Unitary Approximate Message Passing for Sparse Bayesian Learning

Man Luo, Qinghua Guo, *Senior Member, IEEE*, Ming Jin, *Member, IEEE*, Yonina C. Eldar, *Fellow, IEEE*, Defeng Huang, *Senior Member, IEEE*, and Xiangming Meng

***Abstract*—Sparse Bayesian learning (SBL) can be implemented with low complexity based on the approximate message passing (AMP) algorithm. However, it does not work well for a generic measurement matrix, which may cause AMP to diverge. Damped AMP has been used for SBL to alleviate divergence issues at the cost of reducing convergence speed. In this work, we propose a new SBL algorithm based on structured variational inference, leveraging AMP with a unitary transformation. Both single measurement vector and multiple measurement vector problems are investigated. 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.**

***Index Terms*—Sparse Bayesian learning, structured variational inference, approximate message passing.**

I. Introduction

We consider 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 [1]. This problem finds numerous applications in various areas of signal processing, statistics and computer science [1]–[7]. One approach to recovering $\mathbf{x}$ is to use sparse Bayesian learning (SBL), where $\mathbf{x}$ is assumed to have a sparsity-promoting prior [8]. 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 [9], [10] has been proposed for low-complexity implementation of SBL [11], [12]. AMP was originally developed for compressive sensing based on loopy belief propagation (BP) [10]. Compared to convex optimization based algorithms such as LASSO [13] and greedy algorithms such as iterative hard-thresholding [14], 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 $\mathbf{A}$ [15]. AMP was later extended in [16] to solve general estimation problems with a generalized linear observation model [17]. By implementing the E-step using AMP in the expectation maximization (EM) based SBL method, matrix inversion can be avoided, leading to a significant reduction in computational complexity. However, AMP does not work well for a generic matrix such as non-zero mean, rank-deficient, correlated, or ill-conditioned matrix $\mathbf{A}$ [18], resulting in divergence and poor performance.

Many variants of AMP have been proposed to address the divergence issue and achieve better robustness to a generic $\mathbf{A}$, such as the damped AMP [18], swept AMP [19], generalized approximate message passing algorithm (GAMP) with adaptive damping [20], AMP with unitary transformation (UTAMP) [21], vector AMP (VAMP) [22], orthogonal AMP [23], memory AMP [24], convolutional AMP [25] and more. In [26], 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.

For a general linear inverse problem, it was proposed in [21], [27] to apply AMP to a unitary transform of the original model, where the unitary matrix for the transformation can be obtained by the singular value decomposition (SVD) of $\mathbf{A}$. In the case of a circulant $\mathbf{A}$, the normalized discrete Fourier transform matrix can be used for the unitary transformation, enabling highly efficient implementation with the fast Fourier transform algorithm [28]. This leads to the AMP variant UTAMP, which is renamed as unitary AMP (UAMP) in this paper.[1] Here, we apply this concept to SBL, resulting in a new SBL algorithm called UAMP-SBL. UAMP-SBL achieves more efficient sparse signal recovery with significantly enhanced

Manuscript received January 21, 2021; revised June 13, 2021 and September 2, 2021; accepted September 8, 2021. Date of publication September 24, 2021; date of current version November 11, 2021. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Chiara Ravazzi. This work was supported in part by Australian Research Council Discovery Project under Grant DP190100786. The work of Ming Jin was supported in part by the Zhejiang Provincial Natural Science Funds for Distinguished Young Scholars under Grant LR21F010001. The work of Yonina C. Eldar was supported in part by the QuantERA under Grant C'MON-QSENS!. This work was presented in part at the 16th IEEE APWCS 2019 [DOI: 10.1109/VTS-APWCS.2019.8851644]. *(Corresponding author: Qinghua Guo.)*

Man Luo and Qinghua Guo are with the School of Electrical Computer and Telecommunications Engineering, University of Wollongong, Wollongong, NSW 2522, Australia (e-mail: ml857@uowmail.edu.au; qguo@uow.edu.au).

Ming Jin is with the Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo 315211, China (e-mail: jinming@nbu.edu.cn).

Yonina C. Eldar is with the Faculty of Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 7610001, Israel (e-mail: yonina.eldar@weizmann.ac.il).

Defeng Huang is with the School of Engineering, University of Western Australia, Perth 6009, Australia (e-mail: david.huang@uwa.edu.au).

Xiangming Meng is with the Institute for Physics of Intelligence, University of Tokyo, Hongo, Tokyo 113-0033, Japan (e-mail: meng@g.ecc.u-tokyo.ac.jp).

Digital Object Identifier 10.1109/TSP.2021.3114985

[1] SVD plays an important role in both UAMP and VAMP. In UAMP, SVD is used to obtain the unitary transformed model that AMP works with. SVD also connects VAMP and AMP in its analysis. In addition, the linear minimum mean squared error (LMMSE) estimator in VAMP results in cubic complexity in each iteration, and VAMP relies on SVD to implement the LMMSE estimator with low complexity.

robustness, compared to the state-of-the-art AMP-based SBL algorithm GGAMP-SBL [26].

To develop UAMP-SBL, we apply structured variational inference (SVI) [29]–[31]. 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) [30]–[32]. The use of SVMP allows the incorporation of UAMP to the message passing algorithm to handle the most computationally 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. We also investigate empirical SE-based performance prediction for UAMP-SBL and analyze the impact of the shape parameter on SBL. In addition, the UAMP-SBL algorithm is extended from single measurement vector (SMV) problems to multiple measurement vector (MMV) problems [2], [33], [34]. Based on our preliminary results in [35][2], UAMP-SBL was applied to inverse synthetic aperture radar [36], where the measurement matrix can be highly correlated in order to achieve high Doppler resolution. Real data experiments in [36] demonstrate its superiority in terms of both recovery performance and speed.

The rest of the paper is organized as follows. We briefly introduce SBL and (U)AMP in Section II. In Section III, UAMP-SBL is derived for SMV problems and empirical SE-based performance prediction for UAMP-SBL is also discussed. The impact of the shape parameter is analyzed in Section IV. UAMP-SBL is extended to the MMV setting in Section V. Numerical results are provided in Section VI, followed by conclusions in Section VII.

Throughput the paper, we use boldface lowercase and uppercase letters 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. We use $\mathbf{1}$ and $\mathbf{0}$ to denote the all-ones vector and all-zeros vector with proper sizes, respectively. 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. We use $|\cdot|^{\cdot 2}$ to denote the element-wise magnitude squared operation, and $\|\cdot\|$ for the l_2 norm. 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 $E[\cdot]$ is the expectation over all random variables involved in the brackets. We use $Diag(\mathbf{a})$ 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 $\mathbf{a}$. 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.

II. Background

A. Sparse Bayesian Learning

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

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{w}, \tag{1}$$

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

$$p(\mathbf{x}|\boldsymbol{\gamma}) = \prod_n p(x_n|\gamma_n) = \prod_n \mathcal{N}(x_n|0, \gamma_n^{-1}), \tag{2}$$

$$p(\boldsymbol{\gamma}) = \prod_n p(\gamma_n) = \prod_n \mathrm{Ga}(\gamma_n|\epsilon, \eta), \tag{3}$$

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

$$p(x_n) = \int \mathcal{N}(x_n|0, \gamma_n^{-1}) p(\gamma_n) d\gamma_n, \tag{4}$$

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

In the conventional SBL algorithm by Tipping [8], the precision vector $\boldsymbol{\gamma}$ is learned by maximizing the a posteriori probability

$$p(\boldsymbol{\gamma}|\mathbf{y}) \propto p(\mathbf{y}|\boldsymbol{\gamma}) p(\boldsymbol{\gamma}), \tag{5}$$

where the marginal likelihood function is

$$p(\mathbf{y}|\boldsymbol{\gamma}) = \int p(\mathbf{y}|\mathbf{x}) p(\mathbf{x}|\boldsymbol{\gamma}) d\mathbf{x}. \tag{6}$$

It can be shown that [8]

$$\log p(\mathbf{y}|\boldsymbol{\gamma}) = \frac{1}{2}\Big(\log|\boldsymbol{\Sigma}| + \log|Diag(\boldsymbol{\gamma})| - \boldsymbol{\zeta}^H Diag(\boldsymbol{\gamma})\boldsymbol{\zeta} \Big) + const, \tag{7}$$

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

$$\boldsymbol{\Sigma} = \left(\beta \mathbf{A}^H \mathbf{A} + Diag(\boldsymbol{\gamma})\right)^{-1}, \tag{8}$$

$$\boldsymbol{\zeta} = \beta \boldsymbol{\Sigma} \mathbf{A}^H \mathbf{y}. \tag{9}$$

The posterior probability of $\mathbf{x}$ is given by

$$p(\mathbf{x}|\mathbf{y}, \boldsymbol{\gamma}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\zeta}, \boldsymbol{\Sigma}). \tag{10}$$

Taking the logarithm of $p(\boldsymbol{\gamma}|\mathbf{y})$ and ignoring terms independent of $\boldsymbol{\gamma}$, learning of $\boldsymbol{\gamma}$ reduces to maximizing the following objective function [8]

$$\mathcal{L}(\boldsymbol{\gamma}) = \log p(\mathbf{y}|\boldsymbol{\gamma}) + \sum_{n=1}^{N} (\epsilon \log \gamma_n - \eta\gamma_n). \tag{11}$$

As the value of $\boldsymbol{\gamma}$ that maximizes $\mathcal{L}(\boldsymbol{\gamma})$ cannot be obtained in closed form, iterative re-estimation is employed by taking advantage of (7), i.e., with a learned $\boldsymbol{\gamma}$ in the last iteration, compute $\boldsymbol{\Sigma}$ and $\boldsymbol{\zeta}$ using (8) and (9), then update $\boldsymbol{\gamma}$ by maximizing $\mathcal{L}(\boldsymbol{\gamma})$ with (7), which leads to

$$\gamma_n = (2\epsilon + 1)/(2\eta + |\zeta_n|^2 + \Sigma_{n,n}), n = 1, \ldots, N. \tag{12}$$

In summary, Tipping's SBL algorithm (which is called SBL hereafter) executes the following iteration [8]:

Repeat

$$\mathbf{Z} = \left(\beta \mathbf{A}^H \mathbf{A} + Diag(\hat{\boldsymbol{\gamma}})\right)^{-1} \tag{13}$$

$$\hat{\mathbf{x}} = \beta \mathbf{Z} \mathbf{A}^H \mathbf{y} \tag{14}$$

$$\hat{\gamma}_n = (2\epsilon + 1)/(2\eta + |\hat{x}_n|^2 + Z_{n,n}), n = 1, \ldots, N. \tag{15}$$

Until terminated

If the noise precision β is unknown, then its estimation can be incorporated as well. SBL can also be derived based on the

[2]Compared to [35], we present a new derivation of UAMP-SBL, extend it from SMV to MMV, and provide theoretical analyses and comprehensive comparisons.

Algorithm 1: UAMP (UAMPv2 Executes Operations in [ ]).

Initialize $\boldsymbol{\tau}_x^{(0)}$(or $\tau_x^{(0)}$) > 0 and $\mathbf{x}^{(0)}$. Set $\mathbf{s}^{(-1)} = \mathbf{0}$ and $t = 0$. Define vector $\boldsymbol{\lambda} = \boldsymbol{\Lambda}\boldsymbol{\Lambda}^H\mathbf{1}$.

Repeat

1: $\boldsymbol{\tau}_p = |\boldsymbol{\Phi}|^{\cdot 2}\boldsymbol{\tau}_x^t$ [or $\boldsymbol{\tau}_p = \tau_x^t\boldsymbol{\lambda}$]
2: $\mathbf{p} = \boldsymbol{\Phi}\mathbf{x}^t - \boldsymbol{\tau}_p \cdot \mathbf{s}^{t-1}$
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: $\mathbf{1}./\boldsymbol{\tau}_q = |\boldsymbol{\Phi}^H|^{\cdot 2}\boldsymbol{\tau}_s$ [or $\mathbf{1}./\boldsymbol{\tau}_q = (\frac{1}{N}\boldsymbol{\lambda}^H\boldsymbol{\tau}_s)\mathbf{1}$]
6: $\mathbf{q} = \mathbf{x}^t + \boldsymbol{\tau}_q \cdot (\boldsymbol{\Phi}^H\mathbf{s}^t)$
7: $\boldsymbol{\tau}_x^{t+1} = \boldsymbol{\tau}_q \cdot g_x'(\mathbf{q}, \boldsymbol{\tau}_q)$ [or $\tau_x^{t+1} = \frac{1}{N}\mathbf{1}^H(\boldsymbol{\tau}_q \cdot g_x'(\mathbf{q}, \tau_q))$]
8: $\mathbf{x}^{t+1} = g_x(\mathbf{q}, \boldsymbol{\tau}_q)$
9: $t = t + 1$

Until terminated

EM algorithm [8], [26]. The SBL algorithm requires a matrix inverse in (13) in each iteration, resulting in cubic complexity per iteration.

B. (U)AMP

AMP was derived based on loopy BP with Gaussian and Taylor-series approximations [10], [16]. It is known that AMP can easily diverge in the case of a generic measurement matrix $\mathbf{A}$ [18]. Inspired by the work in [28], it was shown in [21] that the robustness of AMP is remarkably improved through simple pre-processing, i.e., performing a unitary transformation to the original linear model [21], [27]. As any matrix $\mathbf{A}$ has an SVD $\mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}$ with $\mathbf{U}$ and $\mathbf{V}$ being two unitary matrices, performing a unitary transformation with $\mathbf{U}^H$ leads to the following model

$$\mathbf{r} = \boldsymbol{\Phi}\mathbf{x} + \boldsymbol{\omega}, \tag{16}$$

where $\mathbf{r} = \mathbf{U}^H\mathbf{y}$, $\boldsymbol{\Phi} = \mathbf{U}^H\mathbf{A} = \boldsymbol{\Lambda}\mathbf{V}$, $\boldsymbol{\Lambda}$ is an $M \times N$ rectangular diagonal matrix, and $\boldsymbol{\omega} = \mathbf{U}^H\mathbf{w}$ remains a zero-mean Gaussian noise vector with the same covariance matrix $\beta^{-1}\mathbf{I}$. Applying the vector step size AMP [16] with model (16) leads to the first version of UAMP (called UAMPv1) shown in Algorithm 1.[3]

We can apply an average operation to two vectors: $\boldsymbol{\tau}_x$ in Line 7 and $|\boldsymbol{\Phi}^H|^{\cdot 2}\boldsymbol{\tau}_s$ in Line 5 of UAMPv1 in Algorithm 1, leading to the second version of UAMP [21] (called UAMPv2), where the operations in the brackets of Lines 1, 5 and 7 are executed (refer to [27] for the derivation). Compared to AMP and UAMPv1, UAMPv2 does not require matrix-vector products in Lines 1 and 5, so that the number of matrix-vector products is reduced from 4 to 2 per iteration. This is a significant reduction in computational complexity because the complexity of AMP-like algorithms is dominated by matrix-vector products.

In the (U)AMP algorithms, $g_x(\mathbf{q}, \boldsymbol{\tau}_q)$ is related to the prior of $\mathbf{x}$ and returns a column vector with the nth element $[g_x(\mathbf{q}, \boldsymbol{\tau}_q)]_n$ given by

$$[g_x(\mathbf{q}, \boldsymbol{\tau}_q)]_n = \frac{\int x_n p(x_n)\mathcal{N}(x_n; q_n, \tau_{q_n})dx_n}{\int p(x_n)\mathcal{N}(x_n; q_n, \tau_{q_n})dx_n}, \tag{17}$$

[3]By replacing $\mathbf{r}$ and $\boldsymbol{\Phi}$ with $\mathbf{y}$ and $\mathbf{A}$ in Algorithm 1 respectively, the original AMP algorithm is recovered.

where we note that $p(x_n)$ represents a general known prior for x_n. The function $g_x'(\mathbf{q}, \boldsymbol{\tau}_q)$ returns a column vector and the nth element is denoted by $[g_x'(\mathbf{q}, \boldsymbol{\tau}_q)]_n$, where the derivative is taken with respect to q_n.

III. Sparse Bayesian Learning Using UAMP

A. Problem Formulation and Approximate Inference

To enable the use of UAMP, we employ the unitary transformed model $\mathbf{r} = \boldsymbol{\Phi}\mathbf{x} + \boldsymbol{\omega}$ in (16). 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

$$p(\mathbf{x}, \boldsymbol{\gamma}, \beta|\mathbf{r}) \propto p(\mathbf{r}|\mathbf{x}, \beta)p(\mathbf{x}|\boldsymbol{\gamma})p(\boldsymbol{\gamma})p(\beta), \tag{18}$$

where $p(\mathbf{x}|\boldsymbol{\gamma})$ and $p(\boldsymbol{\gamma})$ are given by (2) and (3), respectively. We assume an improper prior $p(\beta) \propto 1/\beta$ for the noise precision [8]. According to the transformed model (16), $p(\mathbf{r}|\mathbf{x}, \beta) = \mathcal{N}(\mathbf{r}|\boldsymbol{\Phi}\mathbf{x}, \beta^{-1}\mathbf{I})$. 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 we resort to approximate inference techniques.

Variational inference is a machine learning method for approximate inference [29]–[31]. In variational inference, a tractable trial distribution function is chosen and optimized by minimizing the Kullback-Leibler (KL) divergence between the trial function and the true posterior distribution. 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 [30]–[32], which is termed SVMP. In this work, SVMP is adopted because it facilitates the incorporation of UAMP into SVMP. We will show how UAMP can be used to handle the most computational intensive part of message computations, enabling us to achieve low complexity and high robustness. With SVMP, we can find 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).

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

$$\begin{aligned} &p(\mathbf{x}, \mathbf{h}, \boldsymbol{\gamma}, \beta|\mathbf{r}) \\ &\quad\propto p(\mathbf{r}|\mathbf{h}, \beta)p(\mathbf{h}|\mathbf{x})p(\mathbf{x}|\boldsymbol{\gamma})p(\boldsymbol{\gamma}|\epsilon)p(\beta) \\ &\quad= \prod_{m=1}^{M}\mathcal{N}(r_m|h_m, \beta^{-1})\prod_{m=1}^{M}\delta(h_m - [\boldsymbol{\Phi}]_m\mathbf{x}) \\ &\qquad\prod_{n=1}^{N}\mathcal{N}(x_n|0, \gamma_n^{-1})\prod_{n=1}^{N}\mathrm{Ga}(\gamma_n|\epsilon, \eta)p(\beta). \end{aligned} \tag{19}$$

To facilitate the derivation of the message passing algorithm, a factor graph representation of the factorization in (19) is shown in Fig. 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 $\boldsymbol{\Phi}$.

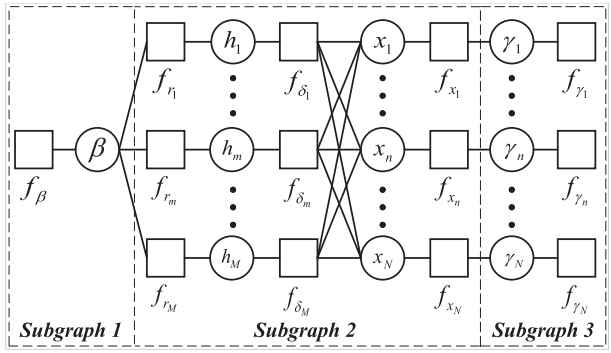

Fig. 1. Factor graph of (19) for deriving UAMP-SBL.

Following SVI, we define the following structured trial function

$$\tilde{q}(\mathbf{x},\mathbf{h},\boldsymbol{\gamma},\beta)=\tilde{q}(\beta)\tilde{q}(\mathbf{x},\mathbf{h})\tilde{q}(\boldsymbol{\gamma}). \tag{20}$$

In terms of SVMP, the use of the above trial function corresponds to a partition of the factor graph shown by the dotted boxes in Fig. 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

$$\mathcal{KL}\left(\tilde{q}(\beta)\tilde{q}(\mathbf{x},\mathbf{h})\tilde{q}(\boldsymbol{\gamma})||p(\mathbf{x},\mathbf{h},\boldsymbol{\gamma},\beta|\mathbf{r})\right), \tag{21}$$

is minimized, it is expected that

$$\tilde{q}(\mathbf{x},\mathbf{h})\approx p(\mathbf{x},\mathbf{h}|\mathbf{r}), \tag{22}$$

$$\tilde{q}(\boldsymbol{\gamma})\approx p(\boldsymbol{\gamma}|\mathbf{r}), \tag{23}$$

$$\tilde{q}(\beta)\approx p(\beta|\mathbf{r}). \tag{24}$$

Integrating out $\mathbf{h}$ in (22), which corresponds to running BP in Subgraph 2 (except the factor nodes connecting external variable nodes), we have $\tilde{q}(\mathbf{x})\approx p(\mathbf{x}|\mathbf{r})$. 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 Appendix A, and the algorithm is summarized in Algorithm 2.

Regarding the UAMP-SBL in Algorithm 2, we have the following remarks:

1. UAMPv2 is employed in Algorithm 2. Similarly, UAMPv1 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 versions 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 [8], and sometimes the value of the shape parameter ϵ is chosen empirically, e.g., $\epsilon=1$ [37]. In UAMP-SBL, we propose to tune the shape parameter automatically (as shown in Line 13) with the following empirical rule
$$\epsilon=\frac{1}{2}\sqrt{\log(\frac{1}{N}\sum\nolimits_n\hat{\gamma}_n)-\frac{1}{N}\sum_n\log\hat{\gamma}_n}, \tag{25}$$
i.e., ϵ is learned iteratively in the iterative process, with a small positive initial value. We note that, as the log function is concave, the parameter ϵ in (25) is guaranteed to be non-negative. In Section IV, we will show that the shape parameter ϵ in SBL functions as a selective amplifier for $\{\gamma_n\}$, and a proper ϵ plays a significant role in promoting sparsity, leading to considerable performance improvement.

Algorithm 2: UAMP-SBL.

Unitary transform: $\mathbf{r}=\mathbf{U}^H\mathbf{y}=\boldsymbol{\Phi}\mathbf{x}+\boldsymbol{\omega}$, where $\boldsymbol{\Phi}=\mathbf{U}^H\mathbf{A}=\boldsymbol{\Lambda}\mathbf{V}$, and $\mathbf{A}$ has SVD $\mathbf{A}=\mathbf{U}\boldsymbol{\Lambda}\mathbf{V}$.
Define vector $\boldsymbol{\lambda}=\boldsymbol{\Lambda}\boldsymbol{\Lambda}^H\mathbf{1}$.
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: $\mathbf{p}=\boldsymbol{\Phi}\hat{\mathbf{x}}^t-\boldsymbol{\tau}_p\cdot\mathbf{s}$
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: $\mathbf{q}=\hat{\mathbf{x}}^t+\tau_q\boldsymbol{\Phi}^H\mathbf{s}$
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: $\hat{\gamma}_n=(2\epsilon+1)/(|\hat{x}_n^{t+1}|^2+\tau_x^{t+1}),n=1,\ldots,N.$
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 $t<t_{\max})$

B. Empirical SE-Based Performance Prediction

In this section, leveraging empirical UAMP SE, we study how to predict the performance of UAMP-SBL. We treat UAMP-SBL as UAMP with a special denoiser, enabling the use of UAMP SE to predict the performance of UAMP-SBL. The denoiser in the UAMP-SBL corresponds to Lines 10-13 of the UAMP-SBL algorithm (Algorithm 2).

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

$$q_n^t=x_n+w_n^t, \tag{26}$$

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 τ^t of the noise is given by

$$\tau^t=\frac{N}{\mathbf{1}^H\left(\boldsymbol{\lambda}./(v_x^t\boldsymbol{\lambda}+\beta^{-1}\mathbf{1})\right)}, \tag{27}$$

which can be simply obtained with the variance-related variables in Lines 1, 5 and 7 of UAMPv2 in Algorithm 1. 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, we simulate the denoiser with the additive Gaussian noise model (26) by varying the variance of noise τ^t (or the SNR), so that we can get a "function" in terms of a table, with the variance of the noise as the input and the MSE as the output, i.e.,

$$v_x=\phi(\tau). \tag{28}$$

The function $\phi(\cdot)$ is independent of the measurement matrix $\mathbf{A}$. The performance of UAMP-SBL can be predicted using the

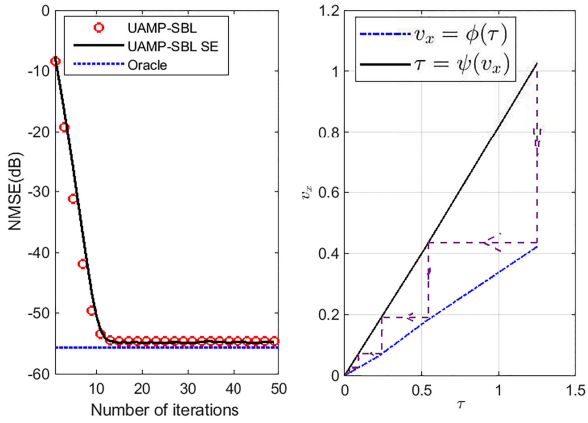

Fig. 2. SE and evolution trajectory of UAMP-SBL with a nonzero mean $\mathbf{A}$ ($N = 10240$, $M = 8192$, SNR $= 50$ dB, sparsity rate $\rho = 0.1$ and matrix mean $\mu = 1.0$).

following iteration with the initialization of v_x:

$$\begin{aligned} &Repeat \\ &\tau = \frac{N}{\mathbf{1}^H\left(\boldsymbol{\lambda}./(v_x\boldsymbol{\lambda} + \beta^{-1}\mathbf{1})\right)} \triangleq \psi(v_x) \\ &v_x = \phi(\tau) \\ &Until\ terminated \end{aligned} \tag{29}$$

We show the predicted performance, simulated performance in terms of normalized MSE (NMSE, which is defined in (41)) and the evolution trajectory of UAMP-SBL in Fig. 2 for a non-zero mean measurement matrix $\mathbf{A}$. It can be seen that the predicted performance matches well the simulated performance.

C. Computational Complexity

UAMP-SBL works well with a simple single loop iteration, which is in contrast to the double loop iterative algorithm GGAMP-SBL [26]. The complexity of UAMP-SBL (with UAMPv2) is dominated by two matrix-vector product operations in Line 2 and Line 9, i.e., $\mathcal{O}(MN)$ per iteration. The algorithm typically converges fast and delivers outstanding performance as shown in Section VI. 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 $\mathcal{O}(\min\{M^2N, MN^2\})$. Note that for the runtime comparison in Section VI, we do not assume off-line SVD computations, and the time consumed by SVD is counted for UAMP-SBL.

IV. Impact of the Shape Parameter ϵ in SBL

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

We consider the conventional SBL algorithm (η is set to be zero) [8]. In the case of identity matrix $\mathbf{A}$, it reduces to

$$\begin{aligned} &Repeat \\ &Z_{n,n} = \left(\beta + \gamma_n^t\right)^{-1} \\ &\hat{x}_n = \beta Z_{n,n} y_n \\ &\gamma_n^{t+1} = (2\epsilon + 1)/(|\hat{x}_n|^2 + Z_{n,n}) \\ &Until\ terminated \end{aligned} \tag{30}$$

Here note that in the above iteration we initialize $\gamma_n^{(0)} > 0$. The iteration in terms of γ_n has a closed form

$$\begin{aligned} \gamma_n^{t+1} &= \frac{2\epsilon + 1}{(\beta(\beta + \gamma_n^t)^{-1} y_n)^2 + (\beta + \gamma_n^t)^{-1}} \\ &= (2\epsilon + 1)\frac{(\beta + \gamma_n^t)^2}{(\beta y_n)^2 + \beta + \gamma_n^t} \\ &\triangleq g_\epsilon(\gamma_n^t). \end{aligned} \tag{31}$$

Next, we investigate the impact of ϵ on the convergence behavior and fixed points of the iteration (31) when $\epsilon = 0$ or ϵ takes a positive value.

For the iteration (31) with a small positive initial value $\gamma_n^{(0)}$, we have the following results.

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

$$\gamma_n' = \frac{\beta}{\beta y_n^2 - 1}; \tag{32}$$

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

Proof: See Appendix B. ■

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

$$\gamma_{n(a)} = \frac{2\beta(1 + 2\epsilon)}{\beta y_n^2 - 4\epsilon - 1 + \sqrt{\beta^2 y_n^4 - 8\epsilon\beta y_n^2 - 2\beta y_n^2 + 1}}; \tag{33}$$

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

Proof: See Appendix C. ■

Based on Proposition 1 and Theorem 1, we make the following remarks:

1. If $\beta y_n^2 \le 1$, 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 $\beta y_n^2 \le 1$, we have $(\beta y_n)^2 \le \beta$. Hence, from (31)
$$\begin{aligned} \gamma_n^{t+1} = g_\epsilon(\gamma_n^t) &\ge (2\epsilon + 1)\frac{(\beta + \gamma_n^t)^2}{2\beta + \gamma_n^t} \\ &= (2\epsilon + 1)\left(\gamma_n^t + \frac{\beta^2}{2\beta + \gamma_n^t}\right) \\ &> (2\epsilon + 1)\gamma_n^t. \end{aligned} \tag{34}$$
From (34), 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 (32) and (33), the ratio of the precisions obtained with $\epsilon > 0$ and $\epsilon = 0$ is given by
$$\frac{\gamma_{n(a)}}{\gamma_n'} = \frac{2(1 + 2\epsilon)}{1 - \frac{4\epsilon}{\beta y_n^2 - 1} + \sqrt{\left(1 - \frac{4\epsilon}{\beta y_n^2 - 1}\right)^2 - \frac{8\epsilon(1 + 2\epsilon)}{(\beta y_n^2 - 1)^2}}}. \tag{35}$$

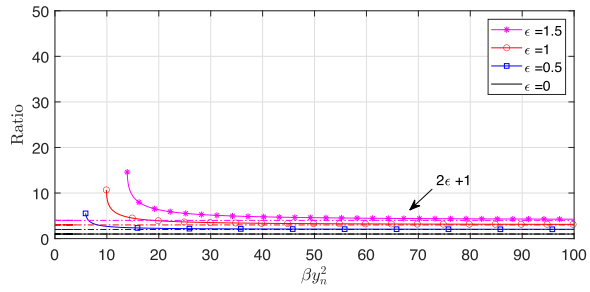

Fig. 3. Ratio of precisions with different ϵ.

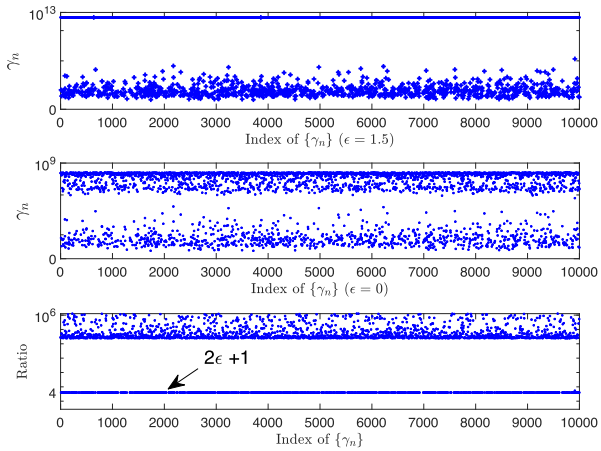

Fig. 4. Precisions and their ratios ($\mathbf{A}$ is an identity matrix).

The ratio is a function of βy_n^2, and

$$\gamma_{n(a)}/\gamma'_n \approx 1 + 2\epsilon, \tag{36}$$

if βy_n^2 is relatively large.

The ratios of the precisions versus βy_n^2 are shown in Fig. 3, 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. We see that a positive ϵ tends to a sparser solution, and a proper value of ϵ leads to much better recovery performance, compared to $\epsilon = 0$.

The precisions of the elements of the sparse vector obtained by the SBL algorithm with $\epsilon = 1.5$ and $\epsilon = 0$ are shown in Fig. 4, where $\mathbf{A}$ is an identity matrix with size 10000×10000, the sparsity rate of the signal is 0.1, and SNR $= 50$ dB. 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

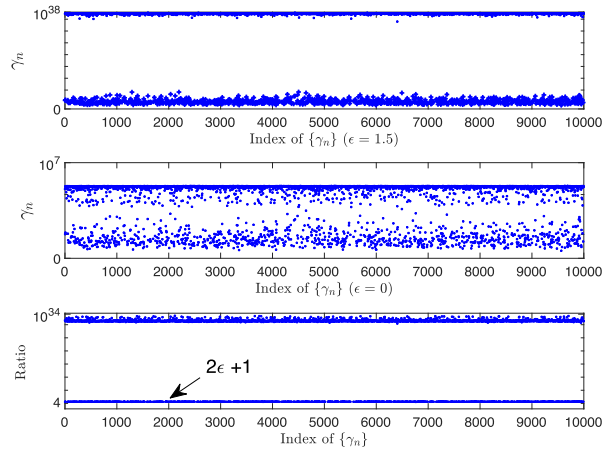

Fig. 5. Precisions and their ratios ($\mathbf{A}$ is i.i.d Gaussian).

$\mathbf{A}$ as demonstrated numerically in Fig. 5, 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 the identity matrix. (Similar observations are observed for other matrices). We see 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, we can find a proper value for ϵ through trial and error. However, this is inconvenient, and the sparsity rate of the signal may not be available. We found the empirical (25) to determine the value of ϵ. Next, we examine its effectiveness with the SBL algorithm. Plugging the shape parameter update rule (25) to the conventional SBL algorithm leads to the following iterative algorithm (assuming the noise precision β is known):

Repeat

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

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

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

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

Until terminated

To demonstrate the effectiveness of the shape parameter update rule (25), we compare the performance of the conventional SBL algorithm with and without shape parameter update. The results are shown in Fig. 6, where the SNR is 50 dB, the size of the measurement matrix is 800×1000, and the sparsity rate $\rho = 0.1$. 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 VI for their generations). It can be seen that there is a clear gap between the performance of conventional SBL and the bounds, and with shape parameter updated with our rule, the SBL algorithm attains the bound. It is worth mentioning that the empirical finding in [26], i.e., replacing the noise variance β^{-1} with $3\beta^{-1}$ can lead to better performance of GGAMP-SBL [26]. We use this for the conventional SBL algorithm and the performance is also included in Fig. 6. We see that it also leads to substantial

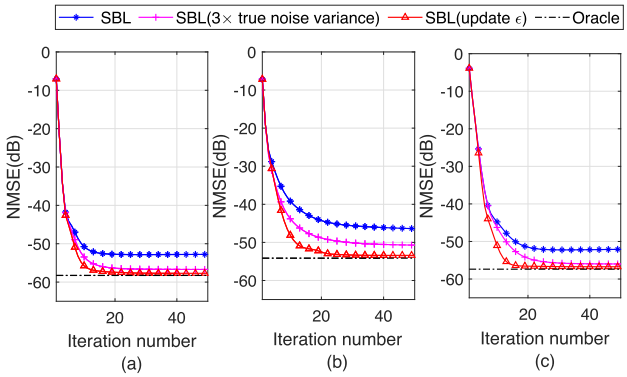

Fig. 6. Performance of the conventional SBL. (a) Gaussian matrix; (b) correlated matrix with $c = 0.3$; (c) low-rank matrix with $R/N = 0.6$.

performance improvement, but its performance is inferior to that of SBL with ϵ updated using (25). 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.

V. Extension to MMV

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

A. UAMP-SBL for MMV

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

$$\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{W}, \tag{37}$$

where we assume 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 $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}$, a unitary transformation with $\mathbf{U}^H$ to (37) can be performed, i.e.,

$$\mathbf{R} = \mathbf{\Phi}\mathbf{X} + \mathbf{\Omega}, \tag{38}$$

where $\mathbf{R} = \mathbf{U}^H\mathbf{Y} = [\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, \ldots, \mathbf{r}^{(L)}]$, $\mathbf{\Phi} = \mathbf{U}^H\mathbf{A} = \mathbf{\Lambda}\mathbf{V}$ and $\mathbf{\Omega} = \mathbf{U}^H\mathbf{W}$ is still white and Gaussian with mean zero and precision β. Define $\mathbf{h}^{(l)} = \mathbf{\Phi}\mathbf{x}^{(l)}$ and $\mathbf{H} = [\mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(L)}]$. Then we have the following joint distribution

$$\begin{aligned} &p(\mathbf{X}, \mathbf{H}, \boldsymbol{\gamma}, \beta | \mathbf{R}) \\ &\propto \prod_{l=1}^{L} p(\mathbf{r}^{(l)}|\mathbf{h}^{(l)}, \beta) p(\mathbf{h}^{(l)}|\mathbf{x}^{(l)}) p(\mathbf{x}^{(l)}|\boldsymbol{\gamma}) p(\boldsymbol{\gamma}) p(\beta) \\ &= \prod_{l=1}^{L}\prod_{m=1}^{M} \mathcal{N}(r_m^{(l)}|h_m^{(l)}, \beta^{-1}) \delta(h_m^{(l)} - [\mathbf{\Phi}]_m \mathbf{x}^{(l)}) \\ &\quad \times \prod_{l=1}^{L}\prod_{n=1}^{N} \mathcal{N}(x_n^{(l)}|0, \gamma_n^{-1}) \prod_{n=1}^{N} \mathrm{Ga}(\gamma_n|\epsilon, \eta) p(\beta). \end{aligned} \tag{39}$$

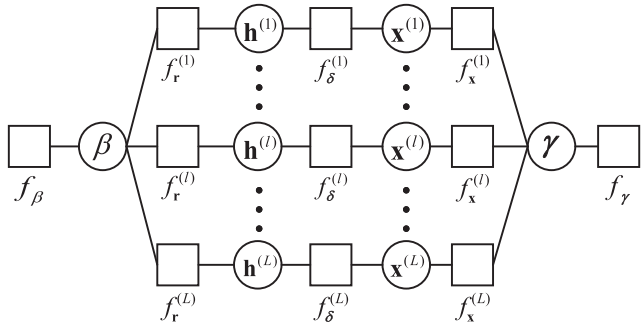

Fig. 7. Factor graph representation of (39).

Algorithm 3: UAMP-SBL for MMV.

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

$\forall l$; ${\boldsymbol{\tau}_p}^l = \tau_x^{l(t)}\boldsymbol{\lambda}$
$\forall l$; $\mathbf{p}^l = \mathbf{\Phi}\hat{\mathbf{x}}^{l(t)} - \boldsymbol{\tau}_p^l \cdot \mathbf{s}^l$
$\forall l$; $\mathbf{v}_h^l = \boldsymbol{\tau}_p^l./(\mathbf{1} + \hat{\beta}\boldsymbol{\tau}_p^l)$
$\forall l$; $\hat{\mathbf{h}}^l = (\hat{\beta}\boldsymbol{\tau}_p^l \cdot \mathbf{r}^l + \mathbf{p}^l)./(\mathbf{1} + \hat{\beta}\boldsymbol{\tau}_p^l)$
$\hat{\beta} = LM/(\sum_l(||\mathbf{r}^l - \hat{\mathbf{h}}^l||^2 + \mathbf{1}^H\mathbf{v}_h^l))$;
$\forall l$; $\boldsymbol{\tau}_s^l = \mathbf{1}./(\boldsymbol{\tau}_p^l + \hat{\beta}^{-1}\mathbf{1})$
$\forall l$; $\mathbf{s}^l = \boldsymbol{\tau}_s^l \cdot (\mathbf{r}^l - \mathbf{p}^l)$
$\forall l$; $1/\tau_q^l = (1/N){\lambda}^H{\boldsymbol{\tau}_s}^l$
$\forall l$; $\mathbf{q}^l = \hat{\mathbf{x}}^{l(t)} + \tau_q^l(\mathbf{\Phi}^H\mathbf{s}^l)$
$\forall l$; $\tau_x^{l(t+1)} = (\tau_q^l/N)\mathbf{1}^H(\mathbf{1}./(\mathbf{1} + \tau_q^l\hat{\boldsymbol{\gamma}}))$
$\forall l$; $\hat{\mathbf{x}}^{l(t+1)} = \mathbf{q}^l./(\mathbf{1} + \tau_q^l\hat{\boldsymbol{\gamma}})$
$\hat{\gamma}_n = \frac{2\epsilon'+1}{(1/L)\sum_{l=1}^{L}(|\hat{x}_n^{l(t+1)}|^2 + \tau_x^{l(t+1)})}, n = 1, \ldots, N.$
$\epsilon' = \frac{1}{2}\sqrt{\log(\frac{1}{N}\sum_n \hat{\gamma}_n) - \frac{1}{N}\sum_n \log\hat{\gamma}_n}$
$t = t + 1$

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 $t < t_{\max}$)

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 (39) is shown in Fig. 7[4], 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 Appendix D. The UAMP-SBL for MMV is summarized in Algorithm 3, where UAMPv2 is employed. The complexity of the algorithm is $\mathcal{O}(MNL)$ per iteration.

[4]The vector variable node $\boldsymbol{\gamma}$ is used in the factor graph to make it neat. We note that each entry $x_n^{(l)}$ of $\boldsymbol{x}^{(l)}$ is connected to γ_n through the function node between them.

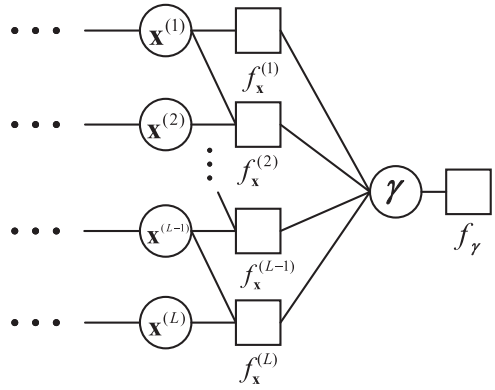

Fig. 8. The additional factor graph for deriving UAMP-TSBL.

B. UAMP-TSBL

With the assumption of a common sparsity profile shared by all sparse vectors, we further consider 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 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 [26], we use an AR(1) process [38] to model the correlation between $x_n^{(l)}$ and $x_n^{(l-1)}$, i.e.,

$$\begin{aligned} x_n^{(l)} &= \alpha x_n^{(l-1)} + \sqrt{1-\alpha^2}\vartheta_n^{(l)} \\ p(x_n^{(l)}|x_n^{(l-1)}) &= \mathcal{N}(x_n^{(l)}|\alpha x_n^{(l-1)}, (1-\alpha^2)\gamma_n^{-1}), l>1 \\ p(x_n^{(1)}) &= \mathcal{N}(x_n^{(1)}|0,\gamma_n^{-1}), \end{aligned} \tag{40}$$

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. We redefine the factors $\{f_{x_n}^{(l)}(x_n^{(l)},\gamma_n)\}$, 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 Fig. 8. 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 Appendix E, and the algorithm is summarized in Algorithm 4. UAMP-TSBL is an extension of the UAMP-SBL algorithm for MMV (Algorithm 3). The complexity of UAMP-TSBL is also dominated by matrix-vector multiplications, and is $\mathcal{O}(MNL)$ per iteration.

VI. Numerical Results

In this section, we compare the proposed UAMP-(T)SBL algorithms with the conventional SBL and state-of-the-art AMP-based SBL algorithms. We evaluate the performance of various algorithms using normalized MSE, defined as

$$\text{NMSE} \triangleq \frac{1}{K}\sum_{k=1}^{K} ||\hat{\mathbf{x}}_k - \mathbf{x}_k||^2/||\mathbf{x}_k||^2, \tag{41}$$

Algorithm 4: UAMP-TSBL.

Unitary transform: $\mathbf{R} = \mathbf{U}^H\mathbf{Y} = \mathbf{\Phi X} + \mathbf{W}$, where $\mathbf{\Phi} = \mathbf{U}^H\mathbf{A} = \mathbf{\Lambda V}$, and $\mathbf{A}$ has SVD $\mathbf{A} = \mathbf{U\Lambda V}$.
Define vector $\boldsymbol{\lambda} = \mathbf{\Lambda\Lambda}^H\mathbf{1}$.
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 $l = 2,\ldots,L$
  $\boldsymbol{\xi}^l = \alpha(\frac{\mathbf{q}^{l-1}}{\boldsymbol{\tau}_q^{l-1}} + \frac{\boldsymbol{\xi}^{l-1}}{\boldsymbol{\psi}^{l-1}}) \cdot (\frac{\boldsymbol{\tau}_q^{l-1}\cdot\boldsymbol{\psi}^{l-1}}{\boldsymbol{\tau}_q^{l-1}+\boldsymbol{\psi}^{l-1}})$
  $\boldsymbol{\psi}^l = \alpha^2(\frac{\boldsymbol{\tau}_q^{l-1}\cdot\boldsymbol{\psi}^{l-1}}{\boldsymbol{\tau}_q^{l-1}+\boldsymbol{\psi}^{l-1}}) + (1-\alpha^2)/\hat{\boldsymbol{\gamma}}^{(t)}$
end
for $l = 1,\ldots,L$
  ${\boldsymbol{\tau}_p}^l = \tau_x^{l(t)}\boldsymbol{\lambda}$
  $\mathbf{p}^l = \mathbf{\Phi}\hat{\mathbf{x}}^{l(t)} - \boldsymbol{\tau}_p^l \cdot \mathbf{s}^{l(t-1)}$
  $\mathbf{v}_h^l = \boldsymbol{\tau}_p^l./(\mathbf{1} + \hat{\beta}\boldsymbol{\tau}_p^l)$
  $\hat{\mathbf{h}}^l = (\hat{\beta}\boldsymbol{\tau}_p^l \cdot \mathbf{r}^l + \mathbf{p}^l)./(\mathbf{1} + \hat{\beta}\boldsymbol{\tau}_p^l)$
end
$\hat{\beta} = LM/(\sum_l(||\mathbf{r}^l - \hat{\mathbf{h}}^l||^2 + \mathbf{1}^H\mathbf{v}_h^l))$
for $l = 1,\ldots,L$
  $\boldsymbol{\tau}_s^l = \mathbf{1}./(\boldsymbol{\tau}_p^l + \hat{\beta}^{-1}\mathbf{1})$
  $\mathbf{s}^{l(t)} = \boldsymbol{\tau}_s^l \cdot (\mathbf{r}^l - \mathbf{p}^l)$
  $1/\tau_q^l = (1/N)\lambda^H{\boldsymbol{\tau_s}}^l$
  $\mathbf{q}^l = \hat{\mathbf{x}}^{l(t)} + \tau_q^l(\mathbf{\Phi}^H\mathbf{s}^{l(t)})$
  $\tau_x^{l(t+1)} = (1/N)\mathbf{1}^H(\mathbf{1}./(\mathbf{1}./\boldsymbol{\tau}_q^l + \mathbf{1}./\boldsymbol{\phi}^l + \mathbf{1}./\boldsymbol{\psi}^l))$
  $\hat{\mathbf{x}}^{l(t+1)} = \tau_x^{l(t+1)}(\mathbf{q}^l./\boldsymbol{\tau}_q^l + \boldsymbol{\theta}^l./\boldsymbol{\phi}^l + \boldsymbol{\xi}^l./\boldsymbol{\psi}^l)$
end
$\boldsymbol{\theta}^{L-1} = \frac{1}{\alpha}\mathbf{q}^L$
$\boldsymbol{\phi}^{L-1} = \frac{1}{\alpha^2}(\boldsymbol{\tau}_q^L + (1-\alpha^2)/\hat{\boldsymbol{\gamma}}^{(t)})$
for $l = L-2,\ldots,1$
  $\boldsymbol{\theta}^l = \frac{1}{\alpha}(\frac{\mathbf{q}^{l+1}}{\boldsymbol{\tau}_q^{l+1}} + \frac{\boldsymbol{\theta}^{l+1}}{\boldsymbol{\phi}^{l+1}}) \cdot (\frac{\boldsymbol{\tau}_q^{l+1}\boldsymbol{\phi}^{l+1}}{\boldsymbol{\tau}_q^{l+1}+\boldsymbol{\phi}^{l+1}})$
  $\boldsymbol{\phi}^l = \frac{1}{\alpha^2}(\frac{\boldsymbol{\tau}_q^{l+1}\boldsymbol{\phi}^{l+1}}{\boldsymbol{\tau}_q^{l+1}+\boldsymbol{\phi}^{l+1}} + (1-\alpha^2)/\hat{\boldsymbol{\gamma}}^{(t)})$
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 $t < t_{\max}$

$$\text{NMSE} \triangleq \frac{1}{KL}\sum_{k=1}^{J}\sum_{l=1}^{L} ||\hat{\mathbf{x}}_k^{(l)} - \mathbf{x}_k^{(l)}||^2/||\mathbf{x}_k^{(l)}||^2 \tag{42}$$

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

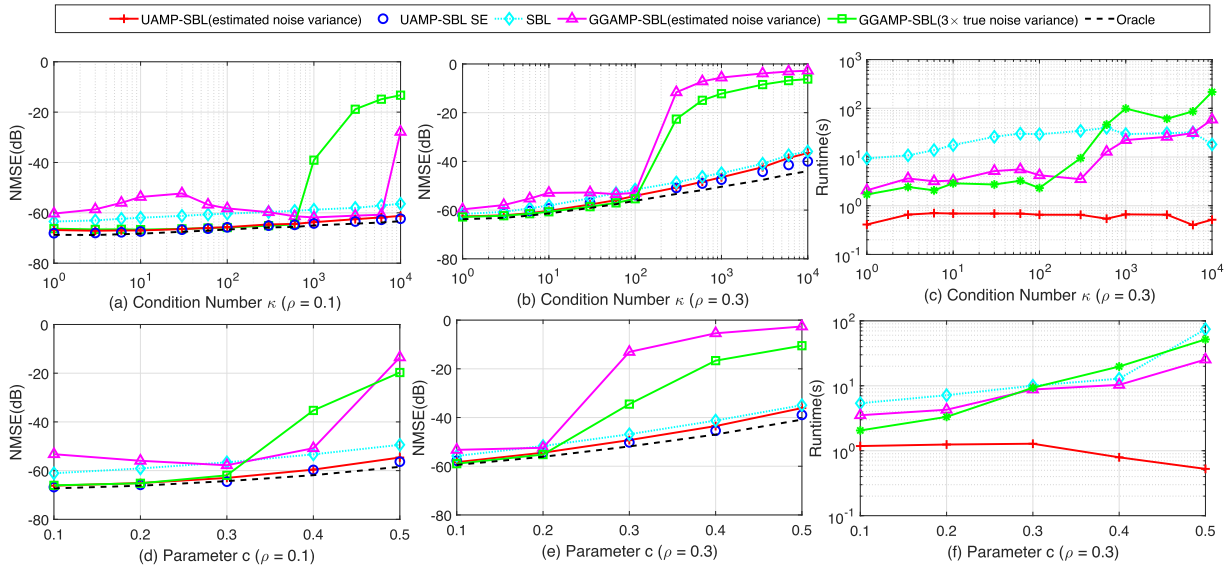

Fig. 9. Performance and runtime comparisons (ill-conditioned matrices and correlated matrices).

converge, as in [26], we measure the runtime of the algorithms to indicate their relative computational complexity. 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, we use the following measurement matrices:

1) Ill-conditioned Matrix: Matrix $\mathbf{A}$ is constructed based on the SVD $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}$ where $\mathbf{\Lambda}$ is a singular value matrix with $\Lambda_{i,i}/\Lambda_{i+1,i+1} = \kappa^{1/(M-1)}$ for $i = 1, 2, \ldots, M-1$ (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 the correlation of matrix $\mathbf{A}$.
3) Non-zero Mean Matrix: The elements of matrix $\mathbf{A}$ 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 $\mathbf{A} = \mathbf{B}\mathbf{C}$, where the size of $\mathbf{B}$ and $\mathbf{C}$ are $M \times R$ and $R \times N$, respectively, and $R < M$. Both $\mathbf{B}$ and $\mathbf{C}$ 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 $\mathbf{A}$ from the i.i.d. Gaussian matrix.

A. Numerical Results for SMV

In this section, we compare UAMP-SBL against the conventional SBL [8] and the state-of-the-art AMP based SBL algorithm GGAMP-SBL [26] with estimated noise variance and 3 times true noise variance. The vector $\mathbf{x}$ is drawn from a Bernoulli-Gaussian distribution with a non-zero probability ρ. The SNR is defined as $\text{SNR} \triangleq E\|\mathbf{A}\mathbf{x}\|^2/E\|\mathbf{w}\|^2$. As a performance benchmark, the support-oracle MMSE bound [26] is also included. We set $M = 800$, $N = 1000$ and the SNR is 60 dB unless it is specified. For UAMP-SBL we set the maximum iteration number $t_{\max} = 300$ (note that there is no inner iteration in UAMP-SBL). GGAMP-SBL is a double loop algorithm, the maximum numbers of E-step and outer iteration are set to be 50 and 1000 respectively. A small damping factor 0.2 is used for GGAMP-SBL 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 compromised robustness and performance.

In Figs. 9(a) and (b), the performance of various algorithms in terms of NMSE versus the condition number is shown for the $\rho = 0.1$ and 0.3, respectively. It can be seen from Fig. 9(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 ρ in Fig. 9(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. Figs. 9(d) and (e) show the performance of various algorithms versus a range of correlation parameter c from 0.1 to 0.5. 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 parameter c is relatively large. The gap between UAMP-SBL and GGAMP-SBL becomes more notable with a higher sparsity rate. In addition, the simulation performance of UAMP-SBL matches well with the predicted performance. The average runtime of various algorithms is shown in Figs. 9(c) and (f) for ill-conditioned matrices and correlated matrices, respectively, where the sparsity rate $\rho = 0.3$. 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, we set its damping factor to be a relatively small value, i.e., 0.2 to achieve better performance and robustness. If the damping factor is increased, GGAMP-SBL could be faster but at the cost of offsetting its performance and robustness.

In Fig. 10, we examine 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 good performance,

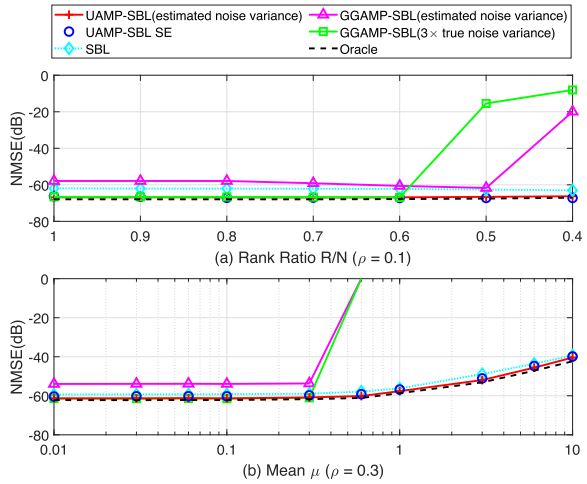

Fig. 10. Performance comparison: (a) low rank matrices; (b) non-zero mean matrices.

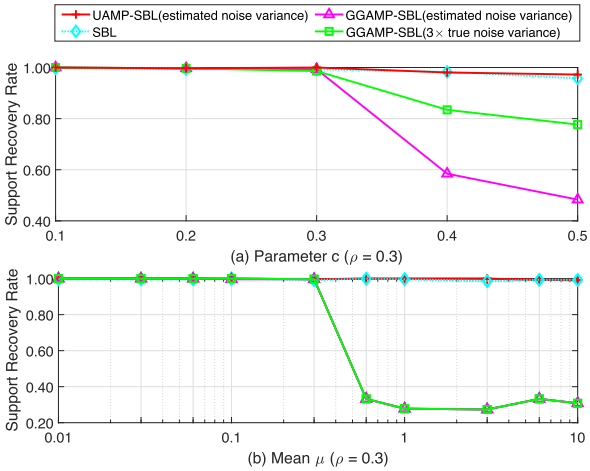

Fig. 11. Support recovery rate comparison: (a) correlated matrices; (b) non-zero mean matrices.

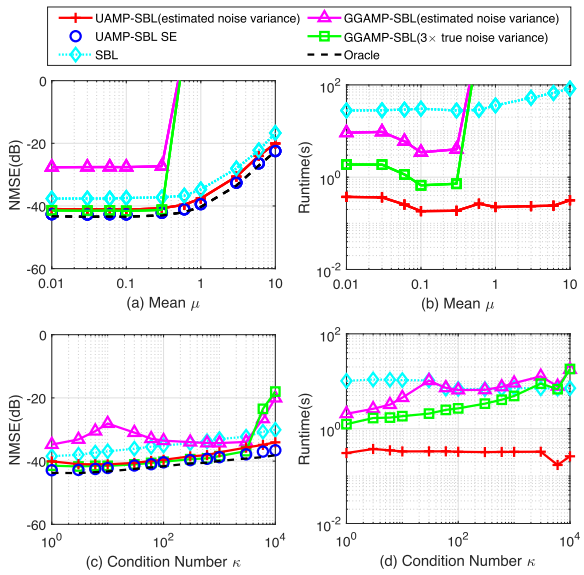

Fig. 12. Performance and runtime comparisons of various algorithms where SNR = 35 dB.

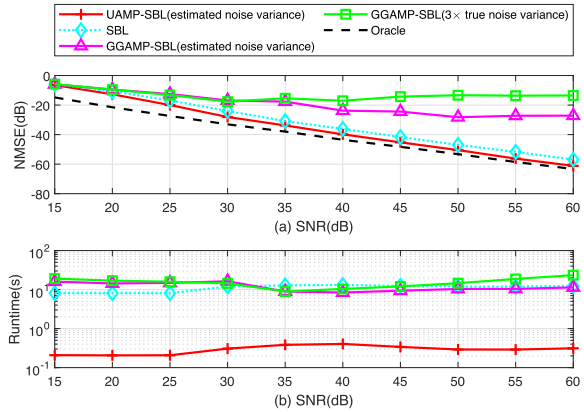

Fig. 13. Performance and runtime comparisons of the algorithms versus SNR for ill-conditioned matrices.

which closely matches the support-oracle bound, and is slightly better than that of SBL. We can also see that GGAMP-SBL diverges when the mean μ is relatively large. The simulated performance of UAMP-SBL matches well with the predicted performance.

In Fig. 11, we evaluate the support recovery rate of the algorithms versus correlation parameter c for correlated matrices in (a) and mean value μ for non-zero mean matrices in (b), where the sparse rate $\rho = 0.3$. The support recovery rate is defined as the percentage of successful trials in the total trials [39]. 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, we can see that UAMP-SBL and SBL deliver similar performance and they can significantly outperform GGAMP-SBL when c or μ is relatively large.

We also compare the performance of various algorithms at SNR $=$ 35 dB, and the NMSE performance and runtime of the algorithms are shown in Fig. 12, where (a) and (b) are for non-zero mean matrices, and (c) and (d) are for ill-conditioned matrices. The sparsity rate $\rho = 0.1$. Again, we can see 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. We show the performance of the algorithms versus SNR in Fig. 13, where the matrices are highly ill-conditioned with a condition number $\kappa = 10^4$. We can see that GGAMP-SBL does not work well, and 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, we test the impact of the unitary transformation on the GGAMP-SBL algorithm, where we first perform the unitary transformation to the original model and then carry out GGAMP-SBL. We call this algorithm UT-GGAMP-SBL, and compare it with UAMP-SBL in the case of correlated matrices.

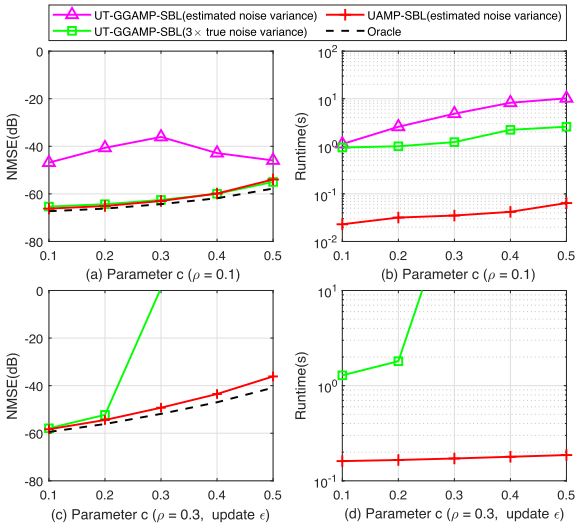

Fig. 14. Performance and runtime comparisons of UAMP-SBL and UT-GGAMP-SBL.

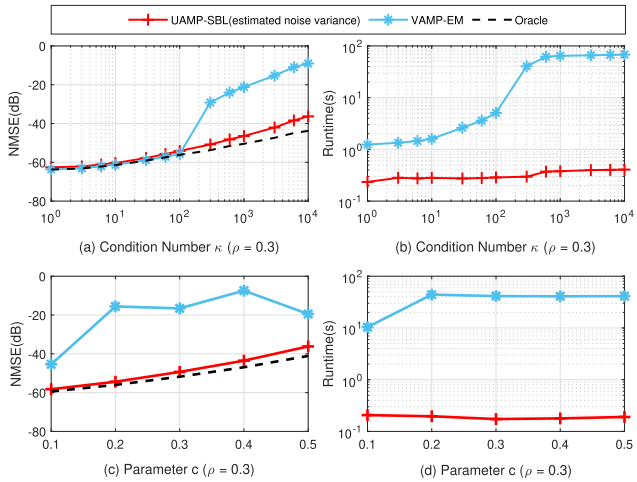

Fig. 15. Performance and runtime comparisons of UAMP-SBL and VAMP-EM.

The performance and the corresponding runtime are shown in Fig. 14, where the hyper-parameter ϵ of UT-GGAMP-SBL is not updated in (a) and (b) while it is updated in (c) and (d). It can be seen that, thanks to the unitary transformation, the stability of GGAMP-SBL is significantly improved as expected. Fig. 14(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. Fig. 14(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 in the figure. UAMP-SBL outperforms UT-GGAMP-SBL with 3 times true noise variance when c is relatively large. In addition, UAMP-SBL is faster.

We then compare UAMP-SBL with VAMP-EM in [40]. In VAMP-EM, Bernoulli-Gaussian priors are employed and the parameters of the priors are learned using EM. The NMSE performance and runtime are shown in Fig. 15, 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 delivers better performance with less runtime.

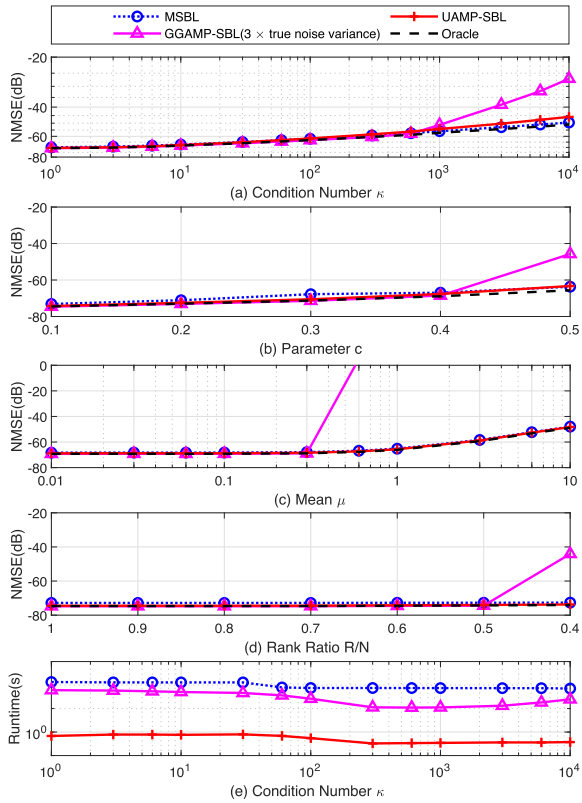

Fig. 16. Performance comparison of various algorithms in the case of MMV.

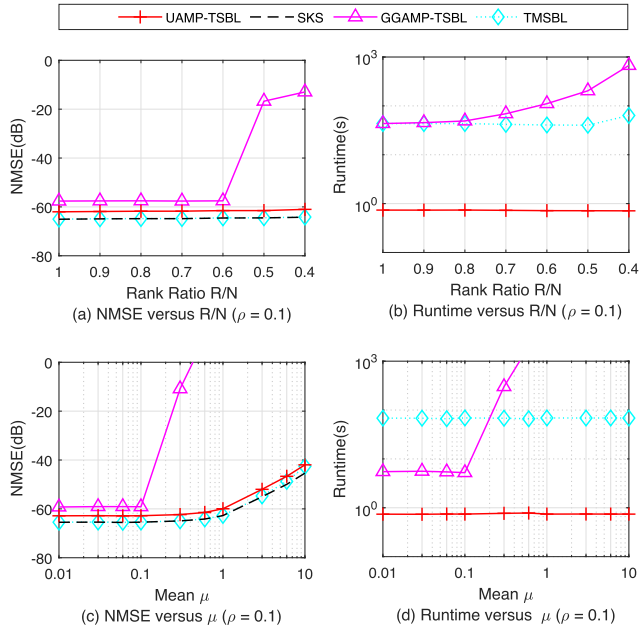

Fig. 17. Performance comparison of various algorithms in the case of MMV with temporal correlation.

B. Numerical Results for MMV

The elements of the sparse vectors $\{\mathbf{x}^{(l)}, l = 1 : L\}$ are drawn from a Bernoulli-Gaussian distribution, and the vectors share a common support. The number of measurement vectors is 5. The performance of the algorithms with ill-conditioned, correlated, non-zero mean and low-rank measurement matrices is shown in Figs. 16(a)-(d), respectively. In this figure, we also include the performance of the direct extension of the conventional SBL algorithm to the MMV model (MSBL) [41] and support-oracle bound. 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 $3\times$ true noise variance) and UAMP-SBL deliver similar performance, and both of them can approach the bound closely. MSBL is 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, we show the average runtime of different algorithms in the case of ill-conditioned matrices in Fig. 16(e), where UAMP-SBL converges significantly faster than GGAMP-SBL and MSBL.

Furthermore, we present a numerical study to illustrate the performance of UAMP-SBL when incorporating the temporal correlation. Besides the temporally correlated SBL (TMSBL) [39] and GGAMP-SBL, we also compare the recovery performance with a lower bound: the achievable NMSE by a support-aware Kalman smoother (SKS) [42] with the knowledge of the support of the sparse vectors and the true values of β, α and $\boldsymbol{\gamma}$. The SKS is implemented in a more efficient way by incorporating UAMP. As examples, we use low rank and non-zero mean measurement matrices to test their performance. The sparsity rate $\rho = 0.1$, SNR $= 50$ dB and the temporal correlation coefficient $\alpha = 0.8$. It can be seen from Fig. 17 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.

VII. Conclusion

In this paper, leveraging UAMP, we proposed UAMP-SBL 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. We demonstrated that, compared to the state-of-the-art AMP based SBL algorithm, UAMP-SBL achieves much better performance in terms of robustness, speed and recovery accuracy. Future work includes rigorous analyses of the state evolution of UAMP-SBL and the update mechanism of the shape parameter.

Appendix A
Derivation of UAMP-SBL With SVMP

We detail the forward and backward message passing in each subgraph of the factor graph in Fig. 1 according to the principle of SVMP [29], [30], [32]. 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.

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 (53).

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 $\boldsymbol{\Phi}$, 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.,

$$\mathcal{M}_{h_m\to f_{r_m}}(h_m) = \mathcal{M}_{f_{\delta_m}\to h_m}(h_m) = \mathcal{N}(h_m|p_m, \tau_{p_m}), \tag{43}$$

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 1), which are also Line 2 and Line 1 of the UAMP-SBL algorithm (Algorithm 2).

Following SVMP [32], the message $\mathcal{M}_{f_{r_m}\to\beta}(\beta)$ from factor node f_{r_m} to variable node β can be expressed as

$$\mathcal{M}_{f_{r_m}\to\beta}(\beta) \propto \exp\left\{\left\langle \log f_{r_m}(r_m|h_m, \beta^{-1})\right\rangle_{b(h_m)}\right\}, \tag{44}$$

where the belief of h_m is given as

$$b(h_m) \propto \mathcal{M}_{h_m\to f_{r_m}}(h_m)\mathcal{M}_{f_{r_m}\to h_m}(h_m). \tag{45}$$

In (51), we 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 given in (54). 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

$$v_{h_m} = (1/\tau_{p_m} + \hat{\beta})^{-1} \tag{46}$$

$$\hat{h}_m = v_{h_m}(\hat{\beta} r_m + p_m/\tau_{p_m}). \tag{47}$$

Note that $\boldsymbol{\tau}_p$ may contain zero elements. To avoid numerical problems in (46) and (47), they can be rewritten (in vector form) as

$$\mathbf{v}_h = \boldsymbol{\tau}_p./(\mathbf{1} + \hat{\beta}\boldsymbol{\tau}_p) \tag{48}$$

$$\hat{\mathbf{h}} = (\hat{\beta}\boldsymbol{\tau}_p \cdot \boldsymbol{r} + \boldsymbol{p})./(\mathbf{1} + \hat{\beta}\boldsymbol{\tau}_p), \tag{49}$$

which are Lines 3 and 4 of the UAMP-SBL algorithm.

From (44) and the Gaussianity of $b(h_m)$, the message $\mathcal{M}_{f_{r_m}\to\beta}(\beta)$ is

$$\mathcal{M}_{f_{r_m}\to\beta}(\beta) \propto \sqrt{\beta}\exp\left\{-\frac{\beta}{2}(|r_m - \hat{h}_m|^2 + v_{h_m})\right\}. \tag{50}$$

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

$$\begin{aligned}\mathcal{M}_{f_{r_m}\to h_m}(h_m) &\propto \exp\left\{\left\langle \log f_{r_m}(r_m|h_m, \beta^{-1})\right\rangle_{b(\beta)}\right\}\\ &\propto \mathcal{N}(h_m|r_m, \hat{\beta}^{-1}),\end{aligned} \tag{51}$$

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

$$b(\beta) = \mathcal{M}_{\beta\to f_{r_m}}(\beta)\mathcal{M}_{f_{r_m}\to\beta}(\beta)$$

$$= f_\beta(\beta)\prod_m \mathcal{M}_{f_{r_m}\to\beta}(\beta)$$

$$\propto \beta^{\frac{M}{2}-1}\exp\left\{-\frac{\beta}{2}\sum_m\left(|r_m-\hat{h}_m|^2+v_{h_m}\right)\right\}, \quad (52)$$

and

$$\mathcal{M}_{\beta\to f_{r_m}}(\beta) = f_\beta(\beta)\prod_{m'\neq m}\mathcal{M}_{f_{r_{m'}}\to\beta}(\beta). \quad (53)$$

It is noted that $b(\beta)$ follows a Gamma distribution with rate parameter $\frac{1}{2}\sum_m(|r_m-\hat{h}_m|^2+v_{h_m})$ and shape parameter $M/2$, so $\hat{\beta}=\langle\beta\rangle_{b(\beta)}$ can be computed as

$$\hat{\beta} = M/\sum_m\left(|r_m-\hat{h}_m|^2+v_{h_m}\right), \quad (54)$$

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

From (51), the Gaussian form of the message $\mathcal{M}_{f_{r_m}\to h_m}(h_m)$ suggests the following model

$$r_m = h_m + w_m, \quad m=1,\ldots,M, \quad (55)$$

where w_m is a Gaussian noise with mean 0 and variance $\hat{\beta}^{-1}$. This fits the forward recursion of the UAMP algorithm with known noise variance. Therefore, Lines 3 - 6 of the UAMP algorithm (Algorithm 1) 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.

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

$$\mathcal{M}_{f_{x_n}\to\gamma_n}(\gamma_n)\propto\exp\left\{\left\langle\log f_{x_n}(x_n|0,\gamma_n^{-1})\right\rangle_{b(x_n)}\right\}, \quad (56)$$

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 (63), 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

$$\tau_{x_n} = (1/\tau_q+\hat{\gamma}_n)^{-1} \quad (57)$$

$$\hat{x}_n = q_n/(1+\tau_q\hat{\gamma}_n). \quad (58)$$

Performing the average operations to $\{\tau_{x_n}\}$ in (57) and arranging (58) in a vector form lead to Lines 10 and 11 of the UAMP-SBL algorithm. According to the above,

$$\mathcal{M}_{f_{x_n}\to\gamma_n}(\gamma_n)\propto\sqrt{\gamma_n}\exp\left\{-\frac{\gamma_n}{2}(|\hat{x}_n|^2+\tau_x)\right\}, \quad (59)$$

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

$$\mathcal{M}_{f_{\gamma_n}\to\gamma_n}(\gamma_n)\propto{\gamma_n}^{\epsilon-1}\exp\{-\eta\gamma_n\}. \quad (60)$$

According to SVMP, the message

$$\mathcal{M}_{f_{x_n}\to x_n}(x_n)\propto\exp\left\{\left\langle\log f_x(x_n|0,\gamma_n^{-1})\right\rangle_{b(\gamma_n)}\right\}, \quad (61)$$

where the belief of γ_n

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

$$\propto{\gamma_n}^{\epsilon-\frac{1}{2}}\exp\left\{-\frac{\gamma_n}{2}(|\hat{x}_n|^2+\tau_x+2\eta)\right\}. \quad (62)$$

Hence, the message

$$\mathcal{M}_{f_{x_n}\to x_n}(x_n)\propto\mathcal{N}(x_n|0,\hat{\gamma}_n^{-1}), \quad (63)$$

where

$$\hat{\gamma}_n = \langle\gamma_n\rangle_{b(\gamma_n)} = \frac{2\epsilon+1}{2\eta+|\hat{x}_n|^2+\tau_x}. \quad (64)$$

Here we set $\eta=0$, and $\hat{\gamma}_n$ is reduced to $\frac{(2\epsilon+1)}{|\hat{x}_n|^2+\tau_x}$, which leads to Line 12 of the UAMP-SBL algorithm.

We propose to tune the parameter automatically with the empirical update rule for ϵ shown in Line 13 of the UAMP-SBL algorithm. The iteration is terminated when either the difference between two consecutive estimates of $\mathbf{x}$ is smaller than a threshold or the iteration number reaches the pre-set maximum value $t_{\max}$.

