Vector Approximate Message Passing for the Generalized Linear Model

Philip Schniter,[*] Sundeep Rangan,[†] and Alyson K. Fletcher[‡]

[*]Dept. of ECE, The Ohio State University, Columbus, OH, 43210. (Email: schniter.1@osu.edu)
[†]Dept. of Electrical and Computer Engineering, New York University, Brooklyn, NY, 11201. (Email: srangan@nyu.edu)
[‡]Depts. of Statistics, Mathematics, and Electrical Engineering, UCLA, Los Angeