Bilinear Recovery Using Adaptive Vector-AMP

Subrata Sarkar, *Student Member, IEEE*, Alyson K. Fletcher, *Member, IEEE*, Sundeep Rangan, *Fellow, IEEE*, and Philip Schniter, *Fellow, IEEE*

***Abstract*—We consider the problem of jointly recovering the vector b and the matrix C from noisy measurements $Y = A(b)C + W$, where $A(\cdot)$ is a known affine linear function of b (i.e., $A(b) = A_0 + \sum_{i=1}^{Q} b_i A_i$ with known matrices A_i). This problem has applications in matrix completion, robust PCA, dictionary learning, self-calibration, blind deconvolution, joint-channel/symbol estimation, compressive sensing with matrix uncertainty, and many other tasks. To solve this bilinear recovery problem, we propose the Bilinear Adaptive Vector Approximate Message Passing (VAMP) algorithm. We demonstrate numerically that the proposed approach is competitive with other state-of-the-art approaches to bilinear recovery, including lifted VAMP and Bilinear Generalized Approximate Message Passing.**

***Index Terms*—Approximate message passing, expectation propagation, expectation maximization, self-calibration, computed tomography, dictionary learning.**

I. Introduction

A. Motivation

MANY problems of interest in science and engineering can be formulated as estimation of a structured matrix $\boldsymbol{Z}$ from noisy or incomplete measurements. The type of structure in $\boldsymbol{Z}$ determines the specific subproblem to be solved.

For example, when $\boldsymbol{Z}$ has a low-rank structure and only a subset of its entries are observed, the problem is known as *matrix completion* [1]. When $\boldsymbol{Z} = \boldsymbol{L} + \boldsymbol{S}$ for low-rank $\boldsymbol{L}$ and sparse $\boldsymbol{S}$, the problem of estimating $\boldsymbol{L}$ and $\boldsymbol{S}$ is known as *robust principle components analysis* (RPCA) [2]. When $\boldsymbol{Z} = \boldsymbol{BC}$ with sparse $\boldsymbol{C}$, the problem of estimating $\boldsymbol{B}$ and $\boldsymbol{C}$ is known as *dictionary learning* [3]. When $\boldsymbol{Z} = \boldsymbol{BC}$ with nonnegative $\boldsymbol{B}$ and $\boldsymbol{C}$, the problem is known as *nonnegative matrix factorization* (NMF) [4].

Sometimes $\boldsymbol{Z}$ has a more complicated structure. For example, the problems of *self-calibration* and *blind (circular) deconvolution* [5] can be formulated using $\boldsymbol{Z} = \mathrm{Diag}(\boldsymbol{Hb})\boldsymbol{\Psi C}$, where $\boldsymbol{H}$ and $\boldsymbol{\Psi}$ are known and $\boldsymbol{b}$ and $\boldsymbol{C}$ are to be estimated.[1] The problem of *compressive sensing (CS) with matrix uncertainty* [6] can be formulated using $\boldsymbol{z} = \sum_i b_i \boldsymbol{A}_i \boldsymbol{c}$, where $\{\boldsymbol{A}_i\}$ are known and where $\boldsymbol{b}$ and sparse $\boldsymbol{c}$ are to be estimated. The latter covers the problem of *joint channel-symbol estimation* [7], in which case b_i are the data symbols, $\boldsymbol{c}$ contains (possibly sparse) channel coefficients, and the known $\{\boldsymbol{A}_i\}$ are determined by the modulation scheme. The more general problem of *matrix CS* [8], [9] results from

$$z_m = \mathrm{tr}\{\boldsymbol{A}_m^{\mathsf{T}}(\boldsymbol{L} + \boldsymbol{S})\} \text{ for } m = 1, \ldots, M, \tag{1}$$

where $\{\boldsymbol{A}_m\}$ are known and the goal is to estimate low-rank $\boldsymbol{L}$ and sparse $\boldsymbol{S}$.

B. Prior Work

Many algorithms have been developed to solve the above problems. Some solve a convex relaxation of the original problem, while others attack non-convex formulations via alternating methods, greedy methods, variational methods, message-passing methods, and other techniques.

For matrix completion, well-known approaches include the nuclear-norm-based convex optimization method IALM [10], the non-convex successive over-relaxation approach LMAFit [11], the Grassmanian gradient-descent approach GROUSE [12], the greedy hard-thresholding approach Matrix-ALPS [13], and the variational-Bayes method VSBL [14]. For RPCA, there are also versions of IALM [10], LMaFit [11], and VSBL [14], as well as a robust cousin of GROUSE, called GRASTA [15]. For dictionary learning, there is the greedy K-SVD algorithm [16], the online SPAMS approach [17], and the ER-SpUD approach from [18]. A unified approach to matrix completion, RPCA, and dictionary learning was proposed in [19]–[21] using an extension of the approximate message-passing (AMP) methodology from [22], [23]. The resulting "bilinear generalized AMP" (BiGAMP) algorithm was compared to the aforementioned methods in [20] and found (empirically) to be competitive, if not superior, in phase transition and runtime. A related approach known as LowRAMP was proposed [24] and analyzed in [25], [26].

Manuscript received August 31, 2018; revised March 8, 2019; accepted May 1, 2019. Date of publication May 10, 2019; date of current version May 28, 2019. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Gwo Giun Lee. The work of S. Sarkar and P. Schniter was supported by the National Science Foundation under Grant 1716388. The work of A. K. Fletcher was supported in part by the National Science Foundation under Grant 1738285 and Grant 1738286, and in part by the Office of Naval Research under Grant N00014-15-1-2677. The work of S. Rangan was supported in part by the National Science Foundation under Grant 1116589, Grant 1302336, and Grant 1547332, and in part by the industrial affiliates of NYU WIRELESS. This paper was presented in part at the Information Theory and Applications Workshop, University of California San Diego, San Diego, CA, USA, February 2018. *(Corresponding author: Philip Schniter.)*

S. Sarkar and P. Schniter are with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43202 USA (e-mail: sarkar.51@osu.edu; schniter.1@osu.edu).

A. K. Fletcher is with the Department of Statistics and Electrical Engineering, the University of California, Los Angeles, Los Angeles, CA 90095 USA (e-mail: akfletcher@ucla.edu).

S. Rangan is with the Department of Electrical and Computer Engineering, New York University, Brooklyn, NY 11201 USA (e-mail: srangan@nyu.edu).

Digital Object Identifier 10.1109/TSP.2019.2916100

[1] Here and in the sequel, we use lowercase bold notation for vectors and uppercase bold notation for matrices.

For self-calibration and blind deconvolution, well-known approaches include the convex relaxations from [5], [27], [28] and the alternating method from [29]. For CS with matrix uncertainty, there is the award-winning non-convex method [6]. For matrix CS, the well-known papers [8], [30], [31] proposed convex approaches and [9], [13] proposed greedy approaches. See the recent overview [32] for many other works. An AMP-based approach to self-calibration, blind deconvolution, CS with matrix uncertainty, and matrix CS was proposed in [33] and analyzed in [34]. This "parametric BiGAMP" (PBiGAMP) was compared to the above works in [33] and found to yield improved empirical phase transitions.

More recently, AMP methods for bilinear inference were proposed using the "lifting" approach (see, e.g., [5], [28], [32], [35] for seminal papers on lifting). To illustrate the idea, suppose that the measurement vector $\boldsymbol{y} \in \mathbb{R}^M$ takes the form

$$\boldsymbol{y} = \sum_{i=1}^{Q} \sum_{j=1}^{N} b_i \boldsymbol{a}_{i,j} c_j + \boldsymbol{w}, \tag{2}$$

where $\boldsymbol{a}_{i,j} \in \mathbb{R}^M$ is known for all i, j and the goal is to recover $\boldsymbol{b} = [b_1, \ldots, b_Q]^\mathsf{T}$ and $\boldsymbol{c} = [c_1, \ldots, c_N]^\mathsf{T}$ in the presence of white noise $\boldsymbol{w}$. Rewriting the measurements as

$$\boldsymbol{y} = \sum_{i=1}^{Q} b_i \underbrace{\left[\boldsymbol{a}_{i,1}, \ldots, \boldsymbol{a}_{i,N}\right]}_{\triangleq \boldsymbol{A}_i} \boldsymbol{c} + \boldsymbol{w} \tag{3}$$

$$= \underbrace{\left[\boldsymbol{A}_1 \cdots \boldsymbol{A}_Q\right]}_{\triangleq \boldsymbol{A}} \begin{bmatrix} b_1 \boldsymbol{c} \\ \vdots \\ b_Q \boldsymbol{c} \end{bmatrix} + \boldsymbol{w} \tag{4}$$

$$= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w} \text{ for } \boldsymbol{x} = \boldsymbol{b} \otimes \boldsymbol{c} = \operatorname{vec}(\boldsymbol{c}\boldsymbol{b}^\mathsf{T}), \tag{5}$$

we see that the noisy bilinear recovery problem (2) can be rewritten as the noisy linear recovery problem (5) with a rank-one structure on (the matrix form of) $\boldsymbol{x}$. Thus, if this low-rank signal structure can be exploited by a linear inference algorithm, then bilinear inference can be accomplished. This is precisely what was proposed in [36], building on the non-separable-denoising version of the AMP algorithm from [37]. A rigorous analysis of "lifted AMP" was presented in [38].

The trouble with AMP is that its behavior is understood only in the case of large [39] or infinitely large, i.i.d. (sub) Gaussian $\boldsymbol{A}$ [40], [41]. Even small deviations from this scenario (e.g., mildly ill-conditioned and/or non-zero-mean $\boldsymbol{A}$) can cause AMP to diverge [42]–[44]. To address this issue, an alternative called Vector AMP (VAMP) was proposed and analyzed in [45], with close connections to expectation propagation [46] (see also [47]–[49]). There it was established that, if $\boldsymbol{A}$ is an infinitely large right-rotationally invariant[2] random matrix and the denoising function used by VAMP is separable and Lipschitz, then VAMP's performance can be exactly predicted by a scalar state-evolution that also provides testable conditions for optimality. Since the class of right-rotationally invariant matrices is much larger than the class of i.i.d. Gaussian matrices, VAMP is much more robust than AMP with regards to the construction of $\boldsymbol{A}$. For example, VAMP has no problem with ill-conditioned or mean-shifted matrices [45].

Very recently, [50] performed a rigorous analysis of VAMP under *non*-separable Lipschitz denoisers, showing that—here too—VAMP's behavior is exactly predicted by a scalar state-evolution when $\boldsymbol{A}$ is infinitely large and right-rotationally invariant. Furthermore, [50] demonstrated the success of lifted VAMP on bilinear problems such as self-calibration and CS with matrix uncertainty. In addition, [50] gave evidence that, like AMP, the PBiGAMP algorithm is sensitive to deviations from the i.i.d. assumptions used in its derivation [33] and analysis [34]. For this reason, lifted VAMP significantly outperformed PBiGAMP in some cases [50].

Despite its good performance and rigorous analyses under infinitely large right-rotationally invariant random $\boldsymbol{A}$, lifted VAMP suffers from computational issues brought on by the lifting itself: The $N + Q$ unknowns $[\boldsymbol{b}, \boldsymbol{c}]$ in the bilinear problem (2) manifest as NQ unknowns $\boldsymbol{x}$ after lifting to (5). This is a serious problem when N and Q are both large. As a concrete example, consider the application of lifting to (square) dictionary learning, where the goal is to recover $\boldsymbol{B} \in \mathbb{R}^{N \times N}$ and sparse $\boldsymbol{C} \in \mathbb{R}^{N \times L}$ from noisy measurements $\boldsymbol{Y} = \boldsymbol{B}\boldsymbol{C} + \boldsymbol{W}$. This bilinear relationship can be lifted via

$$\boldsymbol{Y} = \sum_{ij} b_{i,j} \boldsymbol{A}_{i,j} \boldsymbol{C} + \boldsymbol{W} \tag{6}$$

$$= \underbrace{[\boldsymbol{A}_{1,1} \cdots \boldsymbol{A}_{N,N}]}_{\triangleq \boldsymbol{A} \in \mathbb{R}^{N \times N^3}} \underbrace{(\boldsymbol{b} \otimes \boldsymbol{C})}_{\triangleq \boldsymbol{X} \in \mathbb{R}^{N^3 \times L}} + \boldsymbol{W}, \tag{7}$$

where $\boldsymbol{A}_{i,j} \in \mathbb{R}^{N \times N}$ is constructed with a 1 in the (i, j)th position and zeros elsewhere, and where $\boldsymbol{b} = [b_{1,1}, \ldots, b_{N,N}]^\mathsf{T} \in \mathbb{R}^{N^2}$. Even at the relatively small patch size of 8×8 (i.e., $N = 64$), the matrix $\boldsymbol{A}$ has dimension $64 \times 262\,144$, and the unknown matrix $\boldsymbol{X}$ has dimension $262\,144 \times L$. The rule-of-thumb $L = 5N \ln N$ [18] then gives $L = 1331$, in which case $\boldsymbol{X}$ contains 3.5×10^8 entries, which leads to difficulties with computation and memory.

C. Contributions

In this paper, we present a novel VAMP-based approach to bilinear recovery. With the aim of computational efficiency, we avoid lifting and instead build on the recently proposed Adaptive VAMP framework from [51]. However, different from [51], which focused on noisy *linear* recovery, we focus on noisy *bilinear* recovery.

In particular, we focus on recovering $\{b_i\}$ and $\boldsymbol{C}$ from noisy measurements $\boldsymbol{Y} \in \mathbb{R}^{M \times L}$ of the form

$$\boldsymbol{Y} = \sum_{i=1}^{Q} b_i \boldsymbol{A}_i \boldsymbol{C} + \boldsymbol{W}, \tag{8}$$

[2]If $\boldsymbol{A}$ is right-rotationally invariant then its singular value decomposition $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\mathsf{T}$ has Haar distributed $\boldsymbol{V}$, i.e., $\boldsymbol{V}$ is uniformly distributed over the group of orthogonal matrices.

where $\{\boldsymbol{A}_i\}$ are known and $\boldsymbol{W}$ contains white noise. Note that (8) is a multiple-measurement vector (MMV) extension of (3), and that it covers all of the motivating problems discussed in Section I-A. For example, in self-calibration, where we estimate $\boldsymbol{b}$ and $\boldsymbol{C}$ from $\boldsymbol{Y} = \text{Diag}(\boldsymbol{Hb})\boldsymbol{\Psi}\boldsymbol{C} + \boldsymbol{W}$, we can set $\boldsymbol{A}_i = \text{Diag}(\boldsymbol{h}_i)\boldsymbol{\Psi}$, where $\boldsymbol{h}_i$ is the ith column of $\boldsymbol{H}$. Or, in dictionary learning, where we estimate $\boldsymbol{B}$ and $\boldsymbol{C}$ from $\boldsymbol{Y} = \boldsymbol{BC} + \boldsymbol{W}$, we can write $\boldsymbol{B} = \sum_{i=1}^{MN} b_i \boldsymbol{A}_i$ for $\boldsymbol{A}_i = \boldsymbol{e}_{\langle i-1\rangle_M} \boldsymbol{e}^{\mathsf{T}}_{\lfloor (i-1)/M \rfloor}$, where $\langle i \rangle_M$ denotes i-modulo-M, $\lfloor \cdot \rfloor$ denotes floor, and $\{\boldsymbol{e}_i\}$ is the standard basis.

When deriving[3] the proposed method, we treat $\{b_i\}$ as deterministic unknowns and the entries of $\boldsymbol{C}$ as random variables. The prior distribution on $\boldsymbol{C}$ is assumed to be known up to some (possibly) unknown hyperparameters, which are learned jointly with $\{b_i\}$ and $\boldsymbol{C}$. Also, $\boldsymbol{W}$ is treated as additive white Gaussian noise (AWGN) with an unknown variance that is also learned. More details are provided in the sequel.

We show (empirically) that the proposed Bilinear Adaptive VAMP (BAd-VAMP) method performs as well as the EM-PBiGAMP algorithm from [33], with regard to accuracy and computational complexity, when the underlying matrices are i.i.d., as assumed for the derivation of PBiGAMP. However, we will show that BAd-VAMP outperforms EM-PBiGAMP when the underlying matrices become ill-conditioned. In the ill-conditioned case, we show that BAd-VAMP performs as well as, and sometimes significantly better than, lifted VAMP. However, BAd-VAMP is much more computationally efficient due to its avoidance of lifting. In this sense, the proposed BAd-VAMP is shown to be accurate, robust, and computationally efficient.

Notation: In this paper, we use boldface uppercase letters to denote matrices (e.g., $\boldsymbol{X}$), boldface lowercase letters for vectors (e.g., $\boldsymbol{x}$) and non-bold letters for scalars (e.g., x). Given a matrix $\boldsymbol{X}$, we use $\boldsymbol{x}_l$ to denote the lth column and x_{nl} to denote the element in the nth row and lth column. We use $\mathbb{E}[f(\boldsymbol{x})|b]$ to denote the expectation of $f(\boldsymbol{x})$ w.r.t. the density b, i.e., $\mathbb{E}[f(\boldsymbol{x})|b] = \int f(\boldsymbol{x})b(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$, and we use $\text{var}[f(\boldsymbol{x})|b]$ for the corresponding variance. We use $\text{Diag}(\boldsymbol{x})$ to denote the diagonal matrix created from vector $\boldsymbol{x}$, and $\text{diag}(\boldsymbol{X})$ to denote the vector of elements on the diagonal of the matrix $\boldsymbol{X}$.

II. Proposed Framework

In an effort to make our algorithmic development more consistent with the VAMP papers [45], [50], [51], we now make some minor notational changes relative to (8). First, we will use the notation $\boldsymbol{A}(\boldsymbol{b}) \triangleq \sum_i b_i \boldsymbol{A}_i$ to be concise. Second, the quantities b_i and $\boldsymbol{C}$ in (8) will be changed to $\theta_{A,i}$ and $\boldsymbol{X}$, respectively.

The problem of interest can thus be stated as follows: estimate the matrix $\boldsymbol{X} \in \mathbb{R}^{N \times L}$ and learn the parameters $\boldsymbol{\Theta} \triangleq \{\boldsymbol{\theta}_A, \boldsymbol{\theta}_x, \gamma_w\}$ in the statistical model

$$\boldsymbol{Y} = \boldsymbol{A}(\boldsymbol{\theta}_A)\boldsymbol{X} + \boldsymbol{W} \tag{9a}$$

$$\boldsymbol{X} \sim p_{\mathbf{X}}(\cdot;\boldsymbol{\theta}_x), \quad w_{ml} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \gamma_w^{-1}), \tag{9b}$$

where $\boldsymbol{A}(\cdot)$ is a known matrix-valued linear function, and $p_{\mathbf{X}}(\cdot;\boldsymbol{\theta}_x)$ is a density on $\boldsymbol{X}$ parameterized by the vector $\boldsymbol{\theta}_x$. Here, γ_w is the noise precision, i.e., the inverse noise variance.

More precisely, we aim to compute the maximum-likelihood (ML) estimate of $\boldsymbol{\Theta}$ and, under that estimate, compute the minimum mean-squared error (MMSE) estimate of $\boldsymbol{X}$, i.e.,

$$\widehat{\boldsymbol{\Theta}}_{\mathsf{ML}} = \arg\max_{\boldsymbol{\Theta}} p_{\mathbf{Y}}(\boldsymbol{Y};\boldsymbol{\Theta}) \tag{10}$$

$$\widehat{\boldsymbol{X}}_{\mathsf{MMSE}} = \mathbb{E}[\boldsymbol{X}|\boldsymbol{Y};\widehat{\boldsymbol{\Theta}}_{\mathsf{ML}}]. \tag{11}$$

In (10), $p_{\mathbf{Y}}(\boldsymbol{Y};\boldsymbol{\Theta})$ is the likelihood function of $\boldsymbol{\Theta}$, which can be written as

$$p_{\mathbf{Y}}(\boldsymbol{Y};\boldsymbol{\Theta}) = \int p_{\mathbf{X}}(\boldsymbol{X};\boldsymbol{\Theta}) p_{\mathbf{Y}|\mathbf{X}}(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{\Theta})\,\mathrm{d}\boldsymbol{X}. \tag{12}$$

In (11), the expectation is taken over the posterior density

$$p_{\mathbf{X}|\mathbf{Y}}(\boldsymbol{X}|\boldsymbol{Y};\widehat{\boldsymbol{\Theta}}_{\mathsf{ML}}) = \frac{p_{\mathbf{X}}(\boldsymbol{X};\widehat{\boldsymbol{\Theta}}_{\mathsf{ML}}) p_{\mathbf{Y}|\mathbf{X}}(\boldsymbol{Y}|\boldsymbol{X};\widehat{\boldsymbol{\Theta}}_{\mathsf{ML}})}{p_{\mathbf{Y}}(\boldsymbol{Y};\widehat{\boldsymbol{\Theta}}_{\mathsf{ML}})}. \tag{13}$$

The statistical model (9) implies that[4]

$$p_{\mathbf{Y}|\mathbf{X}}(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{\Theta}) = \prod_{l=1}^{L} p_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}_l|\boldsymbol{x}_l;\boldsymbol{\Theta}) \tag{14}$$

where

$$p_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta}) = \mathcal{N}\big(\boldsymbol{y};\boldsymbol{A}(\boldsymbol{\theta}_A)\boldsymbol{x}, \boldsymbol{I}/\gamma_w\big). \tag{15}$$

For simplicity, we treat $\{\boldsymbol{x}_l\}_{l=1}^{L}$ as i.i.d. in the sequel, so that

$$p_{\mathbf{X}}(\boldsymbol{X};\boldsymbol{\Theta}) = \prod_{l=1}^{L} p_{\mathbf{x}}(\boldsymbol{x}_l;\boldsymbol{\theta}_x) \tag{16}$$

for some density $p_{\mathbf{x}}(\cdot;\boldsymbol{\theta}_x)$ parameterized by $\boldsymbol{\theta}_x$. In this case, the posterior density decouples as

$$p_{\mathbf{X}|\mathbf{Y}}(\boldsymbol{X}|\boldsymbol{Y};\boldsymbol{\Theta}) \propto \prod_{l=1}^{L} p_{\mathbf{x}}(\boldsymbol{x}_l;\boldsymbol{\Theta}) p_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}_l|\boldsymbol{x}_l;\boldsymbol{\Theta}). \tag{17}$$

III. Background

A. Background on VAMP

Recalling (11), we are interested in computing the MMSE estimate of $\boldsymbol{X}$ from the noisy measurements $\boldsymbol{Y}$. This problem is solved (under certain conditions) by the VAMP approach from [45]. We now review VAMP at the detail needed for further development of BAd-VAMP.

From (11), the MMSE estimate of $\boldsymbol{X}$ equals the mean of the posterior pdf $p_{\mathbf{X}|\mathbf{Y}}$. Because $p_{\mathbf{X}|\mathbf{Y}}$ decouples across the columns of $\boldsymbol{X}$, as in (17), it suffices to consider a single column and drop the l notation for simplicity. Also, for now, we will assume that $\boldsymbol{\Theta}$ are the true parameters used to generate $(\boldsymbol{Y},\boldsymbol{X})$ and drop the $\boldsymbol{\Theta}$ notation for simplicity; we will revisit the estimation of $\boldsymbol{\Theta}$ in Section III-B. With these simplifications, (9) reduces to

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}/\gamma_w), \quad \boldsymbol{x} \sim p_{\mathbf{x}}. \tag{18}$$

[3]Although the derivation treats the entries of $\boldsymbol{C}$ as random variables and the associated denoiser as Bayesian, the final algorithm is more general in that it only requires the denoiser to be Lipschitz.

[4]In (14)–(16), to promote notational simplicity, the left side of the equation is written using $\boldsymbol{\Theta}$ even though the right side depends on a subset of $\boldsymbol{\Theta}$.

Recall that the MMSE estimate of $\boldsymbol{x}$ equals

$$\mathbb{E}[\boldsymbol{x}|\boldsymbol{y}] = \int \boldsymbol{x}\, p_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x}|\boldsymbol{y})\, \mathrm{d}\boldsymbol{x} \tag{19}$$

for the posterior density

$$p_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x}|\boldsymbol{y}) = Z(\boldsymbol{y})^{-1} p_{\mathbf{x}}(\boldsymbol{x}) p_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}|\boldsymbol{x}), \tag{20}$$

where $Z(\boldsymbol{y})$ is the normalizing constant

$$Z(\boldsymbol{y}) \triangleq \int p_{\mathbf{x}}(\boldsymbol{x}) p_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}|\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}. \tag{21}$$

For high-dimensional $\boldsymbol{x}$, the integrals in (19) and (21) are difficult to compute directly. Thus other methods must be used.

Variational inference (VI) [52] can be used to bypass the computation of $Z(\boldsymbol{y})$. For example, notice that the true posterior $p_{\mathbf{x}|\mathbf{y}}$ can be recovered by solving the variational optimization (over densities)

$$\widehat{q} = \arg\min_{q} D_{\mathsf{KL}}(q \,\|\, p_{\mathbf{x}|\mathbf{y}}), \tag{22}$$

where $D_{\mathsf{KL}}(q \,\|\, p)$ denotes the KL divergence from p to q, i.e.,

$$D_{\mathsf{KL}}(q \,\|\, p) \triangleq \int q(\boldsymbol{x}) \ln \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\, \mathrm{d}\boldsymbol{x}. \tag{23}$$

Plugging (20) into (23), we see that

$$D_{\mathsf{KL}}(q \,\|\, p_{\mathbf{x}|\mathbf{y}}) = D_{\mathsf{KL}}(q \,\|\, p_{\mathbf{x}}) + D_{\mathsf{KL}}(q \,\|\, p_{\mathbf{y}|\mathbf{x}}) + H(q) + \ln Z(\boldsymbol{y}) \tag{24}$$

where $H(q) \triangleq -\int q(\boldsymbol{x}) \ln q(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}$ is the differential entropy of $\boldsymbol{x} \sim q$. Thus it follows from (22) and (24) that

$$\widehat{q} = \arg\min_{q} \big\{ D_{\mathsf{KL}}(q \,\|\, p_{\mathbf{x}}) + D_{\mathsf{KL}}(q \,\|\, p_{\mathbf{y}|\mathbf{x}}) + H(q) \big\}, \tag{25}$$

which bypasses $Z(\boldsymbol{y})$. Still, solving (25) is difficult in most cases of interest. The typical response is to impose constraints on q, but doing so compromises $\widehat{q}$ and its mean.

We take a different approach. Using the "Gibbs free energy"

$$J(q_1, q_2, q_3) \triangleq D_{\mathsf{KL}}(q_1 \| p_{\mathbf{x}}) + D_{\mathsf{KL}}(q_2 \| p_{\mathbf{y}|\mathbf{x}}) + H(q_3), \tag{26}$$

one can rewrite (25) as[5]

$$\arg\min_{q_1} \min_{q_2} \max_{q_3} J(q_1, q_2, q_3) \tag{27a}$$

$$\text{s.t. } q_1 = q_2 = q_3. \tag{27b}$$

But, as discussed earlier, (27) is difficult to solve. In the *expectation consistent approximate inference* (EC) scheme proposed by Opper and Winther in [46], the density constraint (27b) is relaxed to moment-matching constraints, i.e.,

$$\arg\min_{q_1} \min_{q_2} \max_{q_3} J(q_1, q_2, q_3) \tag{28a}$$

$$\text{s.t. } \mathbb{E}[\boldsymbol{x}|q_1] = \mathbb{E}[\boldsymbol{x}|q_2] = \mathbb{E}[\boldsymbol{x}|q_3] \tag{28b}$$

$$\mathrm{tr}\{\mathrm{Cov}[\boldsymbol{x}|q_1]\} = \mathrm{tr}\{\mathrm{Cov}[\boldsymbol{x}|q_2]\} = \mathrm{tr}\{\mathrm{Cov}[\boldsymbol{x}|q_3]\}, \tag{28c}$$

Algorithm 1: VAMP algorithm [45].

```
1:  initialize:
        $\boldsymbol{r}_1^0, \gamma_1^0$
2:  for $t = 0, \ldots, T_{\max}$ do
3:      $\boldsymbol{x}_1^t = \boldsymbol{g}_1(\boldsymbol{r}_1^t, \gamma_1^t)$
4:      $1/\eta_1^t = \langle \boldsymbol{g}_1'(\boldsymbol{r}_1^t, \gamma_1^t) \rangle / \gamma_1^t$
5:      $\gamma_2^t = \eta_1^t - \gamma_1^t$
6:      $\boldsymbol{r}_2^t = (\eta_1^t \boldsymbol{x}_1^t - \gamma_1^t \boldsymbol{r}_1^t)/\gamma_2^t$
7:      $\boldsymbol{x}_2^t = \boldsymbol{g}_2(\boldsymbol{r}_2^t, \gamma_2^t)$
8:      $1/\eta_2^t = \langle \boldsymbol{g}_2'(\boldsymbol{r}_2^t, \gamma_2^t) \rangle / \gamma_2^t$
9:      $\gamma_1^{t+1} = \eta_2^t - \gamma_2^t$
10:     $\boldsymbol{r}_1^{t+1} = (\eta_2^t \boldsymbol{x}_2^t - \gamma_2^t \boldsymbol{r}_2^t)/(\eta_2^t - \gamma_2^t)$
11: end for
```

where $\mathbb{E}[\boldsymbol{x}|q^t]$ denotes $\mathbb{E}[\boldsymbol{x}]$ under $\boldsymbol{x} \sim q^t$. This yields stationary points of the form

$$q_1(\boldsymbol{x}) \propto p_{\mathbf{x}}(\boldsymbol{x}) \exp\left(-\tfrac{\gamma_1}{2} \|\boldsymbol{x} - \boldsymbol{r}_1\|_2^2\right) \tag{29a}$$

$$q_2(\boldsymbol{x}) \propto p_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}|\boldsymbol{x}) \exp\left(-\tfrac{\gamma_2}{2} \|\boldsymbol{x} - \boldsymbol{r}_2\|_2^2\right) \tag{29b}$$

$$q_3(\boldsymbol{x}) \propto \exp\left(-\tfrac{\eta}{2} \|\boldsymbol{x} - \widehat{\boldsymbol{x}}\|_2^2\right), \tag{29c}$$

for $\{\boldsymbol{r}_1, \gamma_1, \boldsymbol{r}_2, \gamma_2, \widehat{\boldsymbol{x}}, \eta\}$ that lead to satisfaction of (28b)–(28c). Various approaches can be used to solve for $\{\boldsymbol{r}_1, \gamma_1, \boldsymbol{r}_2, \gamma_2, \widehat{\boldsymbol{x}}, \eta\}$. One is to alternate the update of $\{(\boldsymbol{r}_1, \gamma_1), (\widehat{\boldsymbol{x}}, \eta)\}$ and $\{(\boldsymbol{r}_2, \gamma_2), (\widehat{\boldsymbol{x}}, \eta)\}$ such that, at each iteration, the moments of q_3 are consistent with either q_1 or q_2. This approach is summarized in Algorithm 1 using[6]

$$\boldsymbol{g}_1(\boldsymbol{r}_1, \gamma_1) \triangleq \frac{\int \boldsymbol{x}\, p_{\mathbf{x}}(\boldsymbol{x}) \mathcal{N}(\boldsymbol{x}; \boldsymbol{r}_1, \boldsymbol{I}/\gamma_1)\, \mathrm{d}\boldsymbol{x}}{\int p_{\mathbf{x}}(\boldsymbol{x}) \mathcal{N}(\boldsymbol{x}; \boldsymbol{r}_1, \boldsymbol{I}/\gamma_1)\, \mathrm{d}\boldsymbol{x}} \tag{30}$$

$$\boldsymbol{g}_2(\boldsymbol{r}_2, \gamma_2) \triangleq \frac{\int \boldsymbol{x}\, p_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}|\boldsymbol{x}) \mathcal{N}(\boldsymbol{x}; \boldsymbol{r}_2, \boldsymbol{I}/\gamma_2)\, \mathrm{d}\boldsymbol{x}}{\int p_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}|\boldsymbol{x}) \mathcal{N}(\boldsymbol{x}; \boldsymbol{r}_2, \boldsymbol{I}/\gamma_2)\, \mathrm{d}\boldsymbol{x}}, \tag{31}$$

which, under these definitions of $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$, can be recognized as an instance of *expectation propagation* (EP) [53, Section 3.2], [54], [55]. In lines 4 and 8, $\boldsymbol{g}_i'(\boldsymbol{r}_i, \gamma_i) \in \mathbb{R}^N$ denotes the diagonal of the Jacobian matrix of $\boldsymbol{g}_i(\cdot, \gamma_i)$ at $\boldsymbol{r}_i$, i.e.,

$$\boldsymbol{g}_i'(\boldsymbol{r}_i, \gamma_i) \triangleq \mathrm{diag}\left(\frac{\partial \boldsymbol{g}_i(\boldsymbol{r}_i, \gamma_i)}{\partial \boldsymbol{r}_i}\right), \tag{32}$$

and $\langle \boldsymbol{x} \rangle$ denotes the average coefficient value, i.e., $\langle \boldsymbol{x} \rangle \triangleq \frac{1}{N}\sum_{i=1}^N x_i$ for $\boldsymbol{x} \in \mathbb{R}^N$. Due to the form of $p_{\mathbf{y}|\mathbf{x}}$ in (15), it can be shown that

$$\boldsymbol{g}_2(\boldsymbol{r}_2, \gamma_2) = (\gamma_2 \boldsymbol{I} + \gamma_w \boldsymbol{A}^{\mathsf{T}} \boldsymbol{A})^{-1} (\gamma_2 \boldsymbol{r}_2 + \gamma_w \boldsymbol{A}^{\mathsf{T}} \boldsymbol{y}) \tag{33}$$

$$\langle \boldsymbol{g}_2'(\boldsymbol{r}_2, \gamma_2) \rangle = \gamma_2 \mathrm{tr}\big\{(\gamma_2 \boldsymbol{I} + \gamma_w \boldsymbol{A}^{\mathsf{T}} \boldsymbol{A})^{-1}\big\}/N. \tag{34}$$

Meanwhile, the form of $\boldsymbol{g}_1(\cdot)$ depends on $p_{\mathbf{x}}$ through (30).

Based on the description above, one might wonder whether the EC stationary point $\widehat{\boldsymbol{x}} = \mathbb{E}[\boldsymbol{x}|q_1] = \mathbb{E}[\boldsymbol{x}|q_2] = \mathbb{E}[\boldsymbol{x}|q_3]$ is a good approximation of the true conditional mean $\mathbb{E}[\boldsymbol{x}|\boldsymbol{y}]$, and additionally one might question whether Algorithm 1 converges to this $\widehat{\boldsymbol{x}}$. Both of these concerns were resolved in the VAMP paper [45]. In particular, [45] showed that, when

[5]We minimize over q_1 and q_2 because $D_{\mathsf{KL}}(q_1 \| p_{\mathbf{x}})$ and $D_{\mathsf{KL}}(q_2 \| p_{\mathbf{y}|\mathbf{x}})$ are convex, while we maximize over q_3 because $H(q_3)$ is concave.

[6]In [47], different $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ were proposed so that the EP algorithm accomplishes joint MAP estimation of $\boldsymbol{x}$ from $\boldsymbol{y}$, i.e., $\widehat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{y})$.

$\boldsymbol{A}$ is right rotationally invariant and asymptotically large, the per-iteration behavior of Algorithm 1 with $\boldsymbol{g}_2(\cdot)$ from (33) and Lipschitz[7] $\boldsymbol{g}_1(\cdot)$ is exactly predicted by a scalar state evolution. Furthermore, in the case where $\boldsymbol{g}_1(\cdot)$ is matched to generating $p_{\boldsymbol{X}}$ from (18) as in (30), and where $\boldsymbol{g}_2(\cdot)$ uses the true AWGN precision $\gamma_w < \infty$ as in (31), the MSE of the fixed point $\widehat{\boldsymbol{x}}$ of Algorithm 1 was shown in [45] to match the MMSE predicted by the replica method [56]. This replica prediction is conjectured to be correct [57], in which case the $\widehat{\boldsymbol{x}}$ generated by Algorithm 1 under (30) and (33) will be MMSE for infinitely large, right-rotationally invariant $\boldsymbol{A}$ when the state evolution has unique fixed points. Note that, for infinitely large i.i.d. $\boldsymbol{A}$, the replica prediction has been proven to be correct [58], [59].

In the sequel, we will refer to Algorithm 1 with generic Lipschitz $\boldsymbol{g}_1(\cdot)$ as the VAMP algorithm, noting that it coincides with EP in the special case of Bayesian $\boldsymbol{g}_1(\cdot)$ from (30). VAMP is more general than EP because it can be used with denoisers $\boldsymbol{g}_1(\cdot)$ that have no probabilistic interpretation and still lead to precisely predictable behavior under infinitely large, right-rotationally $\boldsymbol{A}$ [45], [50]. We note that, when VAMP is applied to the MMV model (9a), a separate copy of $\{\boldsymbol{r}_1, \gamma_1, \boldsymbol{r}_2, \gamma_2, \widehat{\boldsymbol{x}}, \eta\}$ must be tracked for each column of $\boldsymbol{Y}$.

B. Background on Expectation Maximization

We now return to the case where $\boldsymbol{\Theta}$ is unknown and the goal is to compute its ML estimate, $\widehat{\boldsymbol{\Theta}}_{\mathsf{ML}}$. From (10) and (12), we have

$$\widehat{\boldsymbol{\Theta}}_{\mathsf{ML}} = \arg\min_{\boldsymbol{\Theta}} - \ln \int p_{\boldsymbol{X}}(\boldsymbol{X};\boldsymbol{\Theta}) p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{\Theta})\,\mathrm{d}\boldsymbol{X}, \quad (35)$$

but (35) is impractical to optimize directly due to the high dimensional integral.

Expectation-maximization (EM) [60] is a well known iterative approach to ML that alternates between i) minimizing an upper-bound of the negative log-likelihood and ii) tightening the upper-bound. The EM algorithm is usually written as

$$Q(\boldsymbol{\Theta}; \widehat{\boldsymbol{\Theta}}^t) \triangleq - \mathbb{E}\left[\ln p_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{X},\boldsymbol{Y};\boldsymbol{\Theta}) \,\middle|\, \boldsymbol{Y}; \widehat{\boldsymbol{\Theta}}^t\right] \quad (36a)$$

$$\widehat{\boldsymbol{\Theta}}^{t+1} = \arg\min_{\boldsymbol{\Theta}} Q(\boldsymbol{\Theta}; \widehat{\boldsymbol{\Theta}}^t). \quad (36b)$$

Letting $q^t = p_{\boldsymbol{X}|\boldsymbol{Y}}(\cdot|\boldsymbol{Y}; \widehat{\boldsymbol{\Theta}}^t)$, we can write

$$\begin{aligned} Q(\boldsymbol{\Theta}; \widehat{\boldsymbol{\Theta}}^t) &= - \mathbb{E}\left[\ln p_{\boldsymbol{X}}(\boldsymbol{X};\boldsymbol{\Theta}) \,\middle|\, \boldsymbol{Y}; \widehat{\boldsymbol{\Theta}}^t\right] \\ &\quad - \mathbb{E}\left[\ln p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{\Theta}) \,\middle|\, \boldsymbol{Y}; \widehat{\boldsymbol{\Theta}}^t\right] && (37a) \\ &= - \mathbb{E}\left[\ln p_{\boldsymbol{X}}(\boldsymbol{X};\boldsymbol{\Theta}) \,\middle|\, q^t\right] \\ &\quad - \mathbb{E}\left[\ln p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{\Theta}) \,\middle|\, q^t\right] && (37b) \\ &= J(q^t, q^t, q^t; \boldsymbol{\Theta}) + \text{const.} && (37c) \end{aligned}$$

where J, also known as the Gibbs free energy, is defined as

$$\begin{aligned} J(q_1, q_2, q_3; \boldsymbol{\Theta}) &\triangleq D_{\mathsf{KL}}(q_1 \| p_{\boldsymbol{X}}(\cdot, \boldsymbol{\Theta})) \\ &\quad + D_{\mathsf{KL}}(q_2 \| p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{Y}|\cdot; \boldsymbol{\Theta})) + H(q_3). \quad (38) \end{aligned}$$

[7]While the original VAMP paper [45] focused on separable Lipschitz $\boldsymbol{g}_1(\cdot)$, [50] extended the results to non-separable Lipschitz $\boldsymbol{g}_1(\cdot)$.

Thus, (36) can also be written as [61]

$$q^t = p_{\boldsymbol{X}|\boldsymbol{Y}}(\cdot|\boldsymbol{Y}; \widehat{\boldsymbol{\Theta}}^t) \quad (39a)$$

$$\widehat{\boldsymbol{\Theta}}^{t+1} = \arg\min_{\boldsymbol{\Theta}} J(q^t, q^t, q^t; \boldsymbol{\Theta}). \quad (39b)$$

Note that $J(q^t, q^t, q^t; \boldsymbol{\Theta})$ is an upper bound on $-\ln p_{\boldsymbol{Y}}(\boldsymbol{Y}; \boldsymbol{\Theta})$ for *any* q^t since

$$J(q^t, q^t, q^t; \boldsymbol{\Theta}) = - \ln p_{\boldsymbol{Y}}(\boldsymbol{Y};\boldsymbol{\Theta}) + D_{\mathsf{KL}}(q^t \,\|\, p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{Y}|\cdot;\boldsymbol{\Theta})), \quad (40)$$

where $D_{\mathsf{KL}} \geq 0$ by construction. Thus, while the specific choice of q^t in (39a) yields a tight upper bound in that

$$J(q^t, q^t, q^t; \widehat{\boldsymbol{\Theta}}^t) = - \ln p_{\boldsymbol{Y}}(\boldsymbol{Y}; \widehat{\boldsymbol{\Theta}}^t), \quad (41)$$

other choices of bounding q^t can also be used in EM [61].

IV. Bilinear Adaptive VAMP

We now propose an algorithm that approximates the quantities in (10)–(11), i.e., the ML estimate of $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_A, \boldsymbol{\theta}_x, \gamma_w\}$ and the MMSE estimate of $\boldsymbol{X}$ under the statistical model (9). We start by developing Bilinear EM-VAMP and then add "variance auto-tuning" to obtain Bilinear Adaptive VAMP (BAd-VAMP).

A. Bilinear EM-VAMP

From the descriptions of VAMP and EM in Section III, we see that they both minimize the same Gibbs free energy cost $J(q_1, q_2, q_3; \boldsymbol{\Theta})$ from (38), but w.r.t. different variables; VAMP minimizes J w.r.t. the moment-constrained beliefs $\{q_1, q_2, q_3\}$ for a given $\boldsymbol{\Theta}$, while EM minimizes J w.r.t. $\boldsymbol{\Theta}$ for a given $\{q_1, q_2, q_3\}$. As a result, the two approaches can be straightforwardly merged for *joint* estimation of $\{q_1, q_2, q_3\}$ and $\boldsymbol{\Theta}$. In doing so, the goal is to solve the optimization problem

$$\arg\min_{\boldsymbol{\Theta}, q_1} \min_{q_2} \max_{q_3} J(q_1, q_2, q_3; \boldsymbol{\Theta}) \quad (42a)$$

$$\text{s.t.} \quad \mathbb{E}[\boldsymbol{x}|q_1] = \mathbb{E}[\boldsymbol{x}|q_2] = \mathbb{E}[\boldsymbol{x}|q_3] \quad (42b)$$

$$\operatorname{tr}\{\operatorname{Cov}[\boldsymbol{x}|q_1]\} = \operatorname{tr}\{\operatorname{Cov}[\boldsymbol{x}|q_2]\} = \operatorname{tr}\{\operatorname{Cov}[\boldsymbol{x}|q_3]\}, \quad (42c)$$

and the proposed methodology is to "interleave" the VAMP and EM algorithms, as specified in Algorithm 2. There, the estimation functions $\boldsymbol{g}_1$ in lines 3-4 and $\boldsymbol{g}_{2,l}$ in lines 9-10 are defined as

$$\boldsymbol{g}_1(\boldsymbol{r}_{1,l}, \gamma_{1,l}; \boldsymbol{\theta}_x) \triangleq \frac{\int \boldsymbol{x}\, p_{\boldsymbol{x}}(\boldsymbol{x};\boldsymbol{\theta}_x) \mathcal{N}(\boldsymbol{x}; \boldsymbol{r}_{1,l}, \boldsymbol{I}/\gamma_{1,l})\,\mathrm{d}\boldsymbol{x}}{\int p_{\boldsymbol{x}}(\boldsymbol{x};\boldsymbol{\theta}_x) \mathcal{N}(\boldsymbol{x}; \boldsymbol{r}_{1,l}, \boldsymbol{I}/\gamma_{1,l})\,\mathrm{d}\boldsymbol{x}} \quad (43)$$

$$\boldsymbol{g}_{2,l}(\boldsymbol{r}_{2,l}, \gamma_{2,l}; \boldsymbol{\theta}_A, \gamma_w) \triangleq \frac{\int \boldsymbol{x}\, p_{\boldsymbol{y}|\boldsymbol{x}}(\boldsymbol{y}_l|\boldsymbol{x};\boldsymbol{\theta}_A, \gamma_w) \mathcal{N}(\boldsymbol{x}; \boldsymbol{r}_{2,l}, \boldsymbol{I}/\gamma_{2,l})\,\mathrm{d}\boldsymbol{x}}{\int p_{\boldsymbol{y}|\boldsymbol{x}}(\boldsymbol{y}_l|\boldsymbol{x};\boldsymbol{\theta}_A, \gamma_w) \mathcal{N}(\boldsymbol{x}; \boldsymbol{r}_{2,l}, \boldsymbol{I}/\gamma_{2,l})\,\mathrm{d}\boldsymbol{x}}, \quad (44)$$

The other lines in Algorithm 2 will be detailed in Section IV-C.

B. Bilinear Adaptive VAMP

The VAMP state-evolution from [45, Eq. (34), (35)] shows that when i) $\boldsymbol{A}(\boldsymbol{\theta}_A^t)$ is infinitely large and right-rotationally invariant and ii) the estimation functions $\boldsymbol{g}_1$ and $\boldsymbol{g}_{2,l}$ are "matched"

Algorithm 2: Bilinear EM-VAMP.

```
1:  initialize:
        $\forall l : \boldsymbol{r}_{1,l}^0, \gamma_{1,l}^0, \boldsymbol{\theta}_x^0, \boldsymbol{\theta}_A^0, \gamma_w^0$
2:  for $t = 0, \dots, T_{\max}$ do
3:      $\forall l : \boldsymbol{x}_{1,l}^t = \boldsymbol{g}_1(\boldsymbol{r}_{1,l}^t, \gamma_{1,l}^t; \boldsymbol{\theta}_x^t)$
4:      $\forall l : 1/\eta_{1,l}^t = \langle \boldsymbol{g}_1'(\boldsymbol{r}_{1,l}^t, \gamma_{1,l}^t; \boldsymbol{\theta}_x^t) \rangle / \gamma_{1,l}^t$
5:      $q_1^t(\boldsymbol{X}) \propto \prod_{l=1}^L p_{\boldsymbol{x}}(\boldsymbol{x}_l; \boldsymbol{\theta}_x^t) e^{-\frac{1}{2}\gamma_{1,l}^t \|\boldsymbol{x}_l - \boldsymbol{r}_{1,l}^t\|^2}$
6:      $\boldsymbol{\theta}_x^{t+1} = \arg\max_{\boldsymbol{\theta}_x} \mathbb{E}[\ln p_{\boldsymbol{X}}(\boldsymbol{X}; \boldsymbol{\theta}_x) | q_1^t]$
7:      $\forall l : \gamma_{2,l}^t = \eta_{1,l}^t - \gamma_{1,l}^t$
8:      $\forall l : \boldsymbol{r}_{2,l}^t = (\eta_{1,l}^t \boldsymbol{x}_{1,l}^t - \gamma_{1,l}^t \boldsymbol{r}_{1,l}^t)/\gamma_{2,l}^t$
9:      $\forall l : \boldsymbol{x}_{2,l}^t = \boldsymbol{g}_{2,l}(\boldsymbol{r}_{2,l}^t, \gamma_{2,l}^t; \boldsymbol{\theta}_A^t, \gamma_w^t)$
10:     $\forall l : 1/\eta_{2,l}^t = \langle \boldsymbol{g}_{2,l}'(\boldsymbol{r}_{2,l}^t, \gamma_{2,l}^t; \boldsymbol{\theta}_A^t, \gamma_w^t) \rangle / \gamma_{2,l}^t$
11:     $q_2^t(\boldsymbol{X}) \propto \prod_{l=1}^L p_{\boldsymbol{y}|\boldsymbol{x}}(\boldsymbol{y}_l | \boldsymbol{x}_l; \boldsymbol{\theta}_A^t, \gamma_w^t) e^{-\frac{1}{2}\gamma_{2,l}^t \|\boldsymbol{x}_l - \boldsymbol{r}_{2,l}^t\|^2}$
12:     $\boldsymbol{\theta}_A^{t+1} = \arg\max_{\boldsymbol{\theta}_A} \mathbb{E}[\ln p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{Y}|\boldsymbol{X}; \boldsymbol{\theta}_A, \gamma_w^t) | \boldsymbol{Y}, q_2^t]$
13:     $\gamma_w^{t+1} = \arg\max_{\gamma_w} \mathbb{E}[\ln p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{Y}|\boldsymbol{X}; \boldsymbol{\theta}_A^{t+1}, \gamma_w) | \boldsymbol{Y}, q_2^t]$
14:     $\forall l : \gamma_{1,l}^{t+1} = \eta_{2,l}^t - \gamma_{2,l}^t$
15:     $\forall l : \boldsymbol{r}_{1,l}^{t+1} = (\eta_{2,l}^t \boldsymbol{x}_{2,l}^t - \gamma_{2,l}^t \boldsymbol{r}_{2,l}^t)/\gamma_{1,l}^{t+1}$
16: end for
```

(i.e., MMSE) for the statistical model generating $(\boldsymbol{X}, \boldsymbol{Y})$, the VAMP quantities $\{(\boldsymbol{r}_{i,l}^t, \gamma_{i,l}^t)\}_{l=1}^L$ for $i = 1, 2$ obey

$$\boldsymbol{r}_{1,l}^t = \boldsymbol{x}_l + \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}/\gamma_{1,l}^t) \; \forall l \tag{45a}$$

$$\boldsymbol{x}_l = \boldsymbol{r}_{2,l}^t + \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}/\gamma_{2,l}^t) \; \forall l, \tag{45b}$$

where $\boldsymbol{x}_l$ is the lth column of the true signal realization $\boldsymbol{X}$ that we aim to recover. That is, $\boldsymbol{r}_{1,l}^t$ is an AWGN-corrupted version of the true signal $\boldsymbol{x}_l$ with *known* AWGN precision $\gamma_{1,l}^t$, and the true signal $\boldsymbol{x}_l$ is an AWGN-corrupted version of $\boldsymbol{r}_{2,l}^t$ with *known* AWGN precision $\gamma_{2,l}^t$. In the context of EM-VAMP under (9), this "matched" condition requires that $\boldsymbol{\theta}_A^t$, $\boldsymbol{\theta}_x^t$, and γ_w^t are all perfect estimates. When $\boldsymbol{\theta}_A^t$, $\boldsymbol{\theta}_x^t$, or γ_w^t are *not* perfect, so that $\boldsymbol{g}_1$ and $\boldsymbol{g}_{2,l}$ are mismatched, the VAMP state-evolution shows that $\boldsymbol{r}_{1,l}^t$ is still an AWGN corrupted version of $\boldsymbol{x}_l$, but with an AWGN precision *different* than $\gamma_{1,l}^t$. The impact on EM-VAMP is the following. While the algorithm is trying to learn $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_A, \boldsymbol{\theta}_x, \gamma_w\}$, the value of $\gamma_{i,l}^t$ does *not* correctly characterize the noise precision in $\boldsymbol{r}_{i,l}^t$. As a result, the beliefs q_1^t and q_2^t in lines 5 and 11 of Algorithm 2 become mismatched, which compromises the EM updates of $\boldsymbol{\Theta}^t$.

