big] \tag{5.47}$$

$$\nu_{c_n} = \mathbb{V}\mathrm{ar}\Big[c_n | \{\vec{v}_{c_n}, \vec{c}_n\}, f_\mathbf{c}\Big], \tag{5.48}$$

which are a posterior mean and variance of c_n based on the prior $f_{\mathbf{c}}$ and the following pseudo observation model [35], [90]

$$\vec{c}_n = c_n + w'_n, \tag{5.49}$$

with w'_n denoting a Gaussian noise with mean 0 and variance $\vec{v}_{c_n}$.

Similarly, the message from $f_{x_{n,k}}$ to b_k is calculated, i.e.,

$$\mathcal{M}_{f_{x_{n,k}} \to b_k}(b_k) = \mathcal{N}\left(b_k; \vec{b}_{n,k}, \vec{v}_{b_{n,k}}\right) \tag{5.50}$$

where

$$\vec{b}_{n,k} = \frac{q_{n,k}\hat{c}_n^*}{|\hat{c}_n|^2 + \nu_{c_n}}, \tag{5.51}$$

$$\vec{v}_{b_{n,k}} = \frac{\nu_{\mathbf{q}_k}}{|\hat{c}_n|^2 + \nu_{c_n}} \tag{5.52}$$

with $\hat{c}_n$ and ν_{c_n} being the approximate a posteriori mean and variance of c_n, which are updated in (5.47) and (5.48). Then with BP, the message $\mathcal{M}_{b_k \to f_{\mathbf{b}}}(b_k)$ can be expressed as

$$\mathcal{M}_{b_k \to f_{\mathbf{b}}}(b_k) = \mathcal{N}\left(b_k; \vec{b}_k, \vec{v}_{b_k}\right) \tag{5.53}$$

with

$$\vec{v}_{b_k} = 1 / \sum_{n=1}^{N} \frac{1}{\vec{v}_{b_{n,k}}} \tag{5.54}$$

$$\vec{b}_k = \vec{v}_{b_k} \sum_{n=1}^{N} \frac{\vec{b}_{n,k}}{\vec{v}_{b_{n,k}}}. \tag{5.55}$$

Then the marginal of each b_k is computed,

$$\mathfrak{b}(b_k) = \int_{\mathbf{b} \vee b_k} \prod_k \mathcal{M}_{b_k \to f_{\mathbf{b}}}(b_k) f_{\mathbf{b}}. \tag{5.56}$$

Similarly, it is then projected to be Gaussian, i.e.,

$$\mathfrak{b}'(b_k) = \mathcal{N}\left(b_k; \hat{b}_k, \nu_{b_k}\right) \tag{5.57}$$

with

$$\hat{b}_k = \mathbb{E}\Big[b_k|\{\vec{\nu}_{b_k}, \vec{b}_k\}, f_{\mathbf{b}}\Big] \tag{5.58}$$

$$\nu_{b_k} = \mathbb{V}\mathrm{ar}\Big[b_k|\{\vec{\nu}_{b_k}, \vec{b}_k\}, f_{\mathbf{b}}\Big], \tag{5.59}$$

which are the a posteriori mean and variance of b_k based on the prior $f_{\mathbf{b}}$ and the following pseudo observation model

$$\vec{b}_k = b_k + w_k'' \tag{5.60}$$

with w_k'' denoting a Gaussian noise with mean 0 and variance $\vec{\nu}_{b_k}$. It is noted that, in the case of $b_1 = 1$, $\hat{b}_1 = 1$ and $\nu_{b_1} = 0$ is set. This is the end of the forward message passing.

Next, the backward message passing is investigated. According to the rule of EP, the backward message

$$\mathcal{M}_{b_k \to f_{x_{n,k}}}(b_k) = \frac{\mathfrak{b}'(b_k)}{\mathcal{M}_{f_{x_{n,k}} \to b_k}(b_k)}. \tag{5.61}$$

They are represented collectively as $\mathcal{M}_{\mathbf{b} \to f_{\mathbf{x}}}(\mathbf{b})$, which is Gaussian with mean $\overleftarrow{\mathbf{b}}$ and variance $D(\overleftarrow{\boldsymbol{\nu}}_{\mathbf{b}})$. With the factor graph shown in Figure 5.2, the mean and variance can be calculated as

$$\begin{aligned} \overleftarrow{\boldsymbol{\nu}}_{\mathbf{b}} &= \big((\mathbf{1}./\boldsymbol{\nu}_{\mathbf{b}}) \otimes \mathbf{1}_N - \mathbf{1}./\vec{\boldsymbol{\nu}}_{\mathbf{b}}\big)^{.-1} \\ &= \big((\boldsymbol{\nu}_{\mathbf{b}} \otimes \mathbf{1}_N) \cdot \vec{\boldsymbol{\nu}}_{\mathbf{b}}\big)./\big(\vec{\boldsymbol{\nu}}_{\mathbf{b}} - (\boldsymbol{\nu}_{\mathbf{b}} \otimes \mathbf{1}_N)\big) \end{aligned} \tag{5.62}$$

$$\begin{aligned} \overleftarrow{\mathbf{b}} &= \overleftarrow{\boldsymbol{\nu}}_{\mathbf{b}} \cdot \Big(\big(\hat{\mathbf{b}}./\boldsymbol{\nu}_{\mathbf{b}}\big) \otimes \mathbf{1}_N - \vec{\mathbf{b}}./\vec{\boldsymbol{\nu}}_{\mathbf{b}}\Big), \\ &= \big((\hat{\mathbf{b}} \otimes \mathbf{1}_N) \cdot \vec{\boldsymbol{\nu}}_{\mathbf{b}} - \vec{\mathbf{b}} \cdot (\boldsymbol{\nu}_{\mathbf{b}} \otimes \mathbf{1}_N)\big)./\big(\vec{\boldsymbol{\nu}}_{\mathbf{b}} - (\boldsymbol{\nu}_{\mathbf{b}} \otimes \mathbf{1}_N)\big) \end{aligned} \tag{5.63}$$

where $\boldsymbol{\nu}_{\mathbf{b}} = [\nu_{b_1}, ..., \nu_{b_K}]^T$, $\hat{\mathbf{b}} = [\hat{b}_1, ..., \hat{b}_K]^T$, $[\vec{\boldsymbol{\nu}}_{\mathbf{b}}]_{(k-1)N+n} = \vec{\nu}_{b_{n,k}}$ and $[\ \vec{\mathbf{b}}\ ]_{(k-1)N+n} = \vec{b}_{n,k}$.

Similarly, the message $\mathcal{M}_{\mathbf{c} \to f_{\mathbf{x}}}(\mathbf{c})$ is also Gaussian with mean $\overleftarrow{\mathbf{c}}$, and variance $D(\overleftarrow{\boldsymbol{\nu}}_c)$,

Algorithm 7 Bi-UAMP for SMV

Unitary transform: $\mathbf{r} = \mathbf{U}^H\mathbf{y} = \mathbf{\Phi x} + \boldsymbol{\omega}$, where $\mathbf{A}_{M\times NK} = \mathbf{U\Lambda V}$, $\mathbf{\Phi} = \mathbf{U}^H\mathbf{A} = \mathbf{\Lambda V}$, and $\mathbf{x} = \mathbf{b}\otimes\mathbf{c}$ with $\mathbf{b} = [b_1, ..., b_K]^T$ and $\mathbf{c} = [c_1, ..., c_N]^T$.

Let $\mathbf{\Phi} = [\mathbf{\Phi}_1, ..., \mathbf{\Phi}_K]$, $\boldsymbol{\phi}_k = |\mathbf{\Phi}_k|^2\mathbf{1}_N$, and $\mathbf{x} = \left[\mathbf{x}_1^T, ..., \mathbf{x}_K^T\right]^T$, $k = 1, ...K$ and $n = 1, ..., N$.

Initialize $\hat{b}_k$, $\nu_{b_k} = 1$, $\nu_{\mathbf{x}_k} = 1$, $\hat{\mathbf{x}}_k = \mathbf{0}$, $\mathbf{s} = \mathbf{0}$ and $\hat{\beta} = 1$.

Repeat

$\boldsymbol{\nu}_\mathbf{p} = \sum_k \boldsymbol{\phi}_k\nu_{\mathbf{x}_k}$

$\mathbf{p} = \sum_k \mathbf{\Phi}_k\hat{\mathbf{x}}_k - \boldsymbol{\nu}_\mathbf{p}\cdot\mathbf{s}$

$\boldsymbol{\nu}_\mathbf{z} = \boldsymbol{\nu}_p./(\mathbf{1} + \hat{\beta}\boldsymbol{\nu}_p)$

$\hat{\mathbf{z}} = (\hat{\beta}\boldsymbol{\nu}_p\cdot\boldsymbol{r} + \boldsymbol{p})./(\mathbf{1} + \hat{\beta}\boldsymbol{\nu}_p)$

$\hat{\beta} = M/(\|\mathbf{r} - \hat{\mathbf{z}}\|^2 + \mathbf{1}^T\boldsymbol{\nu}_\mathbf{z})$

$\boldsymbol{\nu}_\mathbf{s} = \mathbf{1}./(\boldsymbol{\nu}_\mathbf{p} + \hat{\beta}^{-1}\mathbf{1}_M)$

$\mathbf{s} = \boldsymbol{\nu}_\mathbf{s}\cdot(\mathbf{r} - \mathbf{p})$

$\forall k : \nu_{\mathbf{q}_k} = 1/\left\langle|\mathbf{\Phi}_k^H|^2\boldsymbol{\nu}_\mathbf{s}\right\rangle$

$\forall k : \mathbf{q}_k = \hat{\mathbf{x}}_k + \nu_{\mathbf{q}_k}\mathbf{\Phi}_k^H\mathbf{s}$

(In the case of $b_1 = 1$, set $\hat{b}_1 = 1$ and $\nu_{b_1} = 0$.)