Appendix B
Proof of Proposition 1

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

$$\gamma_n^{t+1} = g_{\epsilon_0}(\gamma_n^t) = \frac{(\beta+\gamma_n^t)^2}{(\beta y_n)^2+\beta+\gamma_n^t}. \quad (65)$$

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

$$f(\gamma_n) = g_{\epsilon_0}(\gamma_n)-\gamma_n = 0, \quad (66)$$

which leads to the unique root

$$\gamma_n' = \beta/(\beta y_n^2-1). \quad (67)$$

If $\beta y_n^2>1$, the root $\gamma_n'=\beta/(\beta y_n^2-1)>0$. Taking the derivative of $g_{\epsilon_0}(\gamma_n)$ in (65), we have

$$\frac{d}{d\gamma_n}g_{\epsilon_0}(\gamma_n) = 1-\left(\frac{\beta^2 y_n^2}{\beta^2 y_n^2+\beta+\gamma_n}\right)^2. \quad (68)$$

It is easy to verify that, when $\gamma_n>0$, $0<\frac{d}{d\gamma_n}g_{\epsilon_0}(\gamma_n)<1$. Thus, the unique root $\gamma_n'=\frac{\beta}{\beta y_n^2-1}$ is a stable fixed point of the iteration. As $0<\frac{d}{d\gamma_n}g_{\epsilon_0}(\gamma_n)<1$ when $\gamma_n>0$, with an initial value $\gamma_n^{(0)}>0$, γ_n^t will converge to the stable fixed point γ_n' [43]. If $\beta y_n^2\leq 1$, the root $\gamma_n'=\beta/(\beta y_n^2-1)<0$ 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 $g_{\epsilon_0}(0)=\beta^2/((\beta y_n)^2+\beta)>0$, $y=g_{\epsilon_0}(\gamma_n)$ 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.

Appendix C
Proof of Theorem 1

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

$$\frac{dg_\epsilon(\gamma_n)}{d\gamma_n} = (2\epsilon+1)\left(1-\left(\frac{\beta u_n}{\beta u_n+\beta+\gamma_n}\right)^2\right), \quad (69)$$

where $u_n = \beta y_n^2$. To find the fixed points of the iteration, we let $f(\gamma_n) = g_\epsilon(\gamma_n) - \gamma_n = 0$, leading to

$$2\epsilon\gamma_n^2 - \gamma_n\beta(\beta y_n^2 - 4\epsilon - 1) + \beta^2(1+2\epsilon) = 0. \tag{70}$$

The two roots of (70) are given by [5]

$$\gamma_{n(a)} = \frac{2\beta(1+2\epsilon)}{u_n - 4\epsilon - 1 + \sqrt{u_n^2 - 8\epsilon u_n - 2u_n + 1}}, \tag{71}$$

and

$$\gamma_{n(b)} = \frac{2\beta(1+2\epsilon)}{u_n - 4\epsilon - 1 - \sqrt{u_n^2 - 8\epsilon u_n - 2u_n + 1}}. \tag{72}$$

If

$$u_n > 1 + 4\epsilon + 4\sqrt{\epsilon^2 + \epsilon/2}, \tag{73}$$

it is not hard to verify that

$$u_n - 4\epsilon - 1 - \sqrt{u_n^2 - 8\epsilon u_n - 2u_n + 1} > 0, \tag{74}$$

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 (69), we have

$$\left.\frac{d}{d\gamma_n} g_\epsilon(\gamma_n)\right|_{\gamma_n=\gamma_{n(a)}} = (2\epsilon+1)\left(1 - \left(\frac{\beta u_n}{\beta u_n + \beta + \gamma_{n(a)}}\right)^2\right). \tag{75}$$

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

$$\frac{d}{d\gamma_n} g_\epsilon(\gamma_n)|_{\gamma_n=\gamma_{n(a)}} < 1 \tag{76}$$

is equivalent to showing that

$$l(u_n) = (2\epsilon+1)(\beta u_n)^2 - 2\epsilon(\beta u_n + \beta + \gamma_{n(a)})^2 \tag{77}$$

is larger than 0. Inserting (71) into (77),

$$\frac{4\epsilon l(u_n)}{\beta^2} = l_1(u_n) + ((4\epsilon+1)u_n - 1)\sqrt{-l_1(u_n)}, \tag{78}$$

where

$$l_1(u_n) = -(u_n^2 - 8\epsilon u_n - 2u_n + 1) < 0. \tag{79}$$

Then

$$\frac{4\epsilon l(u_n)}{\beta^2} = \sqrt{-l_1(u_n)}\left(-\sqrt{-l_1(u_n)} + (4\epsilon u_n + u_n - 1)\right). \tag{80}$$

Because

$$(4\epsilon u_n + u_n - 1)^2 - (-l_1(u_n)) = 16\epsilon^2 u_n^2 + 8\epsilon u_n > 0, \tag{81}$$

the term in (80)

$$-\sqrt{-l_1(u_n)} + (4\epsilon u_n + u_n - 1) > 0, \tag{82}$$

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

$$\frac{d}{d\gamma_n} g_\epsilon(\gamma_n)|_{\gamma_n=\gamma_{n(a)}} < 1, \tag{83}$$

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.

[5] An alternative form for the quadratic formula is used, which can be deduced from the standard quadratic formula by Vieta's formulas.

We now analyze the convergence behavior. As $\gamma_n > 0$, the derivative (69) is an increasing function and it is positive. In the above, it is already shown that

$$\frac{d}{d\gamma_n} g_\epsilon(\gamma_n)|_{\gamma_n=\gamma_{n(a)}} < 1. \tag{84}$$

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

$$0 < \frac{d}{d\gamma_n} g_\epsilon(\gamma_n) < 1. \tag{85}$$

Thus, with an initial $\gamma_n^{(0)}$ with the range, γ_n^t converges to the stable fixed point $\gamma_{n(a)}$ [43].

Next we consider

$$u_n < 1 + 4\epsilon + 4\sqrt{\epsilon^2 + \epsilon/2}. \tag{86}$$

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

$$u_n^2 - 8\epsilon u_n - 2u_n + 1 < 0, \tag{87}$$

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

$$u_n \le 1 + 4\epsilon - 4\sqrt{\epsilon^2 + \epsilon/2}, \tag{88}$$

it can be shown that

$$u_n^2 - 8\epsilon u_n - 2u_n + 1 \ge 0, \tag{89}$$

and

$$u_n^2 - 8\epsilon u_n - 2u_n + 1 < (u_n - 4\epsilon - 1)^2. \tag{90}$$

Thus

$$u_n - 4\epsilon - 1 < -4\sqrt{\epsilon^2 + \epsilon/2} < 0 \tag{91}$$

and

$$u_n - 4\epsilon - 1 \pm \sqrt{u_n^2 - 8\epsilon u_n - 2u_n + 1} < 0, \tag{92}$$

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

$$u_n < 1 + 4\epsilon + 4\sqrt{\epsilon^2 + \epsilon/2}, \tag{93}$$

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

$$g_\epsilon(0) = (2\epsilon+1)\frac{\beta^2}{(\beta y_n)^2 + \beta} > 0, \tag{94}$$

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

When

$$u_n = 1 + 4\epsilon + 4\sqrt{\epsilon^2 + \epsilon/2}, \tag{95}$$

there is single root

$$\gamma_n^* = \frac{2\beta(1+2\epsilon)}{u_n - 1 - 4\epsilon}. \tag{96}$$

Plugging γ_n^* into (69), we have

$$\frac{d}{d\gamma_n} g_\epsilon(\gamma_n)|_{\gamma_n=\gamma_n^*} = 1. \tag{97}$$

Thus γ_n^* is neutral fixed point [43]. Depending on the initial value $\gamma_n^{(0)}$, γ_n^t may converge to the fixed point γ_n^* or diverge.

APPENDIX D
DERIVATION OF UAMP-SBL FOR MMV

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

$$b(\beta) \propto f_\beta(\beta)\prod_{l,m}\mathcal{M}_{f_{r_m}^{(l)}\to\beta}(\beta)$$
$$\propto 1/\beta\prod_{l,m}\mathcal{N}(h_m^{(l)}|r_m^{(l)},\hat{\beta}^{-1}). \tag{98}$$

Then according to $\hat{\beta} =< \beta >_{b(\beta)}$, we have

$$\hat{\beta} = ML/\sum_{m,l}\left(|r_m^{(l)}-\hat{h}_m^{(l)}|^2 + v_{h_m}^{(l)}\right). \tag{99}$$

According to the factor graph in Fig. 7, the belief $b(\gamma_n)$ can be updated as

$$b(\gamma_n^{(l)}) \propto \mathcal{M}_{f^{(l)}_{\gamma_n\to\gamma_n^{(l)}}}(\gamma_n^{(l)})\mathcal{M}_{f^{(l)}_{x_n\to\gamma_n^{(l)}}}(\gamma_n^{(l)})$$
$$= (\gamma_n^{(l)})^{\epsilon-1+\frac{1}{2}}\exp\left\{-\frac{\gamma_n^{(l)}}{2}(2\eta+(|\hat{x}_n^{(l)}|^2+\tau_x^{(l)}))\right\}. \tag{100}$$

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

$$\hat{\gamma}_n = \frac{2\epsilon'+1}{(1/L)\sum_{l=1}^{L}(|\hat{x}_n^{(l)}|^2+\tau_x^{(l)})}, \tag{101}$$

where $\epsilon' = \epsilon/L$. By comparing (101) with (64), the update of ϵ' can be expressed as

$$\epsilon' = \frac{1}{2}\sqrt{\log\left(\frac{1}{N}\sum\nolimits_n\hat{\gamma}_n\right) - \frac{1}{N}\sum\nolimits_n\log\hat{\gamma}_n}. \tag{102}$$

APPENDIX E
DERIVATION OF UAMP-TSBL

We only derive the message passing for the graph shown in Fig. 8. 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.,

$$\mathcal{M}_{f_{x_n}^{(l)}\to x_n^{(l)}}(x_n^{(l)})$$
$$= \left\langle f_{x_n}^{(l)}(x^{(l)})\right\rangle_{\mathcal{M}_{f_{x_n}^{(l-1)}\to x_n^{(l-1)}}\prod_m\mathcal{M}_{f_{\delta_m}^{(l-1)}\to x_n^{(l-1)}}}$$
$$\propto \mathcal{N}(x_n^{(l)}|\xi_n^{(l)},\psi_n^{(l)}), \tag{103}$$

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

$$\mathcal{M}_{f_{x_n}^{(l+1)}\to x_n^{(l)}}(x_n^{(l)})$$
$$= \left\langle f_{x_n}^{(l+1)}(x^{(l+1)})\right\rangle_{\mathcal{M}_{f_{x_n}^{(l+2)}\to x_n^{(l+1)}}\prod_m\mathcal{M}_{f_{\delta_m}^{(l+1)}\to x_n^{(l+1)}}}$$
$$\propto \mathcal{N}(x_n^{(l)}|\theta_n^{(l)},\phi_n^{(l)}), \tag{104}$$

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

$$b(x_n^{(l)}) \propto \mathcal{M}_{f_{x_n}^{(l)}\to x_n^{(l)}}\mathcal{M}_{f_{x_n}^{(l+1)}\to x_n^{(l)}}\prod_m\mathcal{M}_{f_{\delta_m}^{(l)}\to x_n^{(l)}}$$
$$\propto \mathcal{N}(x_n^{(l)}|\hat{x}_n^{(l)},\tau_x^{(l)}) \tag{105}$$

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

$$\mathcal{M}_{f_{x_n}^{(l)}\to\gamma_n}(\gamma_n) = \exp\left\{\left\langle f_{x_n}^{(l)}(x_n^{(l)}|\gamma_n)\right\rangle_{b(x_n^{(l)})b(x_n^{(l-1)})}\right\}. \tag{106}$$

Then, with the message $\mathcal{M}_{f_{\gamma_n}\to\gamma_n}(\gamma_n)$ in (60), the belief $b(\gamma_n)\propto\mathcal{M}_{f_{\gamma_n}\to\gamma_n}(\gamma_n)\mathcal{M}_{f_{x_n}\to\gamma_n}(\gamma_n)$. The update of $\hat{\gamma}_n$ is then expressed as

$$\hat{\gamma}_n = L(2\epsilon'+1)/(|\hat{\mathbf{x}}_n^{(1)}|^2+\tau_x^{(1)}+\frac{1}{\alpha^2}\sum_{l=2}^{L}(|\hat{\mathbf{x}}_n^{(l)}|^2+\tau_x^{(l)})$$
$$+\frac{\alpha^2}{1-\alpha^2}\sum_{l=1}^{L-1}(|\hat{\mathbf{x}}_n^{(l)}|^2+\tau_x^{(l)}) - \frac{2\alpha}{1-\alpha^2}\sum_{l=2}^{L}(\hat{\mathbf{x}}_n^{(l)}\hat{\mathbf{x}}_n^{(l-1)})), \tag{107}$$

completing the derivation.

REFERENCES

[1] Y. C. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2012.

[2] M. F. Duarte and Y. C. Eldar, "Structured compressed sensing: From theory to applications," *IEEE Trans. Signal Process.*, vol. 59, no. 9, pp. 4053–4085, Sep. 2011.

[3] J. Liu and B. D. Rao, "Sparse Bayesian learning for robust PCA: Algorithms and analyses," *IEEE Trans. Signal Process.*, vol. 67, no. 22, pp. 5837–5849, Oct. 2019.

[4] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Berlin, Germany: Springer, 2010.