To remedy this situation, it was proposed in [62] (in the context of EM-GAMP [63]) to explicitly estimate the precision of the AWGN corruption on $\boldsymbol{r}_{1,l}^t$ and $\boldsymbol{r}_{2,l}^t$ and use it in place of the AMP-supplied estimates $\gamma_{1,l}^t$ and $\gamma_{2,l}^t$. This approach was coined "Adaptive" GAMP in [62] and later extended to (linear) Adaptive VAMP in [51].

For Bilinear Adaptive VAMP, the first goal is to replace the estimation of $\boldsymbol{\theta}_x^t$ in line 6 of Algorithm 2 with the joint ML estimation

$$(\boldsymbol{\theta}_x^t, \boldsymbol{\gamma}_1^t) = \arg\max_{\boldsymbol{\theta}_x, \boldsymbol{\gamma}_1} p(\boldsymbol{R}_1^t; \boldsymbol{\gamma}_1, \boldsymbol{\theta}_x) \tag{46}$$

under the statistical model

$$\boldsymbol{r}_{1,l}^t = \boldsymbol{x}_l + \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}/\gamma_{1,l}) \; \forall l, \qquad \boldsymbol{x}_l \sim p_{\boldsymbol{x}}(\cdot; \boldsymbol{\theta}_x) \; \forall l, \tag{47}$$

with independence across $l = 1, \dots, L$. For this subproblem, we propose to use (inner) EM iterations indexed by τ, i.e.,

$$\begin{aligned}
(\boldsymbol{\theta}_x^{\tau+1}, \boldsymbol{\gamma}_1^{\tau+1}) &\\
= \arg\max_{\boldsymbol{\theta}_x, \boldsymbol{\gamma}_1} &\mathbb{E}\left[\ln p(\boldsymbol{X}, \boldsymbol{R}_1^t; \boldsymbol{\gamma}_1, \boldsymbol{\theta}_x) \,\middle|\, \boldsymbol{R}_1^t; \boldsymbol{\gamma}_1^\tau, \boldsymbol{\theta}_x^\tau\right] \qquad (48)\\
= \arg\max_{\boldsymbol{\theta}_x, \boldsymbol{\gamma}_1} &\Big\{ \mathbb{E}\left[\ln p(\boldsymbol{X}; \boldsymbol{\theta}_x) \,\middle|\, \boldsymbol{R}_1^t; \boldsymbol{\gamma}_1^\tau, \boldsymbol{\theta}_x^\tau\right] \\
&+ \mathbb{E}\left[\ln p(\boldsymbol{R}_1^t | \boldsymbol{X}; \boldsymbol{\gamma}_1) \,\middle|\, \boldsymbol{R}_1^t; \boldsymbol{\gamma}_1^\tau, \boldsymbol{\theta}_x^\tau\right] \Big\}. \qquad (49)
\end{aligned}$$

The previous optimization decouples into

$$\boldsymbol{\theta}_x^{\tau+1} = \arg\max_{\boldsymbol{\theta}_x} \mathbb{E}\left[\ln p(\boldsymbol{X}; \boldsymbol{\theta}_x) \,\middle|\, \boldsymbol{R}_1^t; \boldsymbol{\gamma}_1^\tau, \boldsymbol{\theta}_x^\tau\right] \tag{50}$$

and

$$\begin{aligned}
\boldsymbol{\gamma}_1^{\tau+1} &= \arg\max_{\boldsymbol{\gamma}_1} \mathbb{E}\left[\ln p(\boldsymbol{R}_1^t | \boldsymbol{X}; \boldsymbol{\gamma}_1) \,\middle|\, \boldsymbol{R}_1^t; \boldsymbol{\gamma}_1^\tau, \boldsymbol{\theta}_x^\tau\right] \qquad (51)\\
&= \arg\max_{\boldsymbol{\gamma}_1} \sum_{l=1}^L \mathbb{E}\left[\ln p(\boldsymbol{r}_{1,l}^t | \boldsymbol{x}_l; \gamma_{1,l}) \,\middle|\, \boldsymbol{r}_{1,l}^t; \gamma_{1,l}^\tau, \boldsymbol{\theta}_x^\tau\right], \qquad (52)
\end{aligned}$$

where the latter optimization decouples further into

$$\begin{aligned}
\gamma_{1,l}^{\tau+1} &= \arg\max_{\gamma_{1,l}} \bigg\{ \frac{N}{2} \ln \gamma_{1,l} \\
&\qquad - \frac{\gamma_{1,l}}{2} \mathbb{E}\left[\|\boldsymbol{x}_l - \boldsymbol{r}_{1,l}^t\|_2^2 \,\middle|\, \boldsymbol{r}_{1,l}^t; \gamma_{1,l}^\tau, \boldsymbol{\theta}_x^\tau\right] \bigg\} \qquad (53)\\
&= N \left\{ \mathbb{E}\left[\|\boldsymbol{x}_l - \boldsymbol{r}_{1,l}^t\|_2^2 \,\middle|\, \boldsymbol{r}_{1,l}^t; \gamma_{1,l}^\tau, \boldsymbol{\theta}_x^\tau\right] \right\}^{-1} \qquad (54)\\
&= \left\{ \frac{1}{N} \sum_{n=1}^N \mathbb{E}\left[(x_{nl} - r_{1,nl}^t)^2 \,\middle|\, \boldsymbol{r}_{1,l}^t; \gamma_{1,l}^\tau, \boldsymbol{\theta}_x^\tau\right] \right\}^{-1} \qquad (55)\\
&= \left\{ \frac{1}{N} \|\boldsymbol{x}_{1,l}^\tau - \boldsymbol{r}_{1,l}^t\|^2 + \frac{1}{\eta_{1,l}^\tau} \right\}^{-1}, \qquad (56)
\end{aligned}$$

for $l = 1, \dots, L$ and

$$\begin{aligned}
\boldsymbol{x}_{1,l}^\tau &\triangleq \mathbb{E}\left[\boldsymbol{x}_l \middle| \boldsymbol{r}_{1,l}^t; \gamma_{1,l}^\tau, \boldsymbol{\theta}_x^\tau\right] \qquad (57)\\
&= \boldsymbol{g}_1(\boldsymbol{r}_{1,l}^t, \gamma_{1,l}^\tau; \boldsymbol{\theta}_x^\tau) \qquad (58)\\
1/\eta_{1,l}^\tau &\triangleq \operatorname{tr}\left\{ \operatorname{Cov}\left[\boldsymbol{x}_l \middle| \boldsymbol{r}_{1,l}^t; \gamma_{1,l}^\tau, \boldsymbol{\theta}_x^\tau\right] \right\}/N \qquad (59)\\
&= \langle \boldsymbol{g}_1'(\boldsymbol{r}_{1,l}^t, \gamma_{1,l}^\tau; \boldsymbol{\theta}_x^\tau) \rangle / \gamma_{1,l}^\tau. \qquad (60)
\end{aligned}$$

Above, we detailed the re-estimation of $\boldsymbol{\gamma}_1^t$. A similar procedure can be used for re-estimation of $\boldsymbol{\gamma}_2^t$. The resulting Bilinear Adaptive VAMP (BAd-VAMP) is summarized in Algorithm 3 using $\tau_{1,\max}$ EM iterations for the first inner loop and $\tau_{2,\max}$ EM iterations for the second inner loop. To avoid the complications of a dual-index notation (i.e., t and τ), we use only the single index t in Algorithm 3 and over-write the quantities in each inner loop. Note that, when $\tau_{1,\max} = \tau_{2,\max} = 0$, BAd-VAMP (i.e., Algorithm 3) reduces to bilinear EM-VAMP (i.e., Algorithm 2).

C. Algorithm Details

We now provide additional details on the steps in Algorithm 3.

Algorithm 3: Bilinear Adaptive VAMP.

```
1:  initialize:
        $\forall l: \boldsymbol{r}_{1,l}^0, \gamma_{1,l}^0, \boldsymbol{\theta}_x^0, \boldsymbol{\theta}_A^0, \gamma_w^0$
2:  for $t = 0, \ldots, T_{\max}$ do
3:      for $\tau = 0, \ldots, \tau_{1,\max}$ do
4:          $\forall l: \boldsymbol{x}_{1,l}^t \leftarrow \boldsymbol{g}_1(\boldsymbol{r}_{1,l}^t, \gamma_{1,l}^t; \boldsymbol{\theta}_x^t)$
5:          $\forall l: 1/\eta_{1,l}^t \leftarrow \langle \boldsymbol{g}_1'(\boldsymbol{r}_{1,l}^t, \gamma_{1,l}^t; \boldsymbol{\theta}_x^t)\rangle / \gamma_{1,l}^t$
6:          $\forall l: 1/\gamma_{1,l}^t \leftarrow \frac{1}{N}\|\boldsymbol{x}_{1,l}^t - \boldsymbol{r}_{1,l}^t\|^2 + 1/\eta_{1,l}^t$
7:          $q_1^t(\boldsymbol{X}) \propto \prod_{l=1}^L p_{\mathbf{x}}(\boldsymbol{x}_l; \boldsymbol{\theta}_x^t) e^{-\frac{1}{2}\gamma_{1,l}^t \|\boldsymbol{x}_l - \boldsymbol{r}_{1,l}^t\|^2}$
8:          $\boldsymbol{\theta}_x^t \leftarrow \arg\max_{\boldsymbol{\theta}_x} \mathbb{E}[\ln p_{\mathbf{X}}(\boldsymbol{X}; \boldsymbol{\theta}_x) | q_1^t]$
9:      end for
10:     $\boldsymbol{\theta}_x^{t+1} = \boldsymbol{\theta}_x^t$
11:     $\forall l: \gamma_{2,l}^t = \eta_{1,l}^t - \gamma_{1,l}^t$
12:     $\forall l: \boldsymbol{r}_{2,l}^t = (\eta_{1,l}^t \boldsymbol{x}_{1,l}^t - \gamma_{1,l}^t \boldsymbol{r}_{1,l}^t)/\gamma_{2,l}^t$
13:     for $\tau = 0, \ldots, \tau_{2,\max}$ do
14:         $\forall l: \boldsymbol{x}_{2,l}^t \leftarrow \boldsymbol{g}_{2,l}(\boldsymbol{r}_{2,l}^t, \gamma_{2,l}^t; \boldsymbol{\theta}_A^t, \gamma_w^t)$
15:         $\forall l: 1/\eta_{2,l}^t \leftarrow \langle \boldsymbol{g}_{2,l}'(\boldsymbol{r}_{2,l}^t, \gamma_{2,l}^t; \boldsymbol{\theta}_A^t, \gamma_w^t)\rangle / \gamma_{2,l}^t$
16:         $\forall l: 1/\gamma_{2,l}^t \leftarrow \frac{1}{N}\|\boldsymbol{x}_{2,l}^t - \boldsymbol{r}_{2,l}^t\|^2 + 1/\eta_{2,l}^t$
17:         $q_2^t(\boldsymbol{X}) \propto \prod_l p_{\mathbf{Y}|\mathbf{X}}(\boldsymbol{y}_l|\boldsymbol{x}_l; \boldsymbol{\theta}_A^t, \gamma_w^t) e^{-\frac{1}{2}\gamma_{2,l}^t \|\boldsymbol{x}_l - \boldsymbol{r}_{2,l}^t\|^2}$
18:         $\boldsymbol{\theta}_A^t \leftarrow \arg\max_{\boldsymbol{\theta}_A} \mathbb{E}[\ln p_{\mathbf{Y}|\mathbf{X}}(\boldsymbol{Y}|\boldsymbol{X}; \boldsymbol{\theta}_A, \gamma_w^t) | \boldsymbol{Y}, q_2^t]$
19:         $\gamma_w^t \leftarrow \arg\max_{\gamma_w} \mathbb{E}[\ln p_{\mathbf{Y}|\mathbf{X}}(\boldsymbol{Y}|\boldsymbol{X}; \boldsymbol{\theta}_A^t, \gamma_w) | \boldsymbol{Y}, q_2^t]$
20:     end for
21:     $\boldsymbol{\theta}_A^{t+1} = \boldsymbol{\theta}_A^t$
22:     $\gamma_w^{t+1} = \gamma_w^t$
23:     $\forall l: \gamma_{1,l}^{t+1} = \eta_{2,l}^t - \gamma_{2,l}^t$
24:     $\forall l: \boldsymbol{r}_{1,l}^{t+1} = (\eta_{2,l}^t \boldsymbol{x}_{2,l}^t - \gamma_{2,l}^t \boldsymbol{r}_{2,l}^t)/\gamma_{1,l}^{t+1}$
25: end for
```

1) Estimating $\boldsymbol{X}$: Recalling the definition of $\boldsymbol{g}_{2,l}(\cdot)$ in (44), the form of $p_{\mathbf{Y}|\mathbf{X}}$ in (15) implies that

$$\boldsymbol{g}_{2,l}(\boldsymbol{r}_{2,l}^t, \gamma_{2,l}^t; \boldsymbol{\theta}_A^t, \gamma_w^t) = \boldsymbol{C}_l^t\big(\gamma_{2,l}^t \boldsymbol{r}_{2,l}^t + \gamma_w^t \boldsymbol{A}(\boldsymbol{\theta}_A^t)^{\mathsf{T}} \boldsymbol{y}_l\big) \quad (61)$$

$$\langle \boldsymbol{g}_{2,l}'(\boldsymbol{r}_{2,l}^t, \gamma_{2,l}^t; \boldsymbol{\theta}_A^t, \gamma_w^t)\rangle = \gamma_{2,l}^t \operatorname{tr}\{\boldsymbol{C}_l^t\}/N \quad (62)$$

for

$$\boldsymbol{C}_l^t \triangleq \big(\gamma_{2,l}^t \boldsymbol{I}_N + \gamma_w^t \boldsymbol{A}(\boldsymbol{\theta}_A^t)^{\mathsf{T}} \boldsymbol{A}(\boldsymbol{\theta}_A^t)\big)^{-1}. \quad (63)$$

To avoid computing a separate matrix inverse (63) for each $l = 1, \ldots, L$, one could instead compute the eigenvalue decomposition

$$\boldsymbol{A}(\boldsymbol{\theta}_A^t)^{\mathsf{T}} \boldsymbol{A}(\boldsymbol{\theta}_A^t) = \boldsymbol{U}^t \operatorname{Diag}(\boldsymbol{s}^t) \boldsymbol{U}^{t\mathsf{T}}, \quad (64)$$

and then leverage the fact that

$$\boldsymbol{C}_l^t = \boldsymbol{U}^t \operatorname{Diag}(\gamma_{2,l}^t \boldsymbol{1} + \gamma_w^t \boldsymbol{s}^t)^{-1} \boldsymbol{U}^{t\mathsf{T}}, \quad (65)$$

which reduces to the inversion of a diagonal matrix for each $l = 1, \ldots, L$.