$\forall k : \vec{\mathbf{c}}_k = \mathbf{q}_k\hat{b}_k^*/(|\hat{b}_k|^2 + \nu_{b_k})$

$\forall k : \vec{\boldsymbol{\nu}}_{\mathbf{c}_k} = \mathbf{1}_N\nu_{\mathbf{q}_k}/(|\hat{b}_k|^2 + \nu_{b_k})$

$\vec{\boldsymbol{\nu}}_c = \mathbf{1}_N./(\sum_k \mathbf{1}_N./\vec{\boldsymbol{\nu}}_{c_k})$

$\vec{\mathbf{c}} = \vec{\boldsymbol{\nu}}_\mathbf{c}\cdot\sum_k(\vec{\mathbf{c}}_k./\vec{\boldsymbol{\nu}}_{\mathbf{c}_k})$

$\forall n : \hat{c}_n = \mathbb{E}[c_n|\vec{\boldsymbol{\nu}}_\mathbf{c}, \vec{\mathbf{c}}, f_\mathbf{c}]$

$\forall n : \nu_{c_n} = \mathbb{V}\mathrm{ar}[c_n|\vec{\boldsymbol{\nu}}_\mathbf{c}, \vec{\mathbf{c}}, f_\mathbf{c}]$

$\boldsymbol{\nu}_\mathbf{c} =< [\nu_{c_1}, ..., \nu_{c_N}] > \mathbf{1}_N$, and $\hat{\mathbf{c}} = [\hat{c}_1, ..., \hat{c}_N]^T$

$\forall k : \vec{\boldsymbol{\nu}}_{\mathbf{b}_k} = \nu_{\mathbf{q}_k}\mathbf{1}_N./(|\hat{\mathbf{c}}|^2 + \boldsymbol{\nu}_\mathbf{c})$

$\forall k : \vec{\mathbf{b}}_k = \mathbf{q}_k\cdot\hat{\mathbf{c}}^*./(|\hat{\mathbf{c}}|^2 + \boldsymbol{\nu}_\mathbf{c})$

$\forall k : \vec{\nu}_{b_k} = (\mathbf{1}_N^T(\mathbf{1}_N./\vec{\boldsymbol{\nu}}_{\mathbf{b}_k}))^{-1}$

$\forall k : \vec{b}_k = \vec{\nu}_{b_k}\mathbf{1}_N^T(\vec{\mathbf{b}}_k./\vec{\boldsymbol{\nu}}_{b_k})$

$\forall k : \hat{b}_k = \mathbb{E}[b_k|\{\vec{\nu}_{b_k}, \vec{b}_k\}, f_\mathbf{b}]$

$\forall k : \nu_{b_k} = \mathbb{V}\mathrm{ar}[b_k|\{\vec{\nu}_{b_k}, \vec{b}_k\}, f_\mathbf{b}]$

(In the case of $b_1 = 1$, set $\hat{b}_1 = 1$ and $\nu_{b_1} = 0$.)

$\forall k : \overleftarrow{\boldsymbol{\nu}}_{\mathbf{b}_k} = (\nu_{b_k}\vec{\boldsymbol{\nu}}_{\mathbf{b}_k})./(\vec{\boldsymbol{\nu}}_{\mathbf{b}_k} - \nu_{b_k}\mathbf{1}_N)$

$\forall k : \overleftarrow{\mathbf{b}}_k = (\hat{b}_k\vec{\boldsymbol{\nu}}_{\mathbf{b}_k} - \nu_{b_k}\vec{\mathbf{b}}_k)./(\vec{\boldsymbol{\nu}}_{\mathbf{b}_k} - \nu_{b_n}\mathbf{1}_N)$

$\forall k : \overleftarrow{\nu}_{c_k} = (\mathbf{1}./\nu_c - \mathbf{1}./\vec{\boldsymbol{\nu}}_{\mathbf{c}_k})^{.-1}$

$\forall k : \overleftarrow{\mathbf{c}}_k = \overleftarrow{\boldsymbol{\nu}}_{c_k}\cdot(\hat{\mathbf{c}}./\boldsymbol{\nu}_c - \vec{\mathbf{c}}_k./\vec{\boldsymbol{\nu}}_{\mathbf{c}_k})$

$\forall k : \overleftarrow{\mathbf{x}}_k = \overleftarrow{\mathbf{b}}_k\cdot\overleftarrow{\mathbf{c}}_k$

$\forall k : \overleftarrow{\boldsymbol{\nu}}_{\mathbf{x}_k} = |\overleftarrow{\mathbf{b}}_k|^2\cdot\overleftarrow{\boldsymbol{\nu}}_{\mathbf{c}_k} + \overleftarrow{\boldsymbol{\nu}}_{\mathbf{b}_k}\cdot\left|\overleftarrow{\mathbf{c}}_k\right|^2 + \overleftarrow{\boldsymbol{\nu}}_{\mathbf{b}_k}\cdot\overleftarrow{\boldsymbol{\nu}}_{\mathbf{c}_k}$

$\forall k : \boldsymbol{\nu}_{\mathbf{x}_k} = \left(1/\nu_{\mathbf{q}_k}\mathbf{1}_N + \mathbf{1}./\overleftarrow{\boldsymbol{\nu}}_{\mathbf{x}_k}\right)^{.-1}$

$\forall k : \hat{\mathbf{x}}_k = \boldsymbol{\nu}_{\mathbf{x}_k}\cdot\left(1/\nu_{\mathbf{q}_k}\mathbf{q}_k + \overleftarrow{\mathbf{x}}_k./\overleftarrow{\boldsymbol{\nu}}_{\mathbf{x}_k}\right)$

$\forall k : \nu_{\mathbf{x}_k} =< \boldsymbol{\nu}_{\mathbf{x}_k} >$

Until terminated

which can be calculated as

$$\overleftarrow{\boldsymbol{\nu}}_c = \left(\mathbf{1}_K \otimes (\mathbf{1}./\boldsymbol{\nu}_c) - \mathbf{1}./\vec{\boldsymbol{\nu}}_{\mathbf{c}}\right)^{.-1} \tag{5.64}$$

$$\overleftarrow{\mathbf{c}} = \overleftarrow{\boldsymbol{\nu}}_c \cdot \left(\mathbf{1}_K \otimes (\hat{\mathbf{c}}./\boldsymbol{\nu}_c) - \vec{\mathbf{c}}./\vec{\boldsymbol{\nu}}_{\mathbf{c}}\right), \tag{5.65}$$

where $\boldsymbol{\nu}_{\mathbf{c}} = [\nu_{c_1}, ..., \nu_{c_N}]^T$, $\hat{\mathbf{c}} = [\hat{c}_1, ..., \hat{c}_N]^T$, $[\vec{\boldsymbol{\nu}}_{\mathbf{c}}]_{(n-1)K+k} = \vec{\nu}_{c_{n,k}}$ and $[\vec{\mathbf{c}}]_{(n-1)K+k} = \vec{c}_{n,k}$. Then, the backward message $\mathcal{M}_{f_{\mathbf{x}}\to\mathbf{x}}(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \overleftarrow{\mathbf{x}}, \overleftarrow{\boldsymbol{\nu}}_{\mathbf{x}})$ with

$$\overleftarrow{\mathbf{x}} = \overleftarrow{\mathbf{b}} \cdot \overleftarrow{\mathbf{c}} \tag{5.66}$$

$$\overleftarrow{\boldsymbol{\nu}}_{\mathbf{x}} = |\overleftarrow{\mathbf{b}}|^2 \cdot \overleftarrow{\boldsymbol{\nu}}_{\mathbf{c}} + \overleftarrow{\boldsymbol{\nu}}_{\mathbf{b}} \cdot \left|\overleftarrow{\mathbf{c}}\right|^2 + \overleftarrow{\boldsymbol{\nu}}_{\mathbf{b}} \cdot \overleftarrow{\boldsymbol{\nu}}_{\mathbf{c}}, \tag{5.67}$$

where $\overleftarrow{\mathbf{x}} = [\overleftarrow{\mathbf{x}}_1^T, ..., \overleftarrow{\mathbf{x}}_K^T]^T$ and $\overleftarrow{\boldsymbol{\nu}}_{\mathbf{x}} = [\overleftarrow{\boldsymbol{\nu}}_{\mathbf{x}_1}^T, ..., \overleftarrow{\boldsymbol{\nu}}_{\mathbf{x}_K}^T]^T$. The backward message is combined with the message $\mathcal{M}_{\mathbf{x}\to f_{\mathbf{x}}}(\mathbf{x})$ (the output of the dash-dotted box in last iteration) i.e.,

$$\boldsymbol{\nu}_{\mathbf{x}_k} = \left(1/\nu_{\mathbf{q}_k}\mathbf{1}_N + \mathbf{1}./\overleftarrow{\boldsymbol{\nu}}_{\mathbf{x}_k}\right)^{.-1} \tag{5.68}$$

$$\hat{\mathbf{x}}_k = \boldsymbol{\nu}_{x_k} \cdot \left(1/\nu_{\mathbf{q}_k}\mathbf{q}_k + \overleftarrow{\mathbf{x}}_k./\overleftarrow{\boldsymbol{\nu}}_{\mathbf{x}_k}\right) \tag{5.69}$$

$$\nu_{\mathbf{x}_k} = < \boldsymbol{\nu}_{\mathbf{x}_k} > \tag{5.70}$$

which are then passed to the dash-dotted box as input. This is the end of the backward message passing.

The approximate inference algorithm is called Bi-UAMP for SMV, and it can be organized in a more succinct form, which is summarized in **Algorithm** 7.

5.3 Extension to MMV

In this section, the case of MMV with the model is extended by Bi-UAMP

$$\mathbf{Y} = \sum_{k=1}^{K} b_k \mathbf{A}_k \mathbf{C} + \mathbf{W} \tag{5.71}$$

where $\mathbf{Y}$ is an observation matrix with size $M \times L$, $\mathbf{W}$ denotes a white Gaussian noise matrix with mean 0 and precision β, matrices $\{\mathbf{A}_k\}$ are known, and $\mathbf{C}$ with size $N \times L$ and

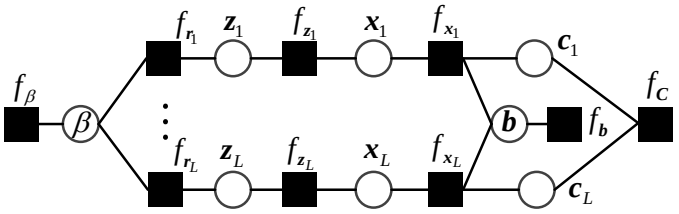

Figure 5.3: Factor graph representation of (5.75).

$\mathbf{b} = [b_1, ..., b_K]^T$ are to be estimated.