[5] E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?," *IEEE Trans. Inf. Theory*, vol. 52, no. 12, pp. 5406–5425, Nov. 2006.

[6] 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 Trans. Comput. Imag.*, vol. 3, no. 4, pp. 566–579, Aug. 2017.

[7] 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 Trans. Veh. Technol.*, vol. 67, no. 10, pp. 9631–9640, Jul. 2018.

[8] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, Jun. 2001.

[9] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing: I motivation and construction," in *Proc. Inf. Theory Workshop Inf. Theory*, 2010, pp. 1–5.

[10] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proc. Nat. Acad. Sci.*, vol. 106, no. 45, pp. 18914–18919, Nov. 2009.

[11] M. Al-Shoukairi and B. Rao, "Sparse Bayesian learning using approximate message passing," in *Proc. 48th Asilomar Conf. Signals, Syst. Comput.*, 2014, pp. 1957–1961.

[12] J. Zhu, L. Han, and X. Meng, "An AMP-based low complexity generalized sparse Bayesian learning algorithm," *IEEE Access*, vol. 7, pp. 7965–7976, Dec. 2018.

[13] R. Tibshirani, "Regression shrinkage and selection via the lasso: A retrospective," *J. Roy. Statist. Soc.: Ser. B. (Statist. Methodol.)*, vol. 73, no. 3, pp. 273–282, Apr. 2011.

[14] T. Blumensath and M. E. Davies, "Normalized iterative hard thresholding: Guaranteed stability and performance," *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 2, pp. 298–309, Apr. 2010.

[15] A. Javanmard and A. Montanari, "State evolution for general approximate message passing algorithms, with applications to spatial coupling," *Inf. Inference: J. IMA*, vol. 2, no. 2, pp. 115–144, Dec. 2013.
[16] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," in *Proc. Int. Symp. Inf. Theory*, 2011, pp. 2168–2172.
[17] X. Meng, S. Wu, and J. Zhu, "A unified Bayesian inference framework for generalized linear models," *IEEE Signal Process. Lett.*, vol. 25, no. 3, pp. 398–402, Mar. 2018.
[18] S. Rangan, P. Schniter, A. K. Fletcher, and S. Sarkar, "On the convergence of approximate message passing with arbitrary matrices," *IEEE Trans. Inf. Theory*, vol. 65, no. 9, pp. 5339–5351, Sep. 2019.
[19] A. Manoel, F. Krzakala, E. W. Tramel, and L. Zdeborová, "Sparse estimation with the swept approximated message-passing algorithm," Jun. 2014, *arXiv:1406.4311*. [Online]. Available: https://arxiv.org/abs/1406.4311
[20] J. Vila, P. Schniter, S. Rangan, F. Krzakala, and L. Zdeborová, "Adaptive damping and mean removal for the generalized approximate message passing algorithm," in *Proc. 40th IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2015, pp. 2021–2025.
[21] Q. Guo and J. Xi, "Approximate message passing with unitary transformation," Apr. 2015, *arXiv:1504.04799*. [Online]. Available: http://arxiv.org/abs/1504.04799
[22] S. Rangan, P. Schniter, and A. K. Fletcher, "Vector approximate message passing," in *Proc. Int. Symp. Inf. Theory*, 2017, pp. 1588–1592.
[23] J. Ma and L. Ping, "Orthogonal AMP," *IEEE Access*, vol. 5, pp. 2020–2033, Jan. 2017.
[24] L. Liu, S. Huang, and B. M. Kurkoski, "Memory approximate message passing," in *Proc. IEEE Int. Symp. Inf. Theory*, 2021, pp. 1379–1384.
[25] K. Takeuchi, "Convolutional approximate message-passing," *IEEE Signal Process. Lett.*, vol. 27, pp. 416–420, Feb. 2020.
[26] M. Al-Shoukairi, P. Schniter, and B. D. Rao, "A GAMP-based low complexity sparse Bayesian learning algorithm," *IEEE Trans. Signal Process.*, vol. 66, no. 2, pp. 294–308, Jan. 2018.
[27] Z. Yuan, Q. Guo, and M. Luo, "Approximate message passing with unitary transformation for robust bilinear recovery," *IEEE Trans. Signal Process.*, vol. 69, pp. 617–630, Dec. 2020.
[28] Q. Guo, D. D. Huang, S. Nordholm, J. Xi, and Y. Yu, "Iterative frequency domain equalization with generalized approximate message passing," *IEEE Signal Process. Lett.*, vol. 20, no. 6, pp. 559–562, Jun. 2013.
[29] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Mach. Learn.*, vol. 37, no. 2, pp. 183–233, Nov. 1999.
[30] J. Winn and C. M. Bishop, "Variational message passing," *J. Mach. Learn. Res.*, vol. 6, pp. 661–694, Apr. 2005.
[31] E. P. Xing, M. I. Jordan, and S. Russell, "A generalized mean field algorithm for variational inference in exponential families," Oct. 2012, *arXiv:1212.2512*. [Online]. Available: https://arxiv.org/abs/1212.2512
[32] J. Dauwels, "On variational message passing on factor graphs," in *Proc. Int. Symp. Inf. Theory*, 2007, pp. 2546–2550.
[33] Y. Jin and B. D. Rao, "Support recovery of sparse signals in the presence of multiple measurement vectors," *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 3139–3157, May 2013.
[34] M. E. Davies and Y. C. Eldar, "Rank awareness in joint sparse recovery," *IEEE Trans. Inf. Theory*, vol. 58, no. 2, pp. 1135–1146, Feb. 2012.
[35] M. Luo, Q. Guo, D. Huang, and J. Xi, "Sparse Bayesian learning based on approximate message passing with unitary transformation," in *Proc. IEEE VTS Asia Pacific Wireless Commun. Symp.*, 2019, pp. 1–5.
[36] 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 J.*, vol. 20, no. 22, pp. 13734–13742, Nov. 2020.
[37] N. L. Pedersen, C. N. Manchón, D. Shutin, and B. H. Fleury, "Application of Bayesian hierarchical prior modeling to sparse channel estimation," in *Proc. IEEE Int. Conf. Commun*, 2012, pp. 3487–3492.
[38] Z. Zhang and B. D. Rao, "Sparse signal recovery in the presence of correlated multiple measurement vectors," in *Proc. 35th IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2010, pp. 3986–3989.
[39] Z. Zhang and B. D. Rao, "Sparse signal recovery with temporally correlated source vectors using sparse Bayesian learning," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 5, pp. 912–926, Sep. 2011.
[40] A. K. Fletcher and P. Schniter, "Learning and free energies for vector approximate message passing," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2017, pp. 4247–4251.
[41] D. P. Wipf and B. D. Rao, "An empirical Bayesian strategy for solving the simultaneous sparse approximation problem," *IEEE Trans. Signal Process.*, vol. 55, no. 7, pp. 3704–3716, Jul. 2007.
[42] R. L. Eubank, *A Kalman Filter Primer*. Boca Raton, FL, USA: CRC Press, 2005.
[43] R. L. Devaney, *A First Course in Chaotic Dynamical Systems: Theory and Experiment*. London, U.K.: Chapman & Hall/CRC, 2020.

Man Luo received the B.E. and M.E. degrees in electronic and communication engineering from the Harbin Institute of Technology, China, in 2012 and 2015, respectively. She is currently a Ph.D. candidate with the School of Electrical, Computer and Telecommunications Engineering, the University of Wollongong, Wollongong, Australia. Her research interests include statistical signal processing and compressed sensing.

Qinghua Guo (Senior Member, IEEE) received the B.E. degree in electronic engineering and the M.E. degree in signal and information processing from Xidian University, Xi'an, China, in 2001 and 2004, respectively, and the Ph.D. degree in electronic engineering from the City University of Hong Kong, Kowloon, Hong Kong SAR, China, in 2008. He is currently an Associate Professor with the School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, Wollongong, NSW, Australia, and an Adjunct Associate Professor with the School of Engineering, The University of Western Australia, Perth, WA, Australia. His research interests include signal processing, machine learning and telecommunications. He was a recipient of the Australian Research Council's inaugural Discovery Early Career Researcher Award.

Ming Jin (Member, IEEE) received the B.E. and Ph.D. degrees in electronic engineering from Xidian University, Xi'an, China, in 2005 and 2010, respectively. From 2013 to 2014, he was an Associate Researcher with the School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, Wollongong, NSW, Australia. He is currently a Professor with the Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo, China. His current research interests include cognitive radio, optimization and machine learning.

Yonina C. Eldar (Fellow, IEEE) received the B.Sc. degree in physics in 1995 and the B.Sc. degree in electrical engineering in 1996 both from Tel-Aviv University (TAU), Tel-Aviv, Israel, and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, USA, in 2002. She is currently a Professor with the Department of Mathematics and Computer Science, Weizmann Institute of Science, Rehovot, Israel. She was previously a Professor with the Department of Electrical Engineering, the Technion. She is also a Visiting Professor with MIT, a Visiting Scientist with the Broad Institute, and an Adjunct Professor with Duke University and was a Visiting Professor with Stanford. She is a member of the Israel Academy of Sciences and Humanities (elected 2017) and a EURASIP Fellow. Her research interests include the broad areas of statistical signal processing, sampling theory and compressed sensing, learning and optimization methods, and their applications to biology, medical imaging and optics.

Dr. Eldar was the recipient of the many awards for excellence in research and teaching, including the IEEE Signal Processing Society Technical Achievement Award (2013), the IEEE/AESS Fred Nathanson Memorial Radar Award (2014), and the IEEE Kiyo Tomiyasu Award (2016). She was a Horev Fellow of the Leaders in Science and Technology program at the Technion and an Alon Fellow. She was the recipient of the Michael Bruno Memorial Award from the Rothschild Foundation, the Weizmann Prize for Exact Sciences, the Wolf Foundation Krill Prize for Excellence in Scientific Research, the Henry Taub Prize for Excellence in Research (twice), the Hershel Rich Innovation Award

(three times), the Award for Women with Distinguished Contributions, the Andre and Bella Meyer Lectureship, the Career Development Chair at the Technion, the Muriel & David Jacknow Award for Excellence in Teaching, and the Technion.s Award for Excellence in Teaching (two times). She was the recipient of the several Best Paper Awards and Best Demo Awards together with her research students and colleagues including the SIAM outstanding Paper Prize, the UFFC Outstanding Paper Award, the Signal Processing Society Best Paper Award and the IET Circuits, Devices and Systems Premium Award, was selected as one of the 50 most influential women in Israel and in Asia, and is a highly cited researcher.

She was a member of the Young Israel Academy of Science and Humanities and the Israel Committee for Higher Education. She is the Editor in Chief of *Foundations and Trends in Signal Processing*, a member of the IEEE Sensor Array and Multichannel Technical Committee and serves on several other IEEE committees. In the past, she was a Signal Processing Society Distinguished Lecturer, member of the IEEE Signal Processing Theory and Methods and Bio Imaging Signal Processing technical committees, and was an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING, *the EURASIP Journal of Signal Processing*, *the SIAM Journal on Matrix Analysis and Applications*, and the *SIAM Journal on Imaging Sciences*. She was the Co-Chair and Technical Co-Chair of several international conferences and workshops. She is author of the book "*Sampling Theory: Beyond Bandlimited Systems*" and coauthor of four other books published by Cambridge University Press.

Defeng Huang (Senior Member, IEEE) received the B.E.E.E. and M.E.E.E. degrees in electronic engineering from Tsinghua University, Beijing, China, in 1996 and 1999, respectively, and the Ph.D. degree in electrical and electronic engineering from the Hong Kong University of Science and Technology, Kowloon, Hong Kong, in 2004.

He joined the School of Electrical, Electronic and Computer Engineering, the University of Western Australia in 2005 as a Lecturer, and has been promoted to be a Professor with the same School/Department since 2011. Before joining UWA, he was a Lecturer with Tsinghua University. He was the Editor (2011 – 2015) of the IEEE WIRELESS COMMUNICATIONS LETTERS, the Editor (2005 – 2011) of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, and the Editorial Assistant (2002 – 2004) for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS.

Xiangming Meng received the B.E. degree in communication engineering from Xidian University, Xi'an, China, in 2011, and the Ph.D. degree in information and communication engineering from Tsinghua University, Beijing, China, in 2016. From 2016 to 2019, he was with Huawei Technologies, Company Ltd., Shanghai, China as a Senior Engineer. He was a Postdoctoral Researcher with RIKEN Center for Advanced Intelligence Project (AIP), Tokyo, Japan from 2019 to 2020. Since April 2020, he has been a Postdoctoral Researcher with the University of Tokyo, Tokyo, Japan. His research interests include graphical models, approximate Bayesian inference, and learning algorithms.