2) Learning $\boldsymbol{\theta}_A$: We now provide details on the update of $\boldsymbol{\theta}_A$ and γ_w in lines 18-19 of Algorithm 3. Given the form of $p_{\mathbf{Y}|\mathbf{X}}$ in (14)–(15), we have that

$$\begin{aligned}&\ln p_{\mathbf{Y}|\mathbf{X}}(\boldsymbol{Y}|\boldsymbol{X}; \boldsymbol{\theta}_A, \gamma_w)\\ &\quad = \tfrac{ML}{2}\ln\gamma_w - \tfrac{\gamma_w}{2}\|\boldsymbol{Y} - \boldsymbol{A}(\boldsymbol{\theta}_A)\boldsymbol{X}\|_F^2 + \text{const} \quad (66)\\ &\quad = \tfrac{ML}{2}\ln\gamma_w - \tfrac{\gamma_w}{2}\big(\operatorname{tr}\{\boldsymbol{Y}\boldsymbol{Y}^{\mathsf{T}}\} - 2\operatorname{tr}\{\boldsymbol{A}(\boldsymbol{\theta}_A)\boldsymbol{X}\boldsymbol{Y}^{\mathsf{T}}\}\\ &\qquad + \operatorname{tr}\{\boldsymbol{A}(\boldsymbol{\theta}_A)\boldsymbol{X}\boldsymbol{X}^{\mathsf{T}}\boldsymbol{A}(\boldsymbol{\theta}_A)^{\mathsf{T}}\}\big) + \text{const}. \quad (67)\end{aligned}$$

Since

$$\mathbb{E}\left[\boldsymbol{X}|q_2^t\right] = \boldsymbol{X}_2^t \quad (68)$$

$$\mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^{\mathsf{T}}|q_2^t\right] = \sum_{l=1}^L \mathbb{E}\left[\boldsymbol{x}_l\boldsymbol{x}_l^{\mathsf{T}}|q_{2,l}^t\right] = \boldsymbol{X}_2^t\boldsymbol{X}_2^{t\mathsf{T}} + \underbrace{\sum_{l=1}^L \boldsymbol{C}_l^t}_{\triangleq\, \boldsymbol{C}^t}, \quad (69)$$

we have that

$$\begin{aligned}&\mathbb{E}\left[\ln p_{\mathbf{Y}|\mathbf{X}}(\boldsymbol{Y}|\boldsymbol{X}; \boldsymbol{\theta}_A, \gamma_w)\big|\boldsymbol{Y}, q_2^t\right]\\ &\quad = \frac{ML}{2}\ln\gamma_w - \frac{\gamma_w}{2}\Big(\operatorname{tr}\{\boldsymbol{Y}\boldsymbol{Y}^{\mathsf{T}}\} - 2\operatorname{tr}\{\boldsymbol{A}(\boldsymbol{\theta}_A)\boldsymbol{X}_2^t\boldsymbol{Y}^{\mathsf{T}}\}\\ &\qquad + \operatorname{tr}\{\boldsymbol{A}(\boldsymbol{\theta}_A)\boldsymbol{X}_2^t\boldsymbol{X}_2^{t\mathsf{T}}\boldsymbol{A}(\boldsymbol{\theta}_A)^{\mathsf{T}}\} + \operatorname{tr}\{\boldsymbol{A}(\boldsymbol{\theta}_A)\boldsymbol{C}^t\boldsymbol{A}(\boldsymbol{\theta}_A)^{\mathsf{T}}\}\Big) \quad (70)\\ &\quad = \frac{ML}{2}\ln\gamma_w - \frac{\gamma_w}{2}\Big(\|\boldsymbol{Y} - \boldsymbol{A}(\boldsymbol{\theta}_A)\boldsymbol{X}_2^t\|_F^2\\ &\qquad + \operatorname{tr}\{\boldsymbol{A}(\boldsymbol{\theta}_A)\boldsymbol{C}^t\boldsymbol{A}(\boldsymbol{\theta}_A)^{\mathsf{T}}\}\Big) + \text{const}. \quad (71)\end{aligned}$$

To maximize (71) over $\boldsymbol{\theta}_A = [\theta_{A,1}, \ldots, \theta_{A,Q}]$ with fixed γ_w, we consider the affine-linear model

$$\boldsymbol{A}(\boldsymbol{\theta}_A) = \boldsymbol{A}_0 + \sum_{i=1}^Q \theta_{A,i}\boldsymbol{A}_i, \quad (72)$$

noting that non-linear models could be handled using similar techniques. Plugging (72) into (70), we get

$$\begin{aligned}&\mathbb{E}\left[\ln p_{\mathbf{Y}|\mathbf{X}}(\boldsymbol{Y}|\boldsymbol{X}; \boldsymbol{\theta}_A, \gamma_w)\big|\boldsymbol{Y}, q_2^t\right]\\ &\quad = \text{const} - \frac{\gamma_w}{2}\sum_{i=1}^Q\sum_{j=1}^Q \theta_{A,i}\operatorname{tr}\{\boldsymbol{A}_i(\boldsymbol{C}^t + \boldsymbol{X}_2^t\boldsymbol{X}_2^{t\mathsf{T}})\boldsymbol{A}_j^{\mathsf{T}}\}\theta_{A,j}\\ &\qquad - \gamma_w\sum_{i=1}^Q \theta_{A,i}\big(\operatorname{tr}\{\boldsymbol{A}_i(\boldsymbol{C}^t + \boldsymbol{X}_2^t\boldsymbol{X}_2^{t\mathsf{T}})\boldsymbol{A}_0^{\mathsf{T}}\} - \operatorname{tr}\{\boldsymbol{A}_i\boldsymbol{X}_2^t\boldsymbol{Y}^{\mathsf{T}}\}\big) \quad (73)\\ &\quad = -\frac{\gamma_w}{2}\Big(\boldsymbol{\theta}_A^{\mathsf{T}}\boldsymbol{H}^t\boldsymbol{\theta}_A - 2\boldsymbol{\theta}_A^{\mathsf{T}}\boldsymbol{\beta}^t\Big) + \text{const} \quad (74)\end{aligned}$$

for

$$\begin{aligned}[\boldsymbol{H}^t]_{ij} &= \operatorname{tr}\{\boldsymbol{A}_i(\boldsymbol{C}^t + \boldsymbol{X}_2^t\boldsymbol{X}_2^{t\mathsf{T}})\boldsymbol{A}_j^{\mathsf{T}}\} \quad (75)\\ &= \operatorname{tr}\{\boldsymbol{A}_j^{\mathsf{T}}\boldsymbol{A}_i(\boldsymbol{C}^t + \boldsymbol{X}_2^t\boldsymbol{X}_2^{t\mathsf{T}})\} \quad (76)\end{aligned}$$

and

$$\begin{aligned}[\boldsymbol{\beta}^t]_i &= \operatorname{tr}\{\boldsymbol{A}_i\boldsymbol{X}_2^t\boldsymbol{Y}^{\mathsf{T}}\} - \operatorname{tr}\{\boldsymbol{A}_i(\boldsymbol{C}^t + \boldsymbol{X}_2^t\boldsymbol{X}_2^{t\mathsf{T}})\boldsymbol{A}_0^{\mathsf{T}}\} \quad (77)\\ &= \operatorname{tr}\{\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{A}_i\boldsymbol{X}_2^t\} - \operatorname{tr}\{\boldsymbol{A}_0^{\mathsf{T}}\boldsymbol{A}_i(\boldsymbol{C}^t + \boldsymbol{X}_2^t\boldsymbol{X}_2^{t\mathsf{T}})\}, \quad (78)\end{aligned}$$

where $\boldsymbol{A}_j^{\mathsf{T}}\boldsymbol{A}_i$ and $\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{A}_i$ can be pre-computed. Zeroing the gradient of (74) w.r.t. $\boldsymbol{\theta}_A$, we find that the maximizer is

$$\boldsymbol{\theta}_A^{t+1} = (\boldsymbol{H}^t)^{-1}\boldsymbol{\beta}^t. \tag{79}$$

A special case of (72) is where $\boldsymbol{A}(\cdot)$ has no structure, i.e.,

$$\boldsymbol{A}(\boldsymbol{\theta}_A) = \sum_{m=1}^{M}\sum_{n=1}^{N}\theta_{A,m,n}\boldsymbol{e}_m\boldsymbol{e}_n^{\mathsf{T}}. \tag{80}$$

where $\boldsymbol{e}_m$ denotes the mth standard basis vector. In this case, it can be shown that

$$\boldsymbol{A}(\boldsymbol{\theta}_A^{t+1}) = \boldsymbol{Y}\boldsymbol{X}_2^{t\,\mathsf{T}}\big(\boldsymbol{C}^t + \boldsymbol{X}_2^t\boldsymbol{X}_2^{t\,\mathsf{T}}\big)^{-1}. \tag{81}$$

3) Learning γ_w: To maximize (71) over γ_w with fixed $\boldsymbol{\theta}_A = \boldsymbol{\theta}_A^{t+1}$, we search for the values of γ_w that zero the derivative of (71). The unique solution is straightforwardly shown to be

$$1/\gamma_w^{t+1} = \frac{1}{ML}\Big(\|\boldsymbol{Y} - \boldsymbol{A}(\boldsymbol{\theta}_A^{t+1})\boldsymbol{X}_2^t\|_F^2 + \operatorname{tr}\big\{\boldsymbol{A}(\boldsymbol{\theta}_A^{t+1})\boldsymbol{C}^t\boldsymbol{A}(\boldsymbol{\theta}_A^{t+1})^{\mathsf{T}}\big\}\Big). \tag{82}$$

4) Summary: In Algorithm 4, BAd-VAMP is rewritten with detailed expressions for the updates of $\boldsymbol{x}_{2,l}^t$, $\eta_{2,l}^t$, $\boldsymbol{\theta}_A^t$, and γ_w^t.

D. Algorithm Enhancements

We now propose several enhancements to the BAd-VAMP algorithm presented in Algorithm 3 and detailed in Algorithm 4.

1) Damping: For fixed $\boldsymbol{\Theta}^t$ and infinitely large right-rotationally invariant $\boldsymbol{A}(\boldsymbol{\theta}_A^t)$, the state-evolution of VAMP guarantees its convergence. But when $\boldsymbol{A}(\boldsymbol{\theta}_A^t)$ deviates from this assumption, damping the VAMP iterations can help maintain convergence [45]. With damping, lines 25-26 of Algorithm 4 (or lines 23-24 of Algorithm 3) would be replaced by

$$\gamma_{1,l}^{t+1} = (1-\zeta)\gamma_{1,l}^t + \zeta(\eta_{2,l}^t - \gamma_{2,l}^t) \tag{83}$$

$$\boldsymbol{r}_{1,l}^{t+1} = (1-\zeta)\boldsymbol{r}_{1,l}^t + \zeta(\eta_{2,l}^t\boldsymbol{x}_{2,l}^t - \gamma_{2,l}^t\boldsymbol{r}_{2,l}^t)/(\eta_{2,l}^t - \gamma_{2,l}^t) \tag{84}$$

for some $\zeta \in (0,1)$. The case $\zeta = 1$ corresponds to no damping.

2) Negative Precisions: Sometimes the precisions $\{\gamma_{1,l}, \gamma_{2,l}\}_l$ can be negative. We suggest to restrict the precisions to the interval $[\gamma_{\min}, \infty)$, for very small $\gamma_{\min} > 0$, in lines 11 and 25 of Algorithm 4 (or lines 11 and 23 of Algorithm 3).

3) Restarts: Due to the non-convex nature of the bilinear inference problem, the algorithm may get stuck at local minima or slowed by saddle points. To mitigate these issues, it sometimes helps to restart the algorithm. For each restart, we suggest to initialize $\boldsymbol{\theta}_A^0$ at the final estimate of $\boldsymbol{\theta}_A$ returned by the previous run.

E. Relation to Previous Work

The proposed Bilinear Adaptive VAMP algorithm extends the (linear) Adaptive VAMP algorithm of [51] from the case where $\boldsymbol{A}(\boldsymbol{\theta}_A)$ is known to the case where $\boldsymbol{A}(\boldsymbol{\theta}_A)$ is unknown. In the known-$\boldsymbol{A}(\boldsymbol{\theta}_A)$ setting, where $\boldsymbol{A}(\boldsymbol{\theta}_A)$ is infinitely large and right-rotationally invariant, it was rigorously established in [51] that Adaptive VAMP obeys a state-evolution similar to that of VAMP, and that its estimates of $\{\boldsymbol{\theta}_x, \gamma_w\}$ are asymptotically consistent under certain identifiability conditions, i.e., they converge to the true values as $t \to \infty$. As future work, it would be interesting to understand whether Bilinear Adaptive VAMP also obeys a state evolution for certain classes of $\boldsymbol{A}(\cdot)$.

Algorithm 4: Bilinear Adaptive VAMP (Detailed).

```
 1: initialize:
        ∀l : $\boldsymbol{r}_{1,l}^0, \gamma_{1,l}^0, \boldsymbol{\theta}_x^0, \boldsymbol{\theta}_A^0, \gamma_w^0$
 2: for $t = 0, \dots, T_{\max}$ do
 3:     for $\tau = 0, \dots, \tau_{1,\max}$ do
 4:         $\forall l : \boldsymbol{x}_{1,l}^t \leftarrow \boldsymbol{g}_1(\boldsymbol{r}_{1,l}^t, \gamma_{1,l}^t; \boldsymbol{\theta}_x^t)$
 5:         $\forall l : 1/\eta_{1,l}^t \leftarrow \langle \boldsymbol{g}_1'(\boldsymbol{r}_{1,l}^t, \gamma_{1,l}^t; \boldsymbol{\theta}_x^t)\rangle/\gamma_{1,l}^t$
 6:         $\forall l : 1/\gamma_{1,l}^t \leftarrow \frac{1}{N}\|\boldsymbol{x}_{1,l}^t - \boldsymbol{r}_{1,l}^t\|^2 + 1/\eta_{1,l}^t$
 7:         $q_1^t(\boldsymbol{X}) \propto \prod_{l=1}^L p_{\mathbf{x}}(\boldsymbol{x}_l; \boldsymbol{\theta}_x^t) e^{-\frac{1}{2}\gamma_{1,l}^t\|\boldsymbol{x}_l - \boldsymbol{r}_{1,l}^t\|^2}$
 8:         $\boldsymbol{\theta}_x^t \leftarrow \arg\max_{\boldsymbol{\theta}_x} \mathbb{E}[\ln p_{\mathbf{X}}(\boldsymbol{X}; \boldsymbol{\theta}_x)|q_1^t]$
 9:     end for
10:     $\boldsymbol{\theta}_x^{t+1} = \boldsymbol{\theta}_x^t$
11:     $\forall l : \gamma_{2,l}^t = \eta_{1,l}^t - \gamma_{1,l}^t$
12:     $\forall l : \boldsymbol{r}_{2,l}^t = (\eta_{1,l}^t\boldsymbol{x}_{1,l}^t - \gamma_{1,l}^t\boldsymbol{r}_{1,l}^t)/\gamma_{2,l}^t$
13:     for $\tau = 0, \dots, \tau_{2,\max}$ do
14:         $\forall l : \boldsymbol{C}_l^t \leftarrow \big(\gamma_{2,l}^t\boldsymbol{I}_N + \gamma_w^t\boldsymbol{A}(\boldsymbol{\theta}_A^t)^{\mathsf{T}}\boldsymbol{A}(\boldsymbol{\theta}_A^t)\big)^{-1}$
15:         $\forall l : \boldsymbol{x}_{2,l}^t \leftarrow \boldsymbol{C}_l^t\big(\gamma_{2,l}^t\boldsymbol{r}_{2,l}^t + \gamma_w^t\boldsymbol{A}(\boldsymbol{\theta}_A^t)^{\mathsf{T}}\boldsymbol{y}_l\big)$
16:         $\forall l : 1/\eta_{2,l}^t \leftarrow \operatorname{tr}\{\boldsymbol{C}_l^t\}/N$
17:         $\boldsymbol{C}^t \leftarrow \sum_{l=1}^L \boldsymbol{C}_l^t$
18:         $\forall i,j : [\boldsymbol{H}^t]_{ij} \leftarrow \operatorname{tr}\big\{\boldsymbol{A}_j^{\mathsf{T}}\boldsymbol{A}_i(\boldsymbol{C}^t + \boldsymbol{X}_2^t\boldsymbol{X}_2^{t\,\mathsf{T}})\big\}$
19:         $\forall i : [\boldsymbol{\beta}^t]_i \leftarrow \operatorname{tr}\big\{\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{A}_i\boldsymbol{X}_2^t\big\}$
                        $-\operatorname{tr}\big\{\boldsymbol{A}_0^{\mathsf{T}}\boldsymbol{A}_i(\boldsymbol{C}^t + \boldsymbol{X}_2^t\boldsymbol{X}_2^{t\,\mathsf{T}})\big\}$
20:         $\boldsymbol{\theta}_A^t \leftarrow (\boldsymbol{H}^t)^{-1}\boldsymbol{\beta}^t$
21:         $1/\gamma_w^t \leftarrow \frac{1}{ML}\big(\|\boldsymbol{Y} - \boldsymbol{A}(\boldsymbol{\theta}^t)\boldsymbol{X}_2^t\|_F^2$
                        $+\operatorname{tr}\big\{\boldsymbol{A}(\boldsymbol{\theta}_A^t)\boldsymbol{C}^t\boldsymbol{A}(\boldsymbol{\theta}_A^t)^{\mathsf{T}}\big\}\big)$
22:     end for
23:     $\boldsymbol{\theta}_A^{t+1} = \boldsymbol{\theta}_A^t$
24:     $\gamma_w^{t+1} = \gamma_w^t$
25:     $\forall l : \gamma_{1,l}^{t+1} = \eta_{2,l}^t - \gamma_{2,l}^t$
26:     $\forall l : \boldsymbol{r}_{1,l}^{t+1} = (\eta_{2,l}^t\boldsymbol{x}_{2,l}^t - \gamma_{2,l}^t\boldsymbol{r}_{2,l}^t)/\gamma_{1,l}^{t+1}$
27: end for
```

The proposed BAd-VAMP algorithm targets the same class[8] of bilinear recovery problems as the EM-PBiGAMP algorithm from [33], and both leverage EM for automated hyperparameter tuning. However, the "AMP" aspects of these algorithms are fundamentally different. PBiGAMP treats the vectors $\{\boldsymbol{a}_{i,j}\}$ in (2) as i.i.d. Gaussian for its derivation, whereas BAd-VAMP treats the matrix $\boldsymbol{A}(\boldsymbol{b}) = \sum_{i=1}^Q b_i\boldsymbol{A}_i$ as right rotationally-invariant for its derivation. The latter allows more freedom in the singular values of $\boldsymbol{A}(\boldsymbol{b})$, which leads to increased robustness in practice, as demonstrated by the numerical experiments in Section V.