Similar to the case of SMV, (5.71) can be reformulated as

$$\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{W} \tag{5.72}$$

where $\mathbf{A} = [\mathbf{A}_1, ..., \mathbf{A}_K]$, and $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_L]$ with

$$\mathbf{x}_l = \mathbf{b} \otimes \mathbf{c}_l. \tag{5.73}$$

With the SVD $\mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}$ and unitary transformation, the following model is given

$$\mathbf{R} = \boldsymbol{\Phi}\mathbf{X} + \overline{\mathbf{W}} \tag{5.74}$$

where $\mathbf{R} = \mathbf{U}^H\mathbf{Y}$, $\boldsymbol{\Phi} = \boldsymbol{\Lambda}\mathbf{V} = \mathbf{U}^H\mathbf{A}$ and $\overline{\mathbf{W}} = \mathbf{U}^H\mathbf{W}$. Define $\mathbf{z}_l = \boldsymbol{\Phi}\mathbf{x}_l$ and $\mathbf{Z} = [\mathbf{z}_1, ..., \mathbf{z}_L]$, then the joint distribution of the variables is factorized in (5.74) as

$$\begin{aligned} & p(\mathbf{X}, \mathbf{C}, \mathbf{b}, \mathbf{Z}, \beta | \mathbf{R}) \\ & \propto p(\mathbf{C}) p(\mathbf{b}) p(\beta) \prod\nolimits_l p(\mathbf{r}_l | \mathbf{z}_l, \beta) p(\mathbf{z}_l | \mathbf{x}_l) p(\mathbf{x}_l | \mathbf{b}, \mathbf{c}_l) \\ & \triangleq f_{\mathbf{C}}(\mathbf{C}) f_{\mathbf{b}}(\mathbf{b}) f_\beta(\beta) \prod\nolimits_l f_{\mathbf{r}_l}(\mathbf{z}_l, \beta) f_{\mathbf{z}_l}(\mathbf{z}_l, \mathbf{x}_l) f_{\mathbf{x}_l}(\mathbf{x}_l, \mathbf{b}, \mathbf{c}_l). \end{aligned} \tag{5.75}$$

The factor graph representation for the factorization in (5.75) is depicted in Figure 5.3. The message updates related to $\mathbf{z}_l$, $\mathbf{x}_l$ and $\mathbf{c}_l$ are the same as those in **Algorithm** 7, and they can be computed in parallel. The major difference lies in the computations of $b(\mathbf{b})$

Algorithm 8 Bi-UAMP for MMV

Unitary transform: $\mathbf{R} = \mathbf{U}^H\mathbf{Y} = \mathbf{\Phi X} + \overline{\mathbf{W}}$, where $\mathbf{A}_{M\times NK} = \mathbf{U\Lambda V}$, $\mathbf{\Phi} = \mathbf{U}^H\mathbf{A} = \mathbf{\Lambda V}$, and $\mathbf{x}_l = \mathbf{b} \otimes \mathbf{c}_l$ with $\mathbf{b} = [b_1, ..., b_K]^T$ and $\mathbf{c}_l = [c_{1,l}, ..., c_{N,l}]^T$.

Let $\mathbf{\Phi} = [\mathbf{\Phi}_1, ..., \mathbf{\Phi}_K]$, $\boldsymbol{\phi}_k = |\mathbf{\Phi}_k|^2\mathbf{1}_N$, and $\mathbf{x}_l = \left[\mathbf{x}_{1,l}^T, ..., \mathbf{x}_{K,l}^T\right]^T$, $k = 1, ..., K, n = 1, ..., N$ and $l = 1, ..., L$.

Initialize: $\hat{b}_k$, $\nu_{b_k} = 1$, $\nu_{\mathbf{x}_{k,l}} = 1$, $\hat{\mathbf{x}}_{k,l} = \mathbf{0}$, $\mathbf{s}_l = \mathbf{0}$, and $\hat{\beta} = 1$.

Repeat

1: $\forall l$: $\boldsymbol{\nu}_{\mathbf{p}_l} = \sum_k \boldsymbol{\phi}_k \nu_{\mathbf{x}_{k,l}}$
2: $\forall l$: $\mathbf{p}_l = \sum_k \mathbf{\Phi}_k\hat{\mathbf{x}}_{k,l} - \boldsymbol{\nu}_{\mathbf{p}_l} \cdot \mathbf{s}_l$
3: $\forall l$: $\boldsymbol{\nu}_{\mathbf{z}_l} = \boldsymbol{\nu}_{\mathbf{p}_l}./(\mathbf{1} + \hat{\beta}\boldsymbol{\nu}_{\mathbf{p}_l})$
4: $\forall l$: $\hat{\mathbf{z}}_l = (\hat{\beta}\boldsymbol{\nu}_{\mathbf{p}_l} \cdot \boldsymbol{r}_l + \boldsymbol{p}_l)./(\mathbf{1} + \hat{\beta}\boldsymbol{\nu}_{\mathbf{p}_l})$
5: $\hat{\beta} = ML/\sum_l \left(\left\|\mathbf{r}_l - \hat{\mathbf{z}}_l\right\|^2 + \mathbf{1}^T\boldsymbol{\nu}_{\mathbf{z}_l}\right)$
6: $\forall l$: $\boldsymbol{\nu}_{\mathbf{s}_l} = \mathbf{1}./(\boldsymbol{\nu}_{\mathbf{p}_l} + \hat{\beta}^{-1}\mathbf{1}_M)$
7: $\forall l$: $\mathbf{s}_l = \boldsymbol{\nu}_{\mathbf{s}_l} \cdot (\mathbf{r}_l - \mathbf{p}_l)$
8: $\forall l, k$: $\nu_{\mathbf{q}_{k,l}} = 1/\langle|\mathbf{\Phi}_k^H|^2\boldsymbol{\nu}_{\mathbf{s}_l}\rangle$
9: $\forall l, k$: $\mathbf{q}_{k,l} = \hat{\mathbf{x}}_{k,l} + \nu_{\mathbf{q}_{k,l}}\mathbf{\Phi}_k^H\mathbf{s}_l$
10: $\forall l, k$: $\vec{\mathbf{c}}_{k,l} = \mathbf{q}_{k,l}\hat{b}_k^*/(|\hat{b}_k|^2 + \nu_{b_k})$
11: $\forall l, k$: $\vec{\boldsymbol{\nu}}_{\mathbf{c}_{k,l}} = \mathbf{1}_N\nu_{\mathbf{q}_{k,l}}/(|\hat{b}_k|^2 + \nu_{b_k})$
12: $\forall l$: $\vec{\boldsymbol{\nu}}_{\mathbf{c}_l} = \mathbf{1}_N./\sum_k(\mathbf{1}_N./\vec{\boldsymbol{\nu}}_{\mathbf{c}_{k,l}})$
13: $\forall l$: $\vec{\mathbf{c}}_l = \vec{\boldsymbol{\nu}}_{\mathbf{c}_l} \cdot \sum_k(\vec{\mathbf{c}}_{k,l}./\vec{\boldsymbol{\nu}}_{\mathbf{c}_{k,l}})$
14: $\forall n, l$: $\hat{c}_{n,l} = \mathbb{E}[c_{n,l}|\{\vec{\boldsymbol{\nu}}_{\mathbf{c}_l}, \vec{\mathbf{c}}_l\}, f_{\mathbf{C}}]$
15: $\forall n, l$: $\nu_{c_{n,l}} = \mathbb{V}\mathrm{ar}[c_{n,l}|\{\vec{\boldsymbol{\nu}}_{\mathbf{c}_l}, \vec{\mathbf{c}}_l\}, f_{\mathbf{C}}]$
16: $\forall l$: $\boldsymbol{\nu}_{\mathbf{c}_l} =< [\nu_{c_{1,l}}, ..., \nu_{c_{N,l}}] > \mathbf{1}_N, \hat{\mathbf{c}}_l = [\hat{c}_{1,l}, ..., \hat{c}_{N,l}]^T$.
17: $\forall l, k$: $\vec{\boldsymbol{\nu}}_{\mathbf{b}_{k,l}} = \nu_{\mathbf{q}_{k,l}}\mathbf{1}_N./(|\hat{\mathbf{c}}_l|^2 + \boldsymbol{\nu}_{\mathbf{c}_l})$
18: $\forall l, k$: $\vec{\mathbf{b}}_{k,l} = \mathbf{q}_{k,l} \cdot \hat{\mathbf{c}}_l^*./(|\hat{\mathbf{c}}_l|^2 + \boldsymbol{\nu}_{c_l})$
19: $\forall k$: $\vec{\nu}_{b_k} = 1/\sum_l(\mathbf{1}_N^T(\mathbf{1}./\vec{\boldsymbol{\nu}}_{\mathbf{b}_{k,l}}))$
20: $\forall k$: $\vec{b}_k = \vec{\nu}_{b_k}\sum_l(\mathbf{1}_N^T(\hat{\mathbf{b}}_{k,l}./\vec{\boldsymbol{\nu}}_{\mathbf{b}_{k,l}}))$
21: $\forall k$: $\hat{b}_k = \mathbb{E}[b_k|\{\vec{\nu}_{b_k}, \vec{b}_k\}, f_{\mathbf{b}}]$
22: $\forall k$: $\nu_{b_k} = \mathbb{V}\mathrm{ar}[b_k|\{\vec{\nu}_{b_k}, \vec{b}_k\}, f_{\mathbf{b}}]$
23: $\forall l, k$: $\overleftarrow{\boldsymbol{\nu}}_{\mathbf{b}_{k,l}} = (\nu_{b_k}\vec{\boldsymbol{\nu}}_{\mathbf{b}_{k,l}})./(\vec{\boldsymbol{\nu}}_{\mathbf{b}_{k,l}} - \nu_{b_k}\mathbf{1}_N)$
24: $\forall l, k$: $\overleftarrow{\mathbf{b}}_{k,l} = (\hat{b}_k\vec{\boldsymbol{\nu}}_{\mathbf{b}_{k,l}} - \nu_{b_k}\vec{\mathbf{b}}_k)./(\vec{\boldsymbol{\nu}}_{\mathbf{b}_{k,l}} - \nu_{b_k}\mathbf{1}_N)$
25: $\forall l, k$: $\overleftarrow{\boldsymbol{\nu}}_{\mathbf{c}_{k,l}} = \left(\mathbf{1}_N./\boldsymbol{\nu}_{\mathbf{c}_l} - \mathbf{1}_N./\vec{\boldsymbol{\nu}}_{\mathbf{c}_{k,l}}\right)^{.-1}$
26: $\forall l, k$: $\overleftarrow{\mathbf{c}}_{k,l} = \overleftarrow{\boldsymbol{\nu}}_{\mathbf{c}_{k,l}} \cdot \left(\hat{\mathbf{c}}_l./\boldsymbol{\nu}_{\mathbf{c}_l} - \vec{\mathbf{c}}_{k,l}./\vec{\boldsymbol{\nu}}_{\mathbf{c}_{k,l}}\right)$
27: $\forall l, k$: $\overleftarrow{\mathbf{x}}_{k,l} = \overleftarrow{\mathbf{b}}_{k,l} \cdot \overleftarrow{\mathbf{c}}_{k,l}$
28: $\forall k, l$: $\overleftarrow{\boldsymbol{\nu}}_{\mathbf{x}_{k,l}} = |\overleftarrow{\mathbf{b}}_{k,l}|^2 \cdot \overleftarrow{\boldsymbol{\nu}}_{\mathbf{c}_{k,l}} + \overleftarrow{\boldsymbol{\nu}}_{\mathbf{b}_{k,l}} \cdot \left|\overleftarrow{\mathbf{c}}_{k,l}\right|^2 + \overleftarrow{\boldsymbol{\nu}}_{\mathbf{b}_{k,l}} \cdot \overleftarrow{\boldsymbol{\nu}}_{\mathbf{c}_{k,l}}$
29: $\forall l, k$: $\boldsymbol{\nu}_{\mathbf{x}_{k,l}} = (1/\nu_{\mathbf{q}_{k,l}}\mathbf{1}_N + \mathbf{1}_N./\boldsymbol{\nu}_{\overleftarrow{\mathbf{x}}_{k,l}})^{.-1}$
30: $\forall k, l$: $\hat{\mathbf{x}}_{k,l} = \boldsymbol{\nu}_{\mathbf{x}_{k,l}} \cdot \left(1/\nu_{\mathbf{q}_{k,l}}\mathbf{q}_{k,l} + \overleftarrow{\mathbf{x}}_{k,l}./\overleftarrow{\boldsymbol{\nu}}_{\mathbf{x}_{k,l}}\right)$
31: $\forall l, k$: $\nu_{\mathbf{x}_{k,l}} =< \boldsymbol{\nu}_{\mathbf{x}_{k,l}} >$