BAd-VAMP and lifted VAMP both leverage the VAMP approach from [45] to solve bilinear inference problems. However,

[8]Note that the BiGAMP algorithm [19]–[20] is a special case of the PBiGAMP algorithm [33]. BiGAMP applies to the recovery of $\boldsymbol{A}$ and $\boldsymbol{X}$ from measurements of the form $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{W}$, whereas PBiGAMP applies to the recovery of $\boldsymbol{b}$ and $\boldsymbol{X}$ from $\boldsymbol{Y} = \boldsymbol{A}(\boldsymbol{b})\boldsymbol{X} + \boldsymbol{W}$ under known affine linear $\boldsymbol{A}(\cdot)$. Both can be combined with EM for hyperparameter learning.

they do so in very different ways. As discussed in Section I, lifted VAMP "lifts" the bilinear problem into a higher-dimensional linear problem, and then uses non-separable denoising to jointly estimate $\boldsymbol{b}$ and $\boldsymbol{c}$ in (5). An unfortunate consequence of lifting is a possibly significant increase in computational complexity and memory. In contrast, BAd-VAMP avoids lifting, and it employs EM to estimate $\boldsymbol{b}$ and VAMP to estimate $\boldsymbol{c}$. Interestingly, Section V shows BAd-VAMP performing equal or better to lifted VAMP in all experiments.

V. Numerical Simulations

In this section, we present the results of numerical simulations that study the behavior of the BAd-VAMP algorithm from Algorithm 4, in comparison to other state-of-the-art algorithms, on several bilinear recovery problems. In all cases, we ran BAd-VAMP with $\tau_{1,\max} = 1$ and $\tau_{2,\max} = 0$ inner EM iterations and we assumed that the signal prior $p_{\boldsymbol{x}}$ is fully known (i.e., $\boldsymbol{\theta}_x$ is known). We nominally used a damping coefficient of $\zeta = 0.8$ and minimum precision of $\gamma_{\min} = 10^{-6}$. We initialized BAd-VAMP by $\gamma_w^0 = 0.1$, $\gamma_{1,l}^0 = 10^{-3}\ \forall l$, and we set $\boldsymbol{r}_{1,l}^0$ and $\boldsymbol{\theta}_A^0$ to random vectors drawn i.i.d. from $\mathcal{N}(0, 10)$ and $\mathcal{N}(0, 1)$ respectively. Unless otherwise noted, no restarts were used.

A. CS With Matrix Uncertainty

In compressive sensing (CS) with matrix uncertainty [6], the goal is to recover the K-sparse signal $\boldsymbol{c} \in \mathbb{R}^N$ and the uncertainty parameters $\boldsymbol{b}$ from measurements $\boldsymbol{y} = \boldsymbol{A}(\boldsymbol{b})\boldsymbol{c} + \boldsymbol{w}$ with $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}/\gamma_w)$. Here, $\boldsymbol{A}(\boldsymbol{b}) = \boldsymbol{A}_0 + \sum_{i=1}^Q b_i \boldsymbol{A}_i$, where $\{\boldsymbol{A}_i\}_{i=0}^Q$ are known. For our experiments, we used $Q = 10$, $N = 256$, and $K = 10$, and we selected γ_w so that $\text{SNR} \triangleq \mathbb{E}[\|\boldsymbol{A}\boldsymbol{c}\|^2]/\,\mathbb{E}[\|\boldsymbol{w}\|^2] = 40$ dB. Also, the uncertainty parameters $\boldsymbol{b}$ were drawn $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and $\boldsymbol{c}$ was drawn with uniformly random support and with K non-zero elements from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. We measured performance using $\text{NMSE}(\widehat{\boldsymbol{b}}) \triangleq \|\widehat{\boldsymbol{b}} - \boldsymbol{b}\|^2/\|\boldsymbol{b}\|^2$ and $\text{NMSE}(\widehat{\boldsymbol{c}}) \triangleq \|\widehat{\boldsymbol{c}} - \boldsymbol{c}\|^2/\|\boldsymbol{c}\|^2$. As a reference, we considered two oracle estimators: the MMSE estimator for $\boldsymbol{b}$ assuming $\boldsymbol{c}$ is known, and the MMSE estimator for $\boldsymbol{c}$ assuming $\boldsymbol{b}$ and the support of $\boldsymbol{c}$ are known.

For our first experiment, the elements of $\{\boldsymbol{A}_i\}_{i=1}^Q$ were drawn i.i.d. $\mathcal{N}(0, 1)$ and the elements of $\boldsymbol{A}_0$ were drawn $\mathcal{N}(0, 20)$. BAd-VAMP was run for a maximum of 200 iterations with a maximum of 2 restarts and damping $\zeta = 0.86$.

Figure 1 shows that the AMP-based algorithms gave near-oracle performance for the tested range of M/N, although lifted VAMP performed slightly worse than the others when $M/N = 0.2$. In contrast, the performance of WSS-TLS from the award-winning paper [6] was significantly worse than the AMP approaches. WSS-TLS aims to solve the non-convex optimization problem

$$(\widehat{\boldsymbol{b}}, \widehat{\boldsymbol{c}}) = \arg\min_{\boldsymbol{b},\boldsymbol{c}} \left\| \left(\boldsymbol{A}_0 + \sum_{i=1}^Q b_i \boldsymbol{A}_i \right) \boldsymbol{c} - \boldsymbol{y} \right\|^2 + \|\boldsymbol{b}\|^2/\gamma_w + \lambda\|\boldsymbol{c}\|_1 \qquad (85)$$

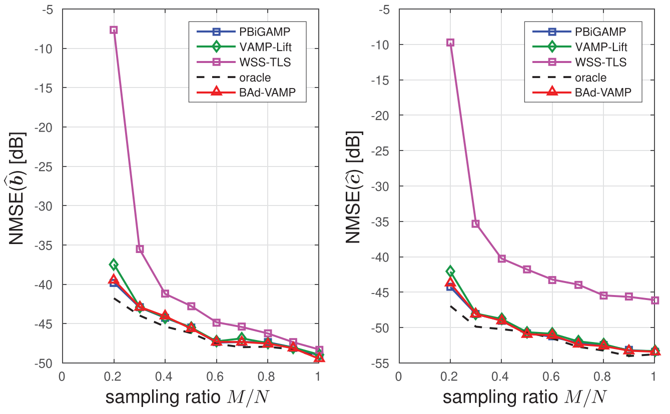

Fig. 1. CS with matrix uncertainty: Median NMSE (over 50 trials) on signal $\boldsymbol{c}$ and uncertainty parameters $\boldsymbol{b}$ versus sampling ratio M/N.

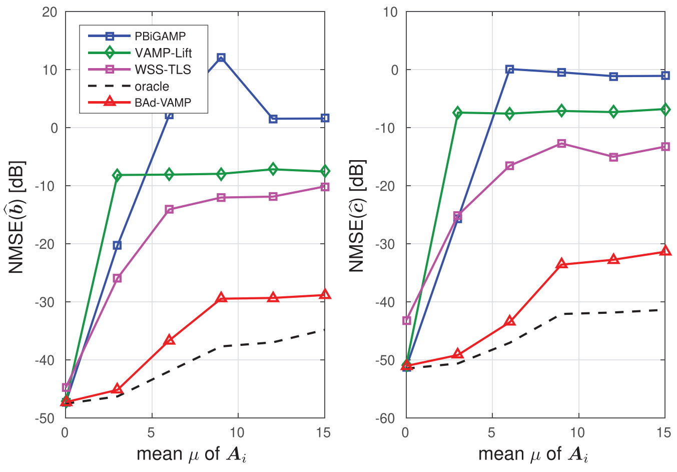

Fig. 2. CS with matrix uncertainty: Median NMSE (over 50 trials) on signal $\boldsymbol{c}$ and uncertainty parameters $\boldsymbol{b}$ versus mean of matrices $\boldsymbol{A}_i$ at $M/N = 0.6$.

using alternating minimization. For WSS-TLS, we used oracle knowledge of γ_w, oracle tuning of the regularization parameter λ, and code from the authors' website.

For our second experiment, we tested algorithm robustness to non-zero mean in $\boldsymbol{A}(\boldsymbol{b})$,[9] since this is a known issue with many AMP algorithms [42]–[44]. For this, we fixed the sampling ratio at $M/N = 0.6$, drew the elements of $\{\boldsymbol{A}_i\}_{i=1}^Q$ from i.i.d. $\mathcal{N}(\mu, 1)$, and drew the elements of $\boldsymbol{A}_0$ from i.i.d. $\mathcal{N}(\mu, 20)$. Figure 2 reports the median NMSE versus mean μ, and shows that BAd-VAMP is much more robust to $\mu > 0$ than the other tested AMP algorithms as well as WSS-TLS.

Figure 3 shows the runtime of the algorithms in Figure 1. Our implementation used MATLAB (R2015b) on an RHEL workstation with an 8-core Intel i7 processor. Although, for WSS-TLS, we used a grid-search to optimize λ in (85), Figure 3 only shows the runtime of WSS-TLS after λ was chosen. Figure 3 shows BAd-VAMP running much faster than lifted VAMP and WSS-TLS, and slightly slower than PBiGAMP.

[9]For the simpler case where $\boldsymbol{b}$ is known and the objective is to recover $\boldsymbol{c}$ from $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{c} + \boldsymbol{w}$, modifications of AMP that temporarily remove the mean from $\boldsymbol{A}$ have been proposed [44]. However, it is not clear how to extend this approach to the bilinear problem of recovering $\boldsymbol{b}$ and $\boldsymbol{c}$ from $\boldsymbol{y} = \boldsymbol{A}(\boldsymbol{b})\boldsymbol{c} + \boldsymbol{w}$.

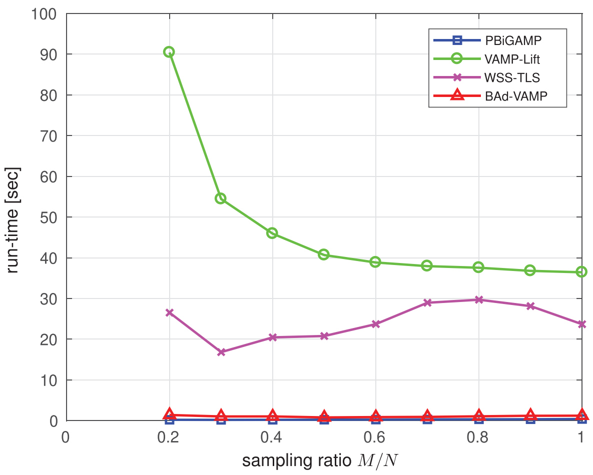

Fig. 3. CS with matrix uncertainty: Median run-time in seconds (over 10 trials) versus sampling ratio M/N.

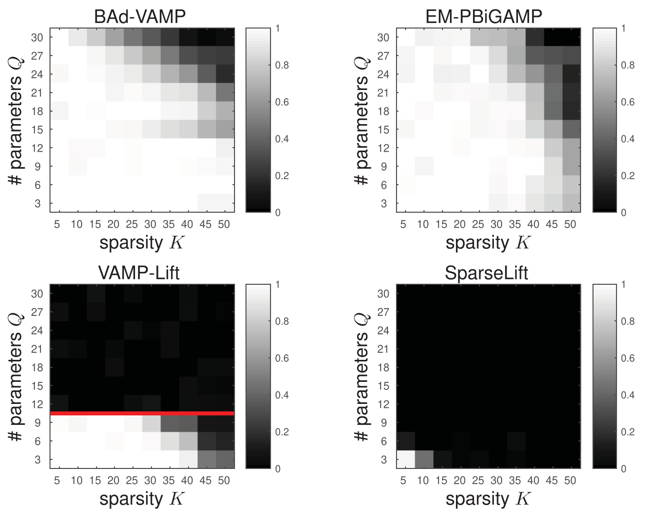

Fig. 4. Self-calibration: Empirical success rate (over 50 trials) for several algorithms versus number of calibration parameters Q and sparsity K. For computational reasons, VAMP-Lift was simulated only for $Q < 10$.

B. Self-Calibration

In self-calibration [5], the goal is to recover the K-sparse signal vector $\boldsymbol{c}$ and the calibration parameters $\boldsymbol{b}$ from measurements of the form $\boldsymbol{y} = \text{Diag}(\boldsymbol{Hb})\boldsymbol{\Psi c}$ with known $\boldsymbol{H} \in \mathbb{R}^{M\times Q}$ and $\boldsymbol{\Psi} \in \mathbb{R}^{M\times N}$. Here, $\boldsymbol{Hb}$ represents an unknown vector of gains on the measurements, where the gain vector is believed to lie in the Q-dimensional subspace spanned by the columns of $\boldsymbol{H}$. For our experiment, $M = 128$, $N = 256$, $\boldsymbol{\Psi}$ and $\boldsymbol{b}$ where drawn i.i.d. $\mathcal{N}(0, 1)$, $\boldsymbol{H}$ was constructed using Q randomly selected columns of the Hadamard matrix, and $\boldsymbol{c}$ was drawn with uniformly random support and with K non-zero elements from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

Figure 4 shows the rate of successful recovery versus subspace dimension Q and sparsity K for several algorithms. A recovery $(\widehat{\boldsymbol{b}}, \widehat{\boldsymbol{c}})$ was considered "successful" when $\|\widehat{\boldsymbol{b}}\widehat{\boldsymbol{c}}^{\mathsf{T}} - \boldsymbol{bc}^{\mathsf{T}}\|_F^2/\|\boldsymbol{bc}^{\mathsf{T}}\|_F^2 \leq -50$ dB. From the figure, we see that the performance of BAd-VAMP is similar to that of EM-PBiGAMP, and even slightly better when Q is small and K is large. Meanwhile, BAd-VAMP appears significantly better than both lifted VAMP and SparseLift from [5]. SparseLift is a convex relaxation with provable guarantees [5]. For computational reasons (recall the discussion in Section I-B), it was difficult to simulate lifted VAMP for $Q \geq 10$.

C. Calibration in Tomography

We consider the problem of reconstructing an image from a sequence of tomographic projections, where the projections along each direction are scaled by an unknown calibration gain. In particular, let $\boldsymbol{\Psi}_\omega$ be the tomographic projection matrix[10] (i.e., Radon transform) corresponding to angle $\omega \in [0, \pi]$. Our goal is to reconstruct the image $\boldsymbol{x}$ from measurements

$$\boldsymbol{y} = \begin{bmatrix} b_1\boldsymbol{\Psi}_{\omega_1} \\ \vdots \\ b_K\boldsymbol{\Psi}_{\omega_K} \end{bmatrix} \boldsymbol{x} + \boldsymbol{w} \text{ for } \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \gamma_w^{-1}\boldsymbol{I}), \tag{86}$$

where $b_k \sim \mathcal{N}(1, \sigma_b^2)$ are unknown. Note that, by defining $\boldsymbol{A}_k = [\boldsymbol{0}, \dots, \boldsymbol{\Psi}_{\omega_k}^{\mathsf{T}}, \boldsymbol{0}]^{\mathsf{T}}$, we can write $\boldsymbol{y} = \boldsymbol{A}(\boldsymbol{b})\boldsymbol{x} + \mathcal{N}(\boldsymbol{0}, \gamma_w^{-1}\boldsymbol{I})$ for $\boldsymbol{A}(\boldsymbol{b}) = \sum_{k=1}^K b_k\boldsymbol{A}_k$, which matches (8).

In an attempt to solve the above problem, we used BAd-VAMP to recover the image $\boldsymbol{x}$ while simultaneously learning the calibration gains $\boldsymbol{b}$. For this, we used BAd-VAMP in "plug-and-play" mode, where the BM3D image denoiser [64] was used to implement the $\boldsymbol{g}_1(\cdot)$ function in Algorithm 1. Due to the inherent scaling ambiguity of the problem (i.e., if $(\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{b}})$ is a solution then so is $(\alpha\widehat{\boldsymbol{x}}, \alpha^{-1}\widehat{\boldsymbol{b}})$ for any $\alpha > 0$), we scaled the image estimate $\widehat{\boldsymbol{x}}$ by the α that minimized $\|\boldsymbol{x} - \alpha\widehat{\boldsymbol{x}}\|$ before computing the PSNR.