Until terminated

and $b(\beta)$, where the messages from $f_{\mathbf{x}_l}$ and $f_{\mathbf{r}_l}$, $\forall l$, should be considered, i.e.,

$$b(\mathbf{b}) \propto \prod\nolimits_l \mathcal{M}_{f_{\mathbf{x}_l}\to\mathbf{b}}(\mathbf{b})\mathcal{M}_{f_{\mathbf{b}}\to\mathbf{b}}(\mathbf{b}) \tag{5.76}$$

$$b(\beta) \propto \prod\nolimits_l \mathcal{M}_{f_{\mathbf{r}_l}\to\beta}(\beta)\mathcal{M}_{f_{\beta}\to\lambda}(\beta). \tag{5.77}$$

Similar to the SMV case, the message passing algorithm can be derived, which are summarized as **Algorithm** 8 (Bi-UAMP for MMV).

5.3.1 Discussions and Complexity Analysis

The following remarks and discussions are about Bi-UAMP:

1: In some problems, b_1 is known, e.g., $b_1 = 1$. In this case, $\hat{b}_1 = 1$ and $v_{b_1} = 0$ in Bi-UAMP are set, which are indicated in **Algorithm** 7.

2: It is not hard to show that, when $\mathbf{b} = b_1 = 1$, Bi-UAMP is reduced to UAMP (**Algorithm** 3) exactly.

3: It is interesting that the robustness of Bi-UAMP can be enhanced by simply damping $\mathbf{s}$, i.e., Line 7 of the SMV Bi-UAMP is changed as

$$\mathbf{s} = (1-\alpha)\mathbf{s} + \alpha\boldsymbol{\nu}_{\mathbf{s}} \cdot (\mathbf{r} - \mathbf{p}) \tag{5.78}$$

with $\alpha \in (0, 1]$, where α is the damping factor and $\alpha = 1$ leads to the case without damping. Accordingly, Line 7 of the MMV Bi-UAMP is changed as $\mathbf{s}_l = (1-\alpha)\mathbf{s}_l + \alpha\boldsymbol{\nu}_{\mathbf{s}_l} \cdot (\mathbf{r} - \mathbf{p})$.

4: The iterative process can be terminated based on some criterion, e.g., the normalized difference between the estimates of $\mathbf{b}$ of two consecutive iterations is smaller than a threshold, i.e., $\|\hat{\mathbf{b}}^t - \hat{\mathbf{b}}^{t-1}\|^2 / \|\hat{\mathbf{b}}(t)\|^2 < \epsilon$ where $\hat{\mathbf{b}}^t$ is the estimate of $\mathbf{b}$ at the tth iteration and ϵ is a threshold.

5: As the bilinear problem has local minima, the same strategy of restart as in [14] is used to mitigate the issue of being stuck at local minima. For each restart, $\{\hat{b}_k\}$ with different values is initialized.

6: In Bi-UAMP, Bayesian is applied to both **b** and **c** (or **C** in MMV). In contrast, **b** is treated as a unknown deterministic variable in BAd-VAMP, and only a point estimate is involved. As discussed in Section III.A, the Bayesian treatment to **b** can make the algorithm more flexible.

7: The computational complexity of Bi-UAMP is analyzed in the following. Bi-UAMP needs pre-processing, i.e., performing economic SVD for **A** and unitary transformation, and the complexity is $O(M^2NK)$. It is noted that the pre-processing can be carried out offline (although this in counting the runtime of Bi-UAMP in the simulations in Section 5.4 is not assumed). It can be seen from the Bi-UAMP algorithms that, there is no matrix inversion involved, and the most computational intensive parts only involve matrix-vector products. So the complexity of Bi-UAMP per iteration is $O(MNKL)$ (in the case of SMV, $L = 1$), which linearly increases with M, N, K and L. For comparison, BAd-VAMP involves one outer loop and two inner loops. The whole matrix $\mathbf{C}_l^t$ with size $N \times N$ in the second inner loop is required in multiple lines in the algorithm and $\mathbf{A}(\theta_{\mathbf{A}}^t)$ is updated in each inner iteration [14]. The computation of the matrix $\mathbf{C}_l^t$ leads to a complexity of $O(LN^3 + KMN)$ per inner iteration. Line 18 is also computational intensive, which requires a complexity of $O(K^2N^2)$ per inner iteration. Also, Line 20 of BAd-VAMP requires a complexity of $O(K^3)$ per inner iteration. It is difficult to have a very precise complexity comparison analytically as the algorithms require different numbers of iterations to converge. So, in Section 5.4, the runtime of several state-of-the-art algorithms as in [14] is compared. As demonstrated in Section 5.4, with much shorter runtime, Bi-UAMP can outperform the state-of-the-art algorithms significantly.

5.3.2 SE-Based Performance Prediction

From the derivation of Bi-UAMP, It can be seen that Bi-UAMP integrates VMP, BP, EP and UAMP. The incorporation of UAMP enables the approximate inference method to deal with the most computational intensive part with low complexity and high robustness. The rigorous performance analysis is difficult, but an attempt to predict its performance based on UAMP SE heuristically is made. The output variance of the UAMP module in

the dash-dotted box with respect to the input variance is tracked. However, the variances are about $\mathbf{x}$ instead of $\mathbf{b}$ and $\mathbf{c}$ (or $\mathbf{C}$ in the MMV case). The method is the same as the SE for (UT)AMP, i.e., $\mathbf{q}_k = \mathbf{x}_k + \mathbf{w}_k$ is modeled as the input to the "denoiser" (which corresponds to $f_{\mathbf{x}}$, $f_{\mathbf{b}}$ and $f_{\mathbf{c}}$ in the factor graph and involves EP and BP), where $\mathbf{w}_k$ denotes a Gaussian noise with mean zero and variance τ_k. However, it is difficult to find an analytic form for the output variance of the denoiser, which is also happened to (UT)AMP due to the priors. This can be solved by simulating the denoiser using $\mathbf{q}_k = \mathbf{x}_k + \mathbf{w}_k$ with different variances of $\mathbf{w}_k$ as input, so that a "function" in terms of a table can be established. In our case, besides the variance of $\mathbf{x}$, the MSE of $\mathbf{b}$ and $\mathbf{c}$ can also be obtained as "byproduct", which allows us to predict the MSE of $\mathbf{b}$ and $\mathbf{c}$, while the variance of $\mathbf{x}$ is used to determine τ_k analytically. As shown in Section 5.4, the prediction is fairly good in some cases. But, in some cases, it is not accurate. More accurate and rigorous performance analysis is our future work.

5.4 Numerical Examples

In this section, the performance of Bi-UAMP is evaluated and compare it with the state-of-the-art bilinear recovery algorithms including the conventional non-message passing based algorithm WSS-TLS in [11], and message passing based algorithms BAd-VAMP in [14] and PC-VAMP in [23]. It is noted that PC-VAMP does not provide an estimate for $\mathbf{b}$. Performance is evaluated in terms of normalized MSE and runtime. Relevant performance bounds are also included for reference.

5.4.1 SMV Case

For the SMV case, compressed sensing with matrix uncertainty [11] is taken as an example. It is the aim to recover a sparse signal vector $\mathbf{c}$ from measurement $\mathbf{y} = \mathbf{A}(\mathbf{b})\mathbf{c} + \mathbf{w}$, where the measurement matrix is modeled as $\mathbf{A}(\mathbf{b}) = \sum_{k=1}^{K} b_k \mathbf{A}_k$ with $b_1 = 1$, $\mathbf{A}_k \in \mathbb{R}^{M \times N}$ are known, and the uncertainty parameter vector $\mathbf{b} = [b_2, ..., b_K]^T$ is unknown. In addition the precision of the noise is unknown as well.