As baselines, we also tested the VAMP [45], total variation (TV) [65], [66] and regularization-by-denoising (RED) [67], [68] approaches (see descriptions below). These approaches all assume a noisy *linear* data model of the form $\boldsymbol{y} = \widehat{\boldsymbol{A}}\boldsymbol{x} + \mathcal{N}(0, \gamma_w^{-1}\boldsymbol{I})$ with known $\widehat{\boldsymbol{A}}$. To apply them to (86), we considered two cases: the *genie-calibrated* (GC) case, where a genie supplies the true gains $\boldsymbol{b}$ and the algorithm uses $\widehat{\boldsymbol{A}} = \boldsymbol{A}(\boldsymbol{b})$, and the *un-calibrated* (UC) case, where $\boldsymbol{b}$ is unknown and the algorithms assume $\widehat{\boldsymbol{A}} = \boldsymbol{A}(\boldsymbol{1})$. In the latter case, the $\widehat{\boldsymbol{A}}$-based model is mismatched to the data-generation model (86). For fair comparison, we scaled the GC and UC image estimates $\widehat{\boldsymbol{x}}$ by the α that minimized $\|\boldsymbol{x} - \alpha\widehat{\boldsymbol{x}}\|_2$ before computing the PSNR.

We now provide additional details on the experimental setup. For $\boldsymbol{x}$, we used the modified Shepp-Logan phantom of size 64×64, shown in the top-left panel of Fig. 5. For $\boldsymbol{A}(\cdot)$, we used $K = 25$ projections spaced uniformly in $\omega \in [0, \pi]$. The calibration gains $\boldsymbol{b}$ were generated using $\sigma_b = 0.06$, and the noise precision γ_w was set to achieve an SNR of $\mathbb{E}[\|\boldsymbol{A}(\boldsymbol{b})\boldsymbol{x}\|^2]/\mathbb{E}[\|\boldsymbol{w}\|^2] = 40$ dB. The TV method [65] computes

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \left\{ \frac{1}{2}\|\boldsymbol{y} - \widehat{\boldsymbol{A}}\boldsymbol{x}\|_2^2 + \lambda_t\|\nabla\boldsymbol{x}\|_{2,1} \right\} \tag{87}$$

[10]We used the matrix form of the Radon transform instead of the operator form to avoid numerical error when implementing the adjoint.

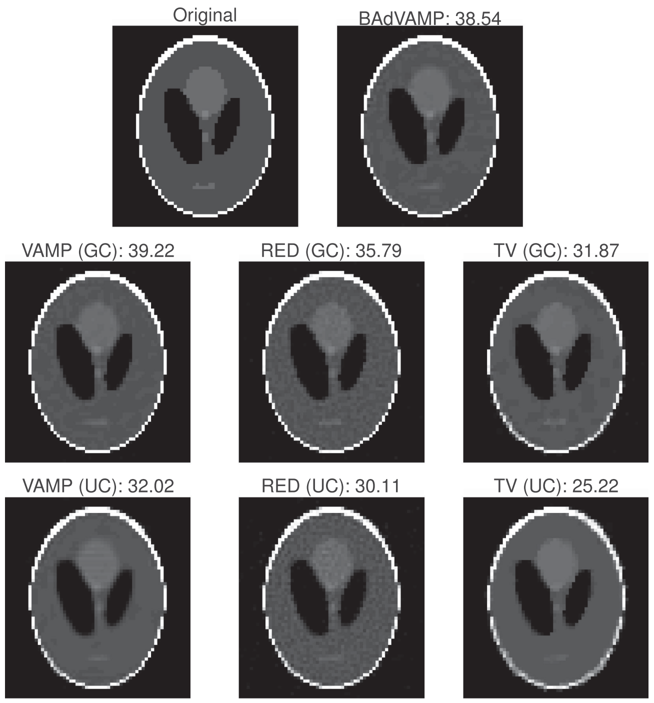

Fig. 5. Calibration in tomography: Reconstruction PSNR (dB) of 64×64 Shepp-Logan phantom from 25 equally spaced tomographic projections. In the genie-calibrated (GC) case, $\widehat{\boldsymbol{A}} = \boldsymbol{A}(\boldsymbol{b})$, while in the un-calibrated (UC) case, $\widehat{\boldsymbol{A}} = \boldsymbol{A}(\mathbf{1})$.

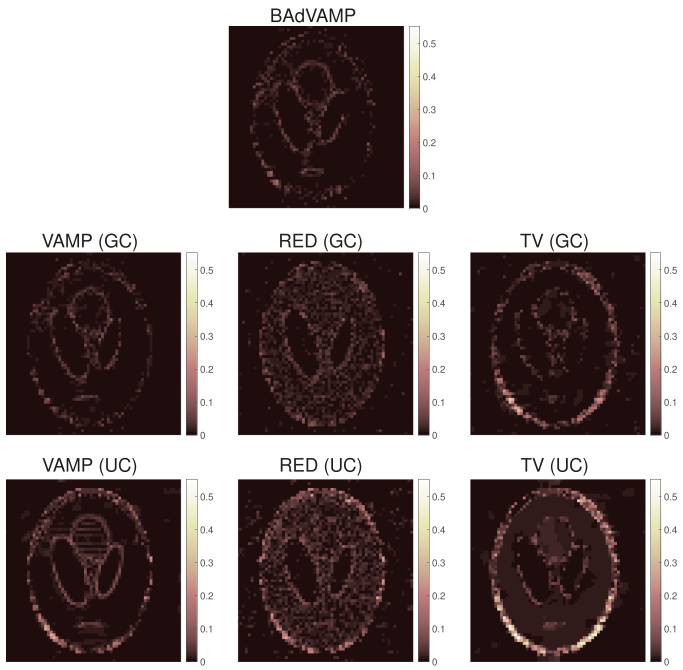

Fig. 6. Calibration in tomography: Error images for the reconstructions shown in Fig. 5.

TABLE I
PSNR (dB) in the Tomography Experiment

measurements	BAd-VAMP	VAMP	RED	TV
genie calibrated (GC)	—	**39.57**	36.56	33.27
un-calibrated (UC)	**38.27**	31.62	31.24	26.79

for the isotropic TV operator

$$\|\nabla \boldsymbol{x}\|_{2,1} = \sum_{i,j} \sqrt{(x_{i,j} - x_{i,j-1})^2 + (x_{i,j} - x_{i-1,j})^2}. \quad (88)$$

We solved (87) using FASTA [69], and tuned λ_t to maximize the PSNR. RED [67], [68] solves the fixed-point equation

$$\widehat{\boldsymbol{A}}^{\mathsf{H}}(\widehat{\boldsymbol{A}}\widehat{\boldsymbol{x}} - \boldsymbol{y}) + \lambda_r(\widehat{\boldsymbol{x}} - \boldsymbol{\rho}(\widehat{\boldsymbol{x}}, \tau)) = \mathbf{0} \quad (89)$$

for $\widehat{\boldsymbol{x}}$, where $\boldsymbol{\rho}(\cdot, \tau)$ is an image denoising algorithm with noise-variance τ. For our experiment, we used BM3D for $\boldsymbol{\rho}(\cdot, \tau)$, solved (89) using the ADMM method from [67] with 200 iterations, and tuned both λ_r and τ to maximize PSNR. For BAd-VAMP, we initialized $\boldsymbol{b}$ to $\mathbf{1}$, used damping $\zeta = 0.1$, assumed known noise precision γ_w, and used at most 100 iterations. For VAMP, we initialized $\gamma_1^0 = 10^{-4}$ and used at most 100 iterations.

Table I reports the median PSNR achieved by each algorithm across 10 random draws of $\boldsymbol{b}$ and $\boldsymbol{w}$, Fig. 5 shows example image recoveries, and Fig. 6 shows the corresponding error images. From Table I, we see that the PSNR performance of BAd-VAMP (which does not know $\boldsymbol{b}$) is nearly as good as genie-calibrated VAMP, and 1.7 dB better than genie-calibrated RED. Furthermore, the PSNR performance of BAd-VAMP is more than 6.6 dB better than un-calibrated VAMP and 7 dB better than un-calibrated RED. The uncalibrated VAMP, RED, and TV recoveries in Fig. 5 are plagued by either streaking artifacts and/or loss of detail (e.g., note the disappearance of the small white dots in uncalibrated TV). But the BAdVAMP image recovery in Fig. 5 shows no streaking artifacts and a high level of detail. Likewise, Fig. 6 shows that TV has trouble correctly recovering the white outer ellipse, RED has trouble in the interior region, but BAd-VAMP does well throughout.

D. Noiseless Dictionary Learning

In dictionary learning (DL) [3], the goal is to find a dictionary matrix $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ and a sparse matrix $\boldsymbol{X} \in \mathbb{R}^{N \times L}$ such that a given matrix $\boldsymbol{Y} \in \mathbb{R}^{M \times L}$ can be approximately factored as $\boldsymbol{Y} \approx \boldsymbol{A}\boldsymbol{X}$. In this section, we test the proposed BAd-VAMP algorithm for DL by generating $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$ such that $\boldsymbol{X}$ has K-sparse columns, and measuring the NMSE on the resulting estimates of $\boldsymbol{A}$ and $\boldsymbol{X}$.

We consider two cases: i) where the true $\boldsymbol{A}$ is structured as $\boldsymbol{A} = \sum_{i=1}^{Q} b_i \boldsymbol{A}_i$ with known $\{\boldsymbol{A}_i\}_{i=1}^{Q}$ (recall (72)), and ii) where the true $\boldsymbol{A}$ is unstructured (recall (80)). In either case, the pair $(\boldsymbol{A}, \boldsymbol{X})$ is recoverable only up to an ambiguity: a scalar ambiguity in the structured case and a generalized permutation ambiguity in the unstructured case. Thus, when measuring reconstruction quality, we consider

$$\text{NMSE}(\widehat{\boldsymbol{A}}) \triangleq \min_{\lambda \in \mathbb{R}} \frac{\|\boldsymbol{A} - \lambda \widehat{\boldsymbol{A}}\|_F^2}{\|\boldsymbol{A}\|_F^2} \quad (90)$$

in the structured case and

$$\text{NMSE}(\widehat{\boldsymbol{A}}) \triangleq \min_{\boldsymbol{P} \in \mathcal{P}} \frac{\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\boldsymbol{P}\|_F^2}{\|\boldsymbol{A}\|_F^2} \quad (91)$$

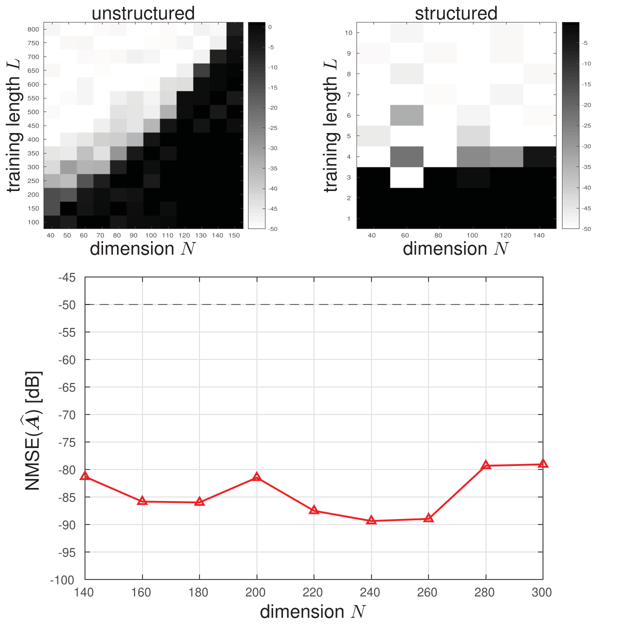

Fig. 7. BAd-VAMP dictionary learning with $N \times N$ dictionary $\boldsymbol{A}$ and $N \times L$ code matrix $\boldsymbol{X}$ with sparsity rate $K/N = 0.2$. Top left: success-rate for unstructured dictionary versus N and L. Top right: success-rate for structured dictionary with $Q = N$ free parameters. Bottom: Median NMSE($\widehat{\boldsymbol{A}}$) over 20 trials versus N for an unstructured dictionary with $L = 6N \ln N$.

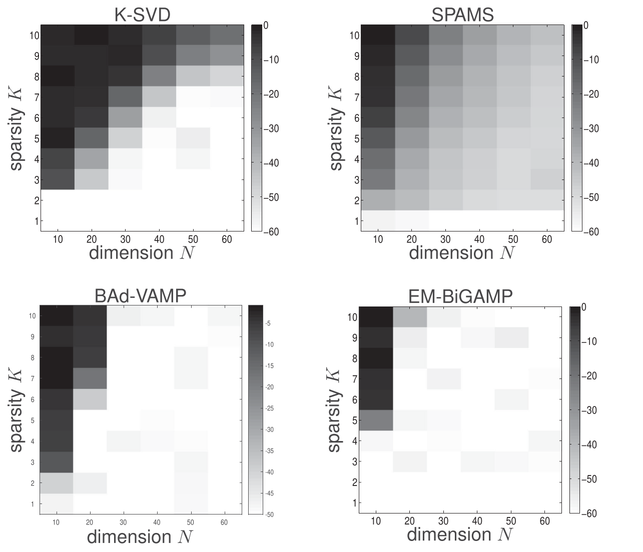

Fig. 8. Noiseless dictionary learning: Median NMSE($\widehat{\boldsymbol{A}}$) in dB (over 20 trials) versus dictionary dimension N and sparsity K with unstructured dictionary $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ and training length $L = 5N \ln N$.

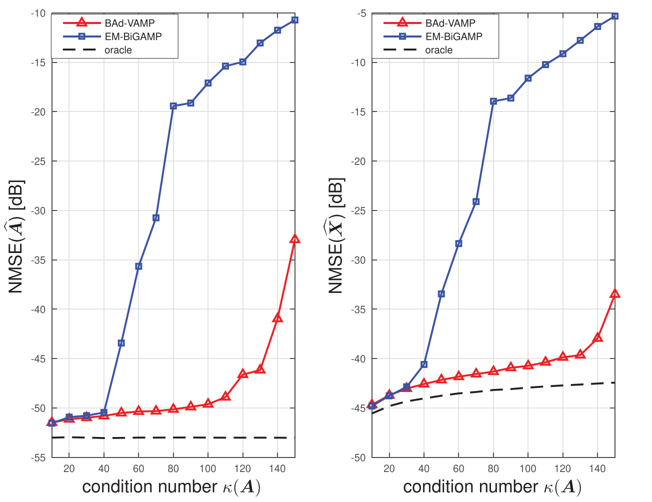

Fig. 9. Noisy dictionary learning: Median NMSE in dB (over 50 trials) versus condition number $\kappa(\boldsymbol{A})$ for unstructured $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ with $N = 64$, sparsity $K = 13$, training length $L = 5N \ln N$, and SNR = 40 dB.

in the unstructured case, where $\mathcal{P}$ denotes the set of generalized permutation matrices.[11] For our experiments, we drew the coefficients of $\{\boldsymbol{A}_i\}_{i=1}^{Q}$ and $\boldsymbol{b}$ as i.i.d. $\mathcal{N}(0, 1)$ with $Q = N$ in the structured case, and we drew the coefficients of $\boldsymbol{A}$ as i.i.d. $\mathcal{N}(0, 1)$ in the unstructured case.

In our first experiment, we fixed the sparsity rate at $K/N = 0.2$ and we varied both the dictionary dimension N and the training length L. The top-right panel of Fig. 7 suggests that, as the dimension N grows, a *fixed* training length L is sufficient to successfully recover $\boldsymbol{A}$ in the structured case with $Q = N$. By "successfully recover," we mean that NMSE($\widehat{\boldsymbol{A}}$) ≤ -50 dB. Note that this latter prescription for L is consistent with the theoretical analysis in [18]. In the unstructured case, the bottom panel of Figure 7 shows the median NMSE($\widehat{\boldsymbol{A}}$) versys N when $L = 6N \ln N$. Together, the top-left and bottom panels of Fig. 7 suggest that a training length of $L = O(N \ln N)$ suffices to successfully recover $\boldsymbol{A}$.

In our second experiment, we focused on the unstructured case, fixed the training length at $L = 5N \ln N$, and varied both the dictionary dimension N and the sparsity K in the columns of $\boldsymbol{X}$. Figure 8 shows that BAd-VAMP performed similarly to EM-BiGAMP [20] for all but very small N, and much better than K-SVD [16] and SPAMS [17]. The advantage of BAd-VAMP over EM-BiGAMP for DL will be illustrated in the sequel.

E. Noisy, Ill-Conditioned Dictionary Learning

In this section, we show the robustness of BAd-VAMP over EM-BiGAMP [19] when learning ill-conditioned dictionaries from noisy measurements. To do so, we generated the measurements as $\boldsymbol{Y} = \boldsymbol{AX} + \boldsymbol{W}$ and tested the algorithms in recovering $\boldsymbol{A}$ and $\boldsymbol{X}$ (up to appropriate ambiguities). The elements of $\boldsymbol{W}$ were drawn i.i.d. $\mathcal{N}(0, 1/\gamma_w)$ with γ_w chosen to achieve SNR $\triangleq \mathbb{E}[\|\boldsymbol{AX}\|_F^2]/\mathbb{E}[\|\boldsymbol{W}\|_F^2] = 40$ dB. The true dictionary was generated as $\boldsymbol{A} = \boldsymbol{U}\,\mathrm{Diag}(\boldsymbol{s})\boldsymbol{V}^{\mathsf{T}}$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ were drawn uniformly over the group of orthogonal matrices, and where the singular values in $\boldsymbol{s}$ were chosen so that $s_i/s_{i-1} = \rho\ \forall i$. The values of s_0 and ρ were selected to obtain a desired condition number $\kappa(\boldsymbol{A})$ while also ensuring $\|\boldsymbol{A}\|_F^2 = N$.