In the experiments, $K = 11$, $N = 256$, $M = 150$ are set and the number of nonzero

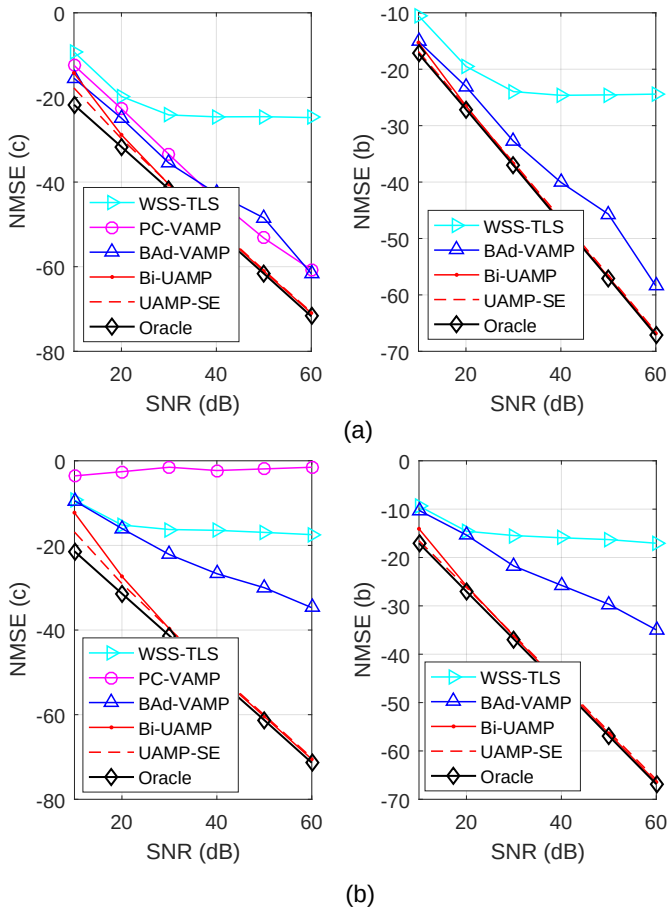

Figure 5.4: Compressive sensing with correlated matrices: NMSE of $\mathbf{b}$ and $\mathbf{c}$ versus SNR with (a) $\rho = 0.3$ and (b) $\rho = 0.4$.

elements in $\mathbf{c}$ is 10. The SNR is defined as SNR $\triangleq \mathbb{E}\left[\|\mathbf{A}(\mathbf{b})\mathbf{c}\|^2\right]/\mathbb{E}\left[\|\mathbf{w}\|^2\right]$. The uncertainty parameters $\{b_2, ...b_k\}$ are drawn from $\mathcal{N}(0,1)$ independently, and the nonzero elements of sparse vector $\mathbf{c}$ are drawn from $\mathcal{N}(0,1)$ independently as well, which are randomly located in $\mathbf{c}$. The performance of the methods are evaluated using NMSE($\mathbf{b}$) $\triangleq \|\hat{\mathbf{b}} - \mathbf{b}\|^2/\|\mathbf{b}\|^2$ and NMSE($\mathbf{c}$) $\triangleq \|\hat{\mathbf{c}} - \mathbf{c}\|^2/\|\mathbf{c}\|^2$, where $\hat{\mathbf{b}}$ and $\hat{\mathbf{c}}$ are the estimates of $\mathbf{b}$ and $\mathbf{c}$, respectively. The performance bounds for the estimation of $\mathbf{b}$ and $\mathbf{c}$ are included, which are the performance of two oracle estimators: the MMSE estimator for $\mathbf{b}$ with the assumption that $\mathbf{c}$ is known, and the MMSE estimator for $\mathbf{c}$ with the assumption that $\mathbf{b}$ and the support of $\mathbf{c}$ are known.

It is noted that, different from [14], median NMSEs is not used, and to better evaluate the robustness of the algorithms, the NMSEs are obtained by averaging the results from all trials. To demonstrate the robustness of Bi-UAMP, tough measurement matrices, e.g., correlated matrices, non-zero mean matrices. and ill-conditioned matrices, are focused on. In addition, Bi-UAMP and BAd-VAMP use a same damping factor of 0.8 to enhance

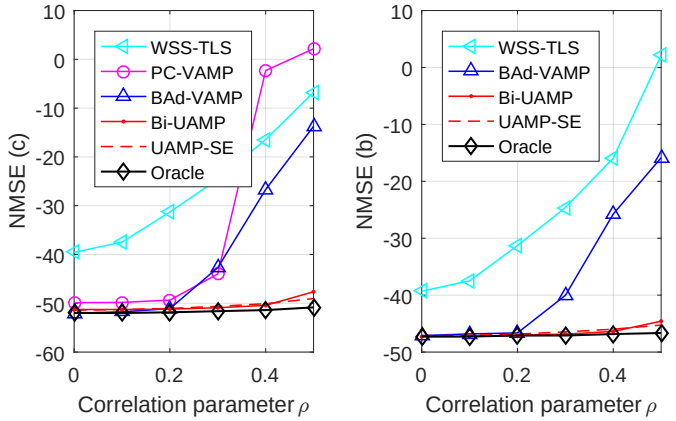

Figure 5.5: Compressive sensing with correlated matrices: NMSE of **b** and **c** versus ρ at SNR = 40dB.

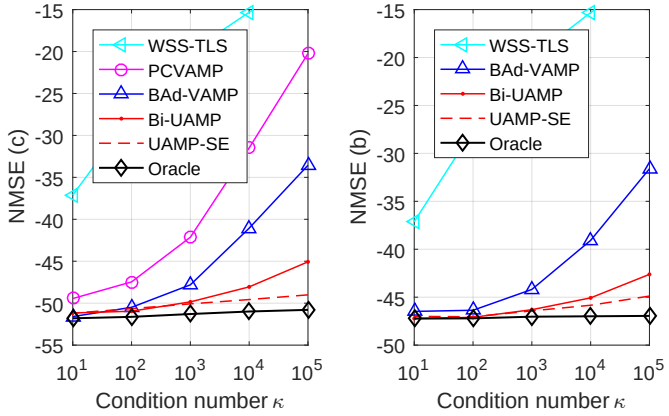

Figure 5.6: Compressive sensing with ill-conditioned matrices: NMSE of **b** and **c** versus κ with SNR = 40dB.

their robustness.

5.4.1.1 Correlated Measurement Matrix

All matrices $\{\mathbf{A}_k\}$ are correlated, and $\mathbf{A}_k$ is constructed using $\mathbf{A}_k = \mathbf{C}_L\mathbf{G}_k\mathbf{C}_R$, where $\mathbf{G}_k$ is an i.i.d. Gaussian matrix, and $\mathbf{C}_L$ is an $M \times M$ matrix with the (m, n)th element given by $\rho^{|m-n|}$ where $\rho \in [0, 1]$. Matrix $\mathbf{C}_R$ is generated in the same way but with a size of $N \times N$. The parameter ρ controls the correlation of matrix $\mathbf{A}_k$. Figure 5.4 shows the NMSE performance of the algorithms versus SNR, where the correlation parameter $\rho = 0.3$ in (a) and $\rho = 0.4$ in (b). It can be seen that when $\rho = 0.3$, all the message passing based algorithms PC-VAMP, BAd-VAMP and Bi-UAMP perform well and they are significantly better than the non-message passing based method WSS-TLS. It can also be seen that

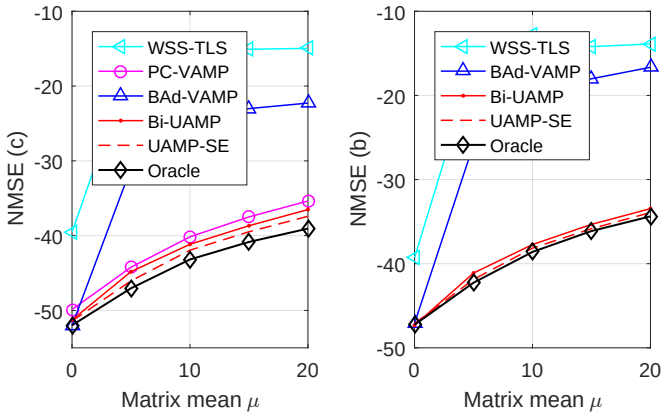

Figure 5.7: Compressive sensing with non-zero mean matrix: NMSE of **b** and **c** versus μ with SNR = 40dB.

Bi-UAMP delivers a performance which is considerably better than that of PC-VAMP and BAd-VAMP. With $\rho = 0.4$, Bi-UAMP still works very well, and it significantly outperforms BAd-VAMP, PC-VAMP and WSS-TLS. It is noted that as PC-VAMP does not estimate **b**, so its performance in the right column is absent.the performance of all algorithms for matrices with different level of correlations by varying the parameter ρ at SNR = 40dB are further evaluated and the results are shown in Figure 5.5, where it can be seen that significant performance gaps between all the other algorithms and Bi-UAMP when ρ is relatively large. The results in Figs. 5.4 and 5.5 demonstrate that Bi-UAMP is more robust than all the other algorithms with correlated measurement matrices. In Figs. 5.4 and 5.5, the predicted performance based on SE for Bi-UAMP is shown, where the predicted performance matches the simulated performance fairly well when the matrix correlation is relatively small.

5.4.1.2 Ill-Conditioned Measurement Matrix

Each matrix $\mathbf{A}_k$ is constructed based on the SVD $\mathbf{A}_k = \mathbf{U}_k\mathbf{\Lambda}_k\mathbf{V}_k$ where $\mathbf{\Lambda}_k$ is a singular value matrix with $\Lambda_{i,i}/\Lambda_{i+1,i+1} = \kappa^{1/(M-1)}$ (i.e., the condition number of the matrix is κ). The NMSE performance of the algorithms versus the condition number is shown in Figure 5.6, where the SNR = 40 dB. It can be seen that Bi-UAMP can significantly outperform all the other algorithms when κ is relatively large, and BAd-VAMP performs better than PC-VAMP and WSS-TLS. It also can be seen that the predicated performance is no longer accurate when κ is large.