Figure 9 reports median NMSE($\widehat{\boldsymbol{A}}$) and NMSE($\widehat{\boldsymbol{X}}$) versus condition number $\kappa(\boldsymbol{A})$ for the recovery of $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ and K-sparse $\boldsymbol{X} \in \mathbb{R}^{N \times L}$ from noisy measurements $\boldsymbol{Y}$. For this

[11]If $\boldsymbol{P}$ is a generalized permutation matrix then $\boldsymbol{P} = \boldsymbol{\Pi D}$, where $\boldsymbol{\Pi}$ is a permutation matrix and $\boldsymbol{D}$ is a diagonal matrix.

figure, we used $K = 13$, $N = 64$, $L = 5N \ln N$, the unstructured definition of NMSE$(\widehat{\boldsymbol{A}})$ from (91), and a similar definition for NMSE$(\widehat{\boldsymbol{X}})$. In addition to showing the performance of BAd-VAMP and EM-PBiGAMP, the figure shows the performance of the known-$\boldsymbol{X}$ oracle for the estimation of $\boldsymbol{A}$, as well as the known-$\boldsymbol{A}$ and known-support oracle for the estimation of $\boldsymbol{X}$. Figure 9 shows that EM-BiGAMP gave near-oracle NMSE for $\kappa(\boldsymbol{A}) \leq 40$, but its performance degraded significantly for larger $\kappa(\boldsymbol{A})$. In contrast, BAd-VAMP gave near-oracle NMSE for $\kappa(\boldsymbol{A}) \leq 110$, which suggests increased robustness to ill-conditioned dictionaries $\boldsymbol{A}$.

VI. Conclusion

In this paper, we considered the problem of jointly recovering the vector $\boldsymbol{b}$ and the matrix $\boldsymbol{C}$ from noisy measurements $\boldsymbol{Y} = \boldsymbol{A}(\boldsymbol{b})\boldsymbol{C} + \boldsymbol{W}$, where $\boldsymbol{A}(\cdot)$ is a known affine linear function of $\boldsymbol{b}$ (i.e., $\boldsymbol{A}(\boldsymbol{b}) = \boldsymbol{A}_0 + \sum_{i=1}^{Q} b_i \boldsymbol{A}_i$ with known matrices $\boldsymbol{A}_i$). To solve this problem, we proposed the BAd-VAMP algorithm, which combines the VAMP algorithm [45], the EM algorithm [61], and variance auto-tuning [62] in a manner appropriate for bilinear recovery. We demonstrated numerically that the proposed approach has robustness advantages over other state-of-the-art bilinear recovery algorithms, including lifted VAMP [50] and EM-PBiGAMP [33]. As future work, we plan to rigorously analyze BAd-VAMP through the state-evolution formalism.

References

[1] E. J. Candès and Y. Plan, “Matrix completion with noise,” *Proc. IEEE*, vol. 98, no. 6, pp. 925–936, Jun. 2010.

[2] E. J. Candès, X. Li, Y. Ma, and J. Wright, “Robust principal component analysis?” *J. ACM*, vol. 58, no. 3, May 2011, Art. no. 11.

[3] R. Rubinstein, A. M. Bruckstein, and M. Elad, “Dictionaries for sparse representation modeling,” *Proc. IEEE*, vol. 98, no. 6, pp. 1045–1057, Jun. 2010.

[4] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in *Proc. 13th Int. Conf. Neural Inform. Process. Syst.*, 2001, pp. 556–562.

[5] S. Ling and T. Strohmer, “Self-calibration and biconvex compressive sensing,” *Inverse Problems*, vol. 31, no. 11, 2015, Art. no. 115002.

[6] H. Zhu, G. Leus, and G. B. Giannakis, “Sparsity-cognizant total least-squares for perturbed compressive sampling,” *IEEE Trans. Signal Process.*, vol. 59, no. 5, pp. 2002–2016, May 2011.

[7] G. K. Kaleh and R. Vallet, “Joint parameter estimation and symbol detection for linear or nonlinear unknown channels,” *IEEE Trans. Commun.*, vol. 42, pp. 2406–2413, Jul. 1994.

[8] E. J. Candès and Y. Plan, “Tight oracle inequalities for low-rank matrix recovery from a minimal number of noisy random measurements,” *IEEE Trans. Inf. Theory*, vol. 57, no. 4, pp. 2342–2359, Apr. 2011.

[9] A. E. Waters, A. C. Sankaranarayanan, and R. G. Baraniuk, “SpaRCS: Recovering low-rank and sparse matrices from compressive measurements,” in *Proc. 24th Int. Conf. Neural Inform. Process. Syst.*, 2011, pp. 1089–1097.

[10] Z. Lin, M. Chen, L. Wu, and Y. Ma, “The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices,” 2010, arXiv:1009.5055.

[11] Z. Wen, W. Yin, and Y. Zhang, “Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm,” *Math. Program. Comput.*, vol. 4, pp. 333–361, 2012.

[12] L. Balzano, R. Nowak, and B. Recht, “Online identification and tracking of subspaces from highly incomplete information,” in *Proc. Allerton Conf. Commun. Control Comput.*, Sep. 2010, pp. 704–711.

[13] A. Kyrillidis and V. Cevher, “Matrix recipes for hard thresholding methods,” *J. Math. Imag. Vis.*, vol. 48, pp. 235–265, 2014.

[14] S. D. Babacan, M. Luessi, R. Molina, and A. K. Katsaggelos, “Sparse Bayesian methods for low-rank matrix estimation,” *IEEE Trans. Signal Process.*, vol. 60, no. 8, pp. 3964–3977, Aug. 2012.

[15] J. He, L. Balzano, and A. Szlam, “Incremental gradient on the Grassmannian for online foreground and background separation in subsampled video,” in *Proc. IEEE Conf. Comp. Vis. Pattern Recognit.*, 2012, pp. 1568–1575.

[16] M. Aharon, M. Elad, and A. Bruckstein, “K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation,” *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[17] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, “Online learning for matrix factorization and sparse coding,” *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, Jan. 2010.

[18] D. A. Spielman, H. Wang, and J. Wright, “Exact recovery of sparsely-used dictionaries,” in *Proc. 25th Annu. Conf. Learn. Theory*, 2012, pp. 37.1–37.18.

[19] J. T. Parker, P. Schniter, and V. Cevher, “Bilinear generalized approximate message passing—Part I: Derivation,” *IEEE Trans. Signal Process.*, vol. 62, no. 22, pp. 5839–5853, Nov. 2014.

[20] J. T. Parker, P. Schniter, and V. Cevher, “Bilinear generalized approximate message passing—Part II: Applications,” *IEEE Trans. Signal Process.*, vol. 62, no. 22, pp. 5854–5867, Nov. 2014.

[21] Y. Kabashima, F. Krzakala, M. Mézard, A. Sakata, and L. Zdeborová, “Phase transitions and sample complexity in Bayes-optimal matrix factorization,” *IEEE Trans. Inf. Theory*, vol. 62, no. 7, pp. 4228–4265, Jul. 2016.

[22] 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.

[23] S. Rangan, “Generalized approximate message passing for estimation with random linear mixing,” in *Proc. IEEE Int. Symp. Inf. Theory*, Aug. 2011, pp. 2168–2172.

[24] R. Matsushita and T. Tanaka, “Low-rank matrix reconstruction and clustering via approximate message passing,” in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 917–925.

[25] T. Lesieur, F. Krzakala, and L. Zdeborová, “Constrained low-rank matrix estimation: Phase transitions, approximate message passing and applications,” *J. Stat. Mech.*, vol. 2017, no. 7, 2017, Art. no. 073403.

[26] L. Miolane, “Fundamental limits of low-rank matrix estimation: The non-symmetric case,” 2017, arXiv:1702.00473.

[27] C. Bilen, G. Puy, and R. Gribonval, “Convex optimization approaches for blind sensor calibration using sparsity,” *IEEE Trans. Signal Process.*, vol. 62, no. 18, pp. 4847–4856, Sep. 2014.

[28] A. Ahmed, B. Recht, and J. Romberg, “Blind deconvolution using convex programming,” *IEEE Trans. Inf. Theory*, vol. 60, no. 3, pp. 1711–1732, Mar. 2014.

[29] C. Hegde and R. G. Baraniuk, “Sampling and recovery of pulse streams,” *IEEE Trans. Signal Process.*, vol. 59, no. 14, pp. 1505–1517, Apr. 2011.

[30] A. Agarwal, S. Negahban, and M. J. Wainwright, “Matrix decomposition via convex relaxation: Optimal rates in high dimensions,” *Ann. Statist.*, vol. 40, no. 2, pp. 1171–1197, 2012.

[31] J. Wright, A. Ganesh, K. Min, and Y. Ma, “Compressive principal component pursuit,” *Inf. Inference*, vol. 2, no. 1, pp. 32–68, 2013.

[32] M. A. Davenport and J. Romberg, “An overview of low-rank matrix recovery from incomplete observations,” *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 4, pp. 608–622, Jun. 2016.

[33] J. T. Parker and P. Schniter, “Parametric bilinear generalized approximate message passing,” *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 4, pp. 795–808, Jun. 2016.

[34] C. Schülke, P. Schniter, and L. Zdeborová, “Phase diagram of matrix compressed sensing,” *Phys. Rev. E*, vol. 94, no. 6, Dec. 2016, Art. no. 062136.

[35] E. J. Candès, T. Strohmer, and V. Voroninski, “PhaseLift: Exact and stable signal recovery from magnitude measurements via convex programming,” *Commun. Pure Appl. Math.*, vol. 66, no. 8, pp. 1241–1274, 2013.

[36] E. Romanov and M. Gavish, “Near-optimal matrix recovery from random linear measurements,” *Proc. Nat. Acad. Sci.*, vol. 115, pp. 7200–7205, 2018.

[37] C. A. Metzler, A. Maleki, and R. G. Baraniuk, “From denoising to compressed sensing,” *IEEE Trans. Inf. Theory*, vol. 62, no. 9, pp. 5117–5144, Sep. 2016.

[38] R. Berthier, A. Montanari, and P.-M. Nguyen, “State evolution for approximate message passing with non-separable functions,” *Inf. Inference*, 2019.

[39] C. Rush and R. Venkataramanan, “Finite-sample analysis of approximate message passing,” in *Proc. IEEE Int. Symp. Inf. Theory*, 2016, pp. 755–759.

[40] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 764–785, Feb. 2011.

[41] M. Bayati, M. Lelarge, and A. Montanari, "Universality in polytope phase transitions and message passing algorithms," *Ann. App. Prob.*, vol. 25, no. 2, pp. 753–822, 2015.

[42] S. Rangan, P. Schniter, E. Riegler, A. Fletcher, and V. Cevher, "Fixed points of generalized approximate message passing with arbitrary matrices," *IEEE Trans. Inf. Theory*, vol. 62, no. 12, pp. 7464–7474, Dec. 2016.

[43] F. Caltagirone, F. Krzakala, and L. Zdeborová, "On convergence of approximate message passing," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2014, pp. 1812–1816.

[44] 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. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2015, pp. 2021–2025.

[45] S. Rangan, P. Schniter, and A. K. Fletcher, "Vector approximate message passing," in *Proc. IEEE Int. Symp. Inf. Theory*, 2017, pp. 1588–1592.

[46] M. Opper and O. Winther, "Expectation consistent free energies for approximate inference," in *Proc. 17th Int. Conf. Neural Inf. Process. Syst.*, 2005, pp. 1001–1008.

[47] A. K. Fletcher, M. Sahraee-Ardakan, S. Rangan, and P. Schniter, "Expectation consistent approximate inference: Generalizations and convergence," in *Proc. IEEE Int. Symp. Inf. Theory*, 2016, pp. 190–194.

[48] J. Ma and L. Ping, "Orthogonal AMP," *IEEE Access*, vol. 5, pp. 2020–2033, 2017.

[49] K. Takeuchi, "Rigorous dynamics of expectation-propagation-based signal recovery from unitarily invariant measurements," in *Proc. IEEE Int. Symp. Inf. Theory*, 2017, pp. 501–505.

[50] A. K. Fletcher, S. Rangan, S. Sarkar, and P. Schniter, "Plug-in estimation in high-dimensional linear inverse problems: A rigorous analysis," in *Proc. Neural Inf. Process. Syst. Conf.*, 2018, pp. 7440–7449.

[51] A. K. Fletcher, M. Sahraee-Ardakan, S. Rangan, and P. Schniter, "Rigorous dynamics and consistent estimation in arbitrarily conditioned linear systems," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 2542–2551.

[52] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Found. Trends Mach. Learn.*, vol. 1, pp. 1–305, Jan. 2008.

[53] T. Minka, "A family of approximate algorithms for Bayesian inference," Ph.D. dissertation, Dept. Comp. Sci. Eng., MIT, Cambridge, MA, USA, Jan. 2001.

[54] M. Seeger, "Expectation propagation for exponential families," Ecole polytechnique fdrale de Lausanne, Lausanne, Switzerland, Tech. Rep. 161464, 2005.

[55] T. Heskes and O. Zoeter, "Expectation propagation for approximate inference in dynamic Bayesian networks," in *Proc. Uncertainty Artif. Intell.*, 2002, pp. 313–320.

[56] A. M. Tulino, G. Caire, S. Verdú, and S. Shamai (Shitz),"Support recovery with sparsely sampled free random matrices," *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4243–4271, Jul. 2013.

[57] G. Reeves, "Additivity of information in multilayer networks via additive Gaussian noise transforms," in *Proc. Allerton Conf. Commun. Control Comput.*, 2017, pp. 1064–1070.

[58] G. Reeves and H. D. Pfister, "The replica-symmetric prediction for compressed sensing with Gaussian matrices is exact," in *Proc. IEEE Int. Symp. Inf. Theory*, 2016, pp. 665–669.

[59] J. Barbier, M. Dia, N. Macris, and F. Krzakala, "The mutual information in random linear estimation," in *Proc. Allerton Conf. Commun. Control Comput.*, 2016, pp. 625–632.

[60] A. Dempster, N. M. Laird, and D. B. Rubin, "Maximum-likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc.*, vol. 39, pp. 1–17, 1977.

[61] R. Neal and G. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," in *Learning in Graphical Models*, M. I. Jordan, Ed. Cambridge, MA, USA: MIT Press, 1998, pp. 355–368.

[62] U. S. Kamilov, S. Rangan, A. K. Fletcher, and M. Unser, "Approximate message passing with consistent parameter estimation and applications to sparse learning," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2969–2985, May 2014.

[63] J. P. Vila and P. Schniter, "Expectation-maximization Gaussian-mixture approximate message passing," *IEEE Trans. Signal Process.*, vol. 61, no. 19, pp. 4658–4672, Oct. 2013.

[64] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[65] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, pp. 259–268, 1992.

[66] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problem," *IEEE Trans. Image Process.*, vol. 18, no. 11, pp. 2419–2434, Nov. 2009.

[67] Y. Romano, M. Elad, and P. Milanfar, "The little engine that could: Regularization by denoising (RED)," *SIAM J. Imag. Sci.*, vol. 10, no. 4, pp. 1804–1844, 2017.

[68] E. T. Reehorst and P. Schniter, "Regularization by denoising: Clarifications and new interpretations," *IEEE Trans. Comp. Imag.*, vol. 5, no. 1, pp. 52–67, Mar. 2019.

[69] T. Goldstein, C. Studer, and R. Baraniuk, "Forward-backward splitting with a FASTA implementation," 2014, arXiv:1411.3406.

Subrata Sarkar (S'12) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Guwahati, India, in 2014, and the M.S. degree in electrical engineering, in 2016, from The Ohio State University, Columbus, OH, USA, where he is currently working toward the Ph.D. degree.

Alyson K. Fletcher (S'03–M'04) received the B.S. degree in mathematics from the University of Iowa, Iowa City, IA, USA, the M.S. degree in mathematics and electrical engineering, and the Ph.D. degree in electrical engineering, both from the University of California at Berkeley, Berkeley, CA, USA. Since 2016, she has been on the Faculty of the Departments of Statistics, Mathematics, Electrical Engineering, and Computer Science, University of California, Los Angeles, Los Angeles, CA, USA.

Sundeep Rangan (S'94–M'98–SM'13–F'16) received the B.A.Sc. degree from the University of Waterloo, Waterloo, ON, Canada, and the M.Sc. and Ph.D. degrees from the University of California, Berkeley, Berkeley, CA, USA, all in electrical engineering. Since 2010, he has been on the Faculty of the Department of Electronics and Communication Engineering, New York University Polytechnic School of Engineering, Brooklyn, NY, USA.

Philip Schniter (S'92–M'93–SM'05–F'14) received the B.S. and M.S. degrees in electrical engineering from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, and the Ph.D. degree in electrical engineering from Cornell University, Ithaca, NY, USA. Since 2000, he has been on the Faculty of the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH, USA.