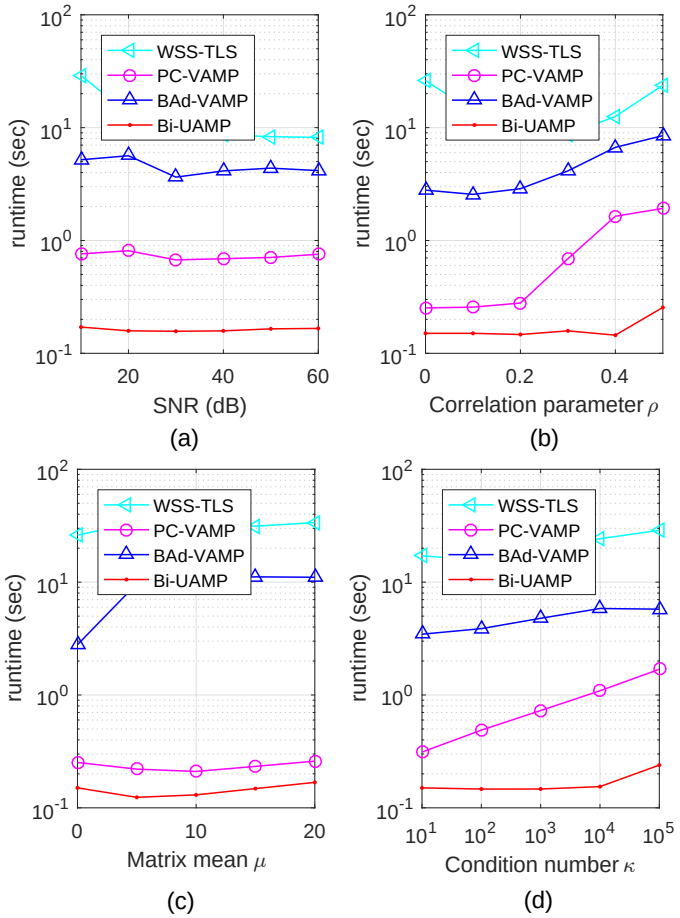

Figure 5.8: Average runtime versus (a) SNR for correlated matrices with $\rho = 0.3$, (b) ρ for correlated matrices, (c) μ for non-zero mean matrices, (d) condition number κ for ill-conditioned matrices. In (b), (c) and (d), SNR = 40 dB.

5.4.1.3 Non-Zero Mean Measurement Matrix

The elements of matrix $\mathbf{A}_k$ are independently drawn from a non-zero mean Gaussian distribution $\mathcal{N}(\mu, v)$. The mean μ measures the derivation from the i. i. d. zero-mean Gaussian matrix. In the simulations, for $\{\mathbf{A}_k, k = 2 : K\}$, $v = 1$, and for $\mathbf{A}_1$, $v = 20$, since these values perform well over a wide range of problems. The NMSE performance of the algorithms versus μ is shown in Figure 5.7, where the SNR = 40 dB. It can be seen from this figure that Bi-UAMP can achieve much better performance compared to WSS-TLS and BAd-VAMP especially when μ is relatively large. PC-VAMP delivers a competitive performance compared to Bi-UAMP, while it does not provide an estimate for $\mathbf{b}$ and is also slower than Bi-UAMP as shown in Figure 5.8.

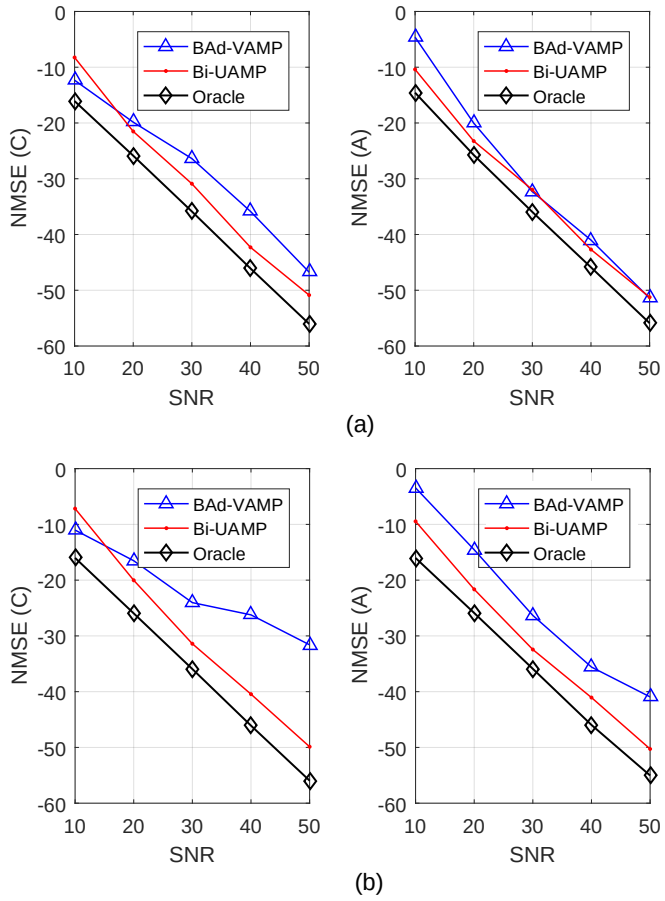

Figure 5.9: Structured dictionary learning: NMSE(**A**) and NMSE(**C**) versus SNR with (a) $\rho = 0$ and (b) $\rho = 0.1$.

5.4.1.4 Runtime Comparison

Figure 5.8 compares the average runtime of all algorithms. In Figure 5.8 (a), correlated matrices are used with the correlation parameter $\rho = 0.3$. With SNR = 40 dB, the average runtime versus different ρ for correlated matrices, different means for non-zero mean matrices and different condition numbers for ill-conditioned matrices is given in Figure 5.8 (b), (c) and (d), respectively. The results are obtained using MATLAB (R2016b) on a computer with a 6-core Intel i7 processor. Figure 5.8 shows that, Bi-UAMP is much faster than BAd-VAMP and WSS-TLS, and it is also considerably faster than PC-VAMP.

5.4.2 MMV Case

The structured dictionary learning (DL) [9] is taken as an example to demonstrate the performance of Bi-UAMP. The goal of structured DL is to find a structured dictionary

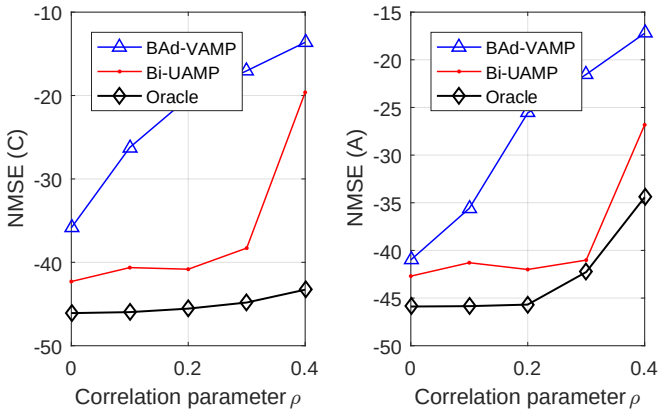

Figure 5.10: Structured dictionary learning: NMSE(A) and NMSE(C) versus ρ with SNR = 40dB.

matrix $\mathbf{A} = \sum_{k=1}^{K} b_k \mathbf{A}_k \in \mathbb{R}^{M \times N}$ from the training samples $\mathbf{Y} \in \mathbb{R}^{M \times L}$ with model $\mathbf{Y} = \mathbf{AC} + \mathbf{W}$ for some sparse coefficient matrix $\mathbf{C} \in \mathbb{R}^{N \times L}$. In the simulations, It is assumed square dictionary matrix $\mathbf{A}$ with $M = N = 100$. The length of vector $\mathbf{b}$ is large, i.e., $K = 100$, and the number of non-zero elements are set to be 20 in each column of $\mathbf{C}$ (the columns are generated independently) and $L = 5$ for the training examples. Since the dictionary matrix $\mathbf{A}$ has a structure, it can be learned with a small number of training samples. Bi-UAMP is run for maximum 100 iterations and 10 restarts. In addition, to enhance the robustness, a damping factor 0.55 for both Bi-UAMP and BAd-VAMP is used. In addition, Lines 19-22 in Bi-UAMP are executed once every two iterations. The performance is evaluated with NMSE of the estimates of $\mathbf{A}$ and $\mathbf{C}$. As the pair $(\mathbf{A}, \mathbf{C})$ is recoverable only up to an ambiguity: a scalar ambiguity in the structured case and a generalized permutation ambiguity in the unstructured [14]. The NMSE is calculated in the same way as in [14], i.e.,

$$\text{NMSE}(\hat{\mathbf{A}}) \triangleq \min_d \frac{\|\mathbf{A} - d\hat{\mathbf{A}}\|^2}{\|\mathbf{A}\|^2} \tag{5.79}$$

$$\text{NMSE}(\hat{\mathbf{C}}) \triangleq \min_d \frac{\|\mathbf{C} - d\hat{\mathbf{C}}\|^2}{\|\mathbf{C}\|^2}. \tag{5.80}$$

Different from [14], the NMSEs are obtained by averaging the results from all trials. To test the performance and robustness of the algorithms, correlated matrices $\{\mathbf{A}_k\}$ generated in the same way as in the SMV case are used.

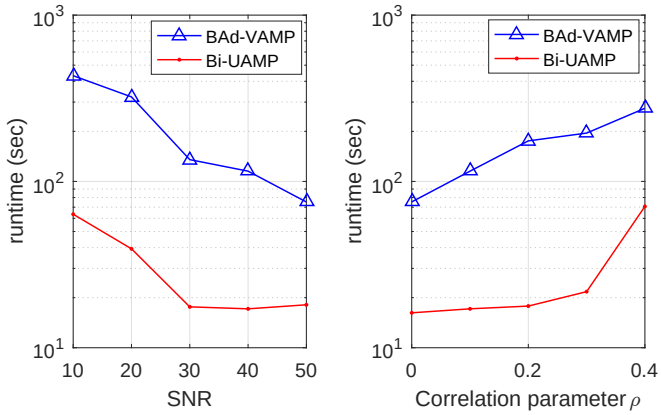

Figure 5.11: Structured dictionary learning: Average runtime versus SNR (left) and ρ (right).

Figure 5.9 shows the NMSE performance NMSE($\hat{\mathbf{A}}$) and NMSE($\hat{\mathbf{C}}$) versus SNR with correlation parameter (a) $\rho = 0$ and (b) $\rho = 0.1$. It can be seen that when $\rho = 0$, i.e., $\{\mathbf{A}_k\}$ are i.i.d. Gaussian, BAd-VAMP and Bi-UAMP have similar performance. When $\rho = 0.1$, Bi-UAMP can outperform BAd-UAMP considerably. Figure 5.10 shows the NMSE versus ρ at SNR = 40dB, where It can be seen that Bi-UAMP can achieve significantly better performance than BAd-VAMP. From these results, Bi-UAMP is more robust. Figure 5.11 shows the average runtime versus (a) SNR and (b) ρ. Again, the results show that Bi-UAMP is much faster than BAd-VAMP.

5.5 Conclusions

In this thesis, approximate Bayesian inference for the problem of bilinear recovery is investigated. A new approximate inference algorithm Bi-UAMP is designed, where UAMP is integrated with BP, EP and VMP to achieve efficient recovery of the unknown variables. It can be shown that Bi-UAMP is much more robust and faster than the state-of-the-art algorithms, leading to significantly better performance.

Chapter 6

Conclusions and Future Work

6.1 Conclusion

In this thesis, based on the UAMP algorithm, solutions for SBL and bilinear recovery problems are proposed in this thesis, which are developed to be with better efficiency and robustness compared to other state-of-the-art algorithms. The major contributions of this thesis are summarized in the following.

In Chapter 2, an overview of SBL and bilinear recovery problem and state-of-the-art works are surveyed. First of all, AMP is reviewed and various AMP-based techniques designed for solving the signal recovery problem are illustrated. Then, the SBL algorithm is detailed. Since SBL uses matrix inversions at each iteration, its complexity is too high for large-scale problems. After that, AMP and its variants used for the low complexity implementation of SBL are reviewed. Lastly, the bilinear recovery problem is presented. Since AMP based algorithms are vulnerable to difficult A matrices. To achieve robust bilinear recovery, existing works on conventional non-message passing based algorithm and message passing based algorithms are discussed.

In Chapter 3, VI, AMP and UAMP are described. First of all, a brief introduction of variational inference is described. Then the AMP algorithm which was developed based on the loopy BP and has low complexity, is explained. However, the convergence of AMP cannot be guaranteed. Thus, UAMP algorithm which is more efficient and robust algorithms is introduced. To be more specific, two versions of UAMP algorithm are reviewed

in detail. The empirical SE-based performance prediction for UAMP is investigated. The state evolution equation is simple but effective. Leveraging the SE of UAMP, how to predict the performance of UAMP-SBL empirically is also investigated. The UAMP-SBL is treated as UAMP with a special denoiser, enabling the use of UAMP-SE to predict the performance of UAMP-SBL

In Chapter 4, leveraging UAMP, UAMP-SBL is proposed for sparse signal recovery with the framework of structured variational inference, which inherits the low complexity and robustness of UAMP against a generic measurement matrix. It is demonstrated that, compared to the state-of-the-art AMP based SBL algorithm, UAMP-SBL can achieve much better performance in terms of robustness, speed and recovery accuracy.

In Chapter 5, approximate Bayesian inference for the problem of bilinear recovery is investigated. A novel approximate inference algorithm Bi-UAMP is designed, where UAMP is integrated with BP, EP and VMP to achieve efficient recovery of unknown variables. It is shown that Bi-UAMP is much more robust and faster than the other state-of-the-art algorithms, leading to significantly better performance.

6.2 Future work

This thesis has made successful investigations on the problems of SBL and bilinear recovery based on UAMP. However, there are still further works worth investigations.

- The performance of UAMP matches well with the empirical state evolution. A rigorous theoretical analysis of the state evolution of UAMP is the next step.

- It is discovered in our work that the update of ϵ is crucial for the SBL algorithms to achieve support-oracle bound and we have proposed an empirical update. Future work includes rigorous analyses of the state evolution of UAMP-SBL and the update mechanism of the shape parameter.

- In Bi-UAMP, future work includes a rigorous analysis of the performance of Bi-UAMP and generalizing it to handle nonlinear measurements, e.g., quantization.

By taking advantage of the robustness and low complexity of UAMP, UAMP will be applied to solving more challenging problems in communications and radar.

Bibliography

(1) J. H. Ender, "On compressive sensing applied to radar", *Signal Processing*, 2010, **90**, 1402–1414.

(2) D. Malioutov, M. Cetin and A. S. Willsky, "A sparse signal reconstruction perspective for source localization with sensor arrays", *IEEE transactions on signal processing*, 2005, **53**, 3010–3022.

(3) M. Lustig, D. L. Donoho, J. M. Santos and J. M. Pauly, "Compressed sensing MRI", *IEEE signal processing magazine*, 2008, **25**, 72–82.

(4) I. F. Gorodnitsky, J. S. George and B. D. Rao, "Neuromagnetic source imaging with FOCUSS: a recursive weighted minimum norm algorithm", *Electroencephalography and clinical Neurophysiology*, 1995, **95**, 231–251.

(5) Z. Zhang and B. D. Rao, "Clarify some issues on the sparse Bayesian learning for sparse signal recovery", *University of California, San Diego, Tech. Rep*, 2011.

(6) M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine", *Journal of machine learning research*, 2001, **1**, 211–244.

(7) D. P. Wipf and B. D. Rao, "Sparse Bayesian learning for basis selection", *IEEE Transactions on Signal processing*, 2004, **52**, 2153–2164.

(8) M. Al-Shoukairi, P. Schniter and B. D. Rao, "A GAMP-based low complexity sparse Bayesian learning algorithm", *IEEE Transactions on Signal Processing*, 2018, **66**, 294–308.

(9) R. Rubinstein, A. M. Bruckstein and M. Elad, "Dictionaries for sparse representation modeling", *Proceedings of the IEEE*, 2010, **98**, 1045–1057.

(10) S. Ling and T. Strohmer, "Self-Calibration and Biconvex Compressive Sensing", *Inverse Problems*, 2015, **31**, DOI: 10.1088/0266-5611/31/11/115002.

(11) H. Zhu, G. Leus and G. B. Giannakis, "Sparsity-Cognizant Total Least-Squares for Perturbed Compressive Sampling", *IEEE Transactions on Signal Processing*, 2011, **59**, 2002–2016.

(12) Z. Lin, M. Chen and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices", *arXiv preprint arXiv:1009.5055*, 2010.

(13) S. Sarkar, A. K. Fletcher, S. Rangan and P. Schniter, "Bilinear recovery using adaptive vector-AMP", *IEEE Transactions on Signal Processing*, 2019, **67**, 3383–3396.

(14) S. Sarkar, A. K. Fletcher, S. Rangan and P. Schniter, "Bilinear Recovery Using Adaptive Vector-AMP", *IEEE Transactions on Signal Processing*, 2019, **67**, 3383–3396.

(15) S. Rangan, "Generalized approximate message passing for estimation with random linear mixing", *International Symposium on Information Theory Proceedings*, 2011, 2168–2172.

(16) J. T. Parker, P. Schniter and V. Cevher, "Bilinear Generalized Approximate Message Passing Part II: Applications", *IEEE Transactions on Signal Processing*, 2014, **62**, 5854–5867.

(17) J. T. Parker and P. Schniter, "Parametric Bilinear Generalized Approximate Message Passing", *IEEE Journal of Selected Topics in Signal Processing*, 2016, **10**, 795–808.

(18) M. Davenport and J. Romberg, "An Overview of Low-Rank Matrix Recovery From Incomplete Observations", *IEEE Journal of Selected Topics in Signal Processing*, 2016, **10**, 1–1.

(19) A. Ahmed, B. Recht and J. Romberg, "Blind Deconvolution Using Convex Programming", *IEEE Transactions on Information Theory*, 2014, **60**, 1711–1732.

(20) S. Rangan, P. Schniter, A. K. Fletcher and S. Sarkar, "On the convergence of approximate message passing with arbitrary matrices", *IEEE Transactions on Information Theory*, 2019, **65**, 5339–5351.

(21) S. Rangan, P. Schniter and A. K. Fletcher, "Vector Approximate Message Passing", *IEEE Transactions on Information Theory*, 2019, **65**, 6664–6684.

(22) A. K. Fletcher, P. Pandit, S. Rangan, S. Sarkar and P. Schniter, "Plug-in estimation in high-dimensional linear inverse problems: A rigorous analysis", *Advances in Neural Information Processing Systems*, 2018, **31**.

(23) J. Zhu, Q. Zhang, X. Meng and Z. Xu, "Vector approximate message passing algorithm for compressed sensing with structured matrix perturbation", *Signal Processing*, 2020, **166**, 107248.

(24) Q. Guo, D. D. Huang, S. Nordholm, J. Xi and Y. Yu, "Iterative frequency domain equalization with generalized approximate message passing", *IEEE Signal Processing Letters*, 2013, **20**, 559–562.

(25) Q. Guo and J. Xi, "Approximate Message Passing with Unitary Transformation", *CoRR*, 2015, **abs/1504.04799**.

(26) H. Kang, J. Li, Q. Guo and M. Martorella, "Pattern Coupled Sparse Bayesian Learning Based on UTAMP for Robust High Resolution ISAR Imaging", *IEEE Sensors Journal*, 2020, **20**, 13734–13742.

(27) Y. Mao, M. Luo, D. Gao and Q. Guo, "Low complexity DOA estimation using AMP with unitary transformation and iterative refinement", *Digital Signal Processing*, 2020, 102800.

(28) Z. Yuan, F. Liu, W. Yuan, Q. Guo, Z. Wang and J. Yuan, "Iterative detection for orthogonal time frequency space modulation with unitary approximate message passing", *IEEE transactions on wireless communications*, 2021.

(29) J. Ma and L. Ping, "Orthogonal AMP", *IEEE Access*, 2017, **5**, 2020–2033.

(30) D. L. Donoho, A. Maleki and A. Montanari, "Message-passing algorithms for compressed sensing", *Proceedings of the National Academy of Sciences*, 2009, **106**, 18914–18919.

(31) A. Javanmard and A. Montanari, "State evolution for general approximate message passing algorithms, with applications to spatial coupling", *Information and Inference: A Journal of the IMA*, 2013, **2**, 115–144.

(32) R. Tibshirani, "Regression shrinkage and selection via the lasso: a retrospective", *Journal of the Royal Statistical Society: Series B (Statistical Methodological)*, 2011, **73**, 273–282.

(33) T. Blumensath and M. E. Davies, "Normalized iterative hard thresholding: Guaranteed stability and performance", *IEEE Journal of selected topics in signal processing*, 2010, **4**, 298–309.

(34) M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing", *IEEE Transactions on Information Theory*, 2011, **57**, 764–785.

(35) X. Meng, S. Wu and J. Zhu, "A unified Bayesian inference framework for generalized linear models", *IEEE Signal Processing Letters*, 2018, **25**, 398–402.

(36) A. Manoel, F. Krzakala, E. W. Tramel and L. Zdeborová, Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, JMLR.org, Lille, France, 2015, pp. 1123–1132.

(37) J. Vila, P. Schniter, S. Rangan, F. Krzakala and L. Zdeborová, Proceedings of 40th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 2021–2025.

(38) L. Liu, S. Huang and B. M. Kurkoski, 2021 IEEE International Symposium on Information Theory (ISIT), 2021, pp. 1379–1384.

(39) K. Takeuchi, "Convolutional approximate message-passing", *IEEE Signal Processing Letters*, 2020, **27**, 416–420.

(40) T. P. Minka, "Expectation Propagation for Approximate Bayesian Inference", *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, 2001, 362–369.

(41) K. Takeuchi, "On the Convergence of Orthogonal/Vector AMP: Long-Memory Message-Passing Strategy", *arXiv preprint arXiv:2111.05522*, 2021.

(42) K. Takeuchi, "Bayes-optimal convolutional AMP", *IEEE Transactions on Information Theory*, 2021.

(43) J. Palmer, D. Wipf, K. Kreutz-Delgado and B. Rao, "Variational EM algorithms for non-Gaussian latent variable models", *Advances in neural information processing systems*, 2006, **18**, 1059.

(44) D. F. Andrews and C. L. Mallows, "Scale mixtures of normal distributions", *Journal of the Royal Statistical Society: Series B (Methodological)*, 1974, **36**, 99–102.

(45) Ghassan Kawas Kaleh and R. Vallet, "Joint parameter estimation and symbol detection for linear or nonlinear unknown channels", *IEEE Transactions on Communications*, 1994, **42**, 2406–2413.

(46) R. Berthier, A. Montanari and P.-M. Nguyen, "State evolution for approximate message passing with non-separable functions", *Information and Inference: A Journal of the IMA*, 2020, **9**, 33–79.

(47) M. Bayati, M. Lelarge and A. Montanari, "Universality in polytope phase transitions and message passing algorithms", *The Annals of Applied Probability*, 2015, **25**, 753–822.

(48) X. Meng and J. Zhu, "Bilinear Adaptive Generalized Vector Approximate Message Passing", *IEEE Access*, 2019, **7**, 4807–4815.

(49) G. Parisi and R. Shankar, "Statistical field theory", *Physics Today*, 1988, **41**, 110.

(50) M. I. Jordan, Z. Ghahramani, T. S. Jaakkola and L. K. Saul, "An introduction to variational methods for graphical models", *Machine learning*, 1999, **37**, 183–233.

(51) M. Svensén and C. M. Bishop, *Pattern recognition and machine learning*, 2007.

(52) J. Winn and C. M. Bishop, "Variational message passing", *Journal of Machine Learning Research*, 2005, **6**, 661–694.

(53) J. Dauwels, "On variational message passing on factor graphs", *Proceedings of international symposium on information theory*, 2007, 2546–2550.

(54) E. P. Xing, M. I. Jordan and S. Russell, "A generalized mean field algorithm for variational inference in exponential families", *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, 2002, 583–591.

(55) D. L. Donoho et al., "Compressed sensing", *IEEE Transactions on Information Theory*, 2006, **52**, 1289–1306.

(56) S. Kullback and R. A. Leibler, "On information and sufficiency", *The annals of mathematical statistics*, 1951, **22**, 79–86.

(57) C. Peterson, "A mean field theory learning algorithm for neural networks", *Complex systems*, 1987, **1**, 995–1019.

(58) C. A. Metzler, A. Maleki and R. G. Baraniuk, "From denoising to compressed sensing", *IEEE Transactions of Information Theory*, 2016, **62**, 5117–5144.

(59) A. M. Tulino, G. Caire, S. Verdu and S. Shamai, "Support recovery with sparsely sampled free random matrices", *IEEE Transactions on Information Theory*, 2013, **59**, 4243–4271.

(60) D. L. Donoho, A. Maleki and A. Montanari, "The noise-sensitivity phase transition in compressed sensing", *IEEE Transactions on Information Theory*, 2011, **57**, 6920–6941.

(61) M. Luo, Q. Guo, D. Huang and J. Xi, "Sparse Bayesian learning based on approximate message passing with unitary transformation", *VTS Asia Pacific Wireless Communications Symposium (APWCS)*, 2019, 1–5.

(62) Y. C. Eldar and G. Kutyniok, *Compressed sensing: theory and applications*, 2012.

(63) M. F. Duarte and Y. C. Eldar, "Structured compressed sensing: From theory to applications", *IEEE Transactions on Signal Processing*, 2011, **59**, 4053–4085.

(64) J. Liu and B. D. Rao, "Sparse Bayesian learning for robust PCA: Algorithms and analyses", *IEEE Transactions on Signal Process.*, 2019, **67**, 5837–5849.

(65) M. Elad, *Sparse and redundant representations: from theory to applications in signal and image processing*, Springer, 2010.

(66) E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?", *IEEE Transactions on Information Theory*, 2006, **52**, 5406–5425.

(67) F. Wen, L. Pei, Y. Yang, W. Yu and P. Liu, "Efficient and robust recovery of sparse signal and image using generalized nonconvex regularization", *IEEE Transactions on Computational Imaging*, 2017, **3**, 566–579.

(68) Y. Zhang, Q. Guo, Z. Wang, J. Xi and N. Wu, "Block sparse Bayesian learning based joint user activity detection and channel estimation for grant-free NOMA systems", *IEEE Transactions on Vehicular Technology*, 2018, **67**, 9631–9640.

(69) D. L. Donoho, A. Maleki and A. Montanari, "Message passing algorithms for compressed sensing: I. motivation and construction", *IEEE Information Theory Workshop on Information Theory (ITW 2010, Cairo)*, 2010, 1–5.

(70) M. Al-Shoukairi and B. Rao, "Sparse Bayesian learning using approximate message passing", *Proceedings of 48th Asilomar Conference on Signals, Systems and Computers*, 2014, 1957–1961.

(71) J. Zhu, L. Han and X. Meng, "An AMP-based low complexity generalized sparse Bayesian learning algorithm", *IEEE Access*, 2018, **7**, 7965–7976.

(72) Y. Jin and B. D. Rao, "Support recovery of sparse signals in the presence of multiple measurement vectors", *IEEE Transactions on Information Theory*, 2013, **59**, 3139–3157.

(73) M. E. Davies and Y. C. Eldar, "Rank awareness in joint sparse recovery", *IEEE Transactions on Information Theory*, 2012, **58**, 1135–1146.

(74) H. Kang, J. Li, Q. Guo and M. Martorella, "Pattern coupled sparse Bayesian learning based on UTAMP for robust high resolution ISAR Imaging", *IEEE Sensors Journal*, 2020, **20**, 13734–13742.

(75) N. L. Pedersen, C. N. Manchón, D. Shutin and B. H. Fleury, "Application of Bayesian hierarchical prior modeling to sparse channel estimation", *IEEE International Conference on Communications*, 2012, 3487–3492.

(76) R. L. Devaney, *A first course in chaotic dynamical systems: theory and experiment*, Chapman and Hall/CRC, 2020.

(77) Z. Zhang and B. D. Rao, "Sparse signal recovery in the presence of correlated multiple measurement vectors", *Proceedings of 35th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2010, 3986–3989.

(78) Z. Zhang and B. D. Rao, "Sparse signal recovery with temporally correlated source vectors using sparse Bayesian learning", *IEEE Journal of Selected Topics in Signal Processing*, 2011, **5**, 912–926.

(79) A. K. Fletcher and P. Schniter, "Learning and free energies for vector approximate message passing", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, 4247–4251.

(80) D. P. Wipf and B. D. Rao, "An empirical Bayesian strategy for solving the simultaneous sparse approximation problem", *IEEE Transactions on Signal Processing*, 2007, **55**, 3704–3716.

(81) R. L. Eubank, *A Kalman filter primer*, CRC Press, 2005.

(82) D. L. Donoho, A. Maleki and A. Montanari, "Message passing algorithms for compressed sensing: II. analysis and validation", *IEEE Information Theory Workshop on Information Theory (ITW 2010, Cairo)*, 2010, 1–5.

(83) J. Pearl, *Reverend Bayes on inference engines: A distributed hierarchical approach*, Cognitive Systems Laboratory, School of Engineering and Applied Science, University of California, Los Angeles., 1982.

(84) F. R. Kschischang, B. J. Frey and H.-A. Loeliger, "Factor graphs and the sum-product algorithm", *IEEE Transactions on Information Theory*, 2001, **47**, 498–519.

(85) C. Zhang, Z. Yuan, Z. Wang and Q. Guo, "Low complexity sparse Bayesian learning using combined belief propagation and mean field with a stretched factor graph", *Signal Processing*, 2017, **131**, 344–349.

(86) Z. Yuan, C. Zhang, Z. Wang, Q. Guo and J. Xi, "An Auxiliary Variable-Aided Hybrid Message Passing Approach to Joint Channel Estimation and Decoding for MIMO-OFDM", *IEEE Signal Processing Letters*, 2017, **24**, 12–16.

(87) D. Geiger and C. Meek, "Structured variational inference procedures and their realizations", *Proceedings of 10th International Workshop on Artificial Intelligence and Statistics.*, 2005.

(88) F. Krzakala, M. Mézard, F. Sausset, Y. Sun and L. Zdeborová, "Probabilistic reconstruction in compressed sensing: algorithms, phase diagrams, and threshold achieving matrices", *Journal of Statistical Mechanics: Theory and Experiment*, 2012, **61**, P08009.

(89) J. P. Vila and P. Schniter, "Expectation-Maximization Gaussian-Mixture Approximate Message Passing", *IEEE Transactions on Signal Processing*, 2013, **61**, 4658–4672.

(90) X. Wang and H. V. Poor, "Iterative (turbo) soft interference cancellation and decoding for coded CDMA", *IEEE Transactions on Communications*, 1999, **47**, 1046–1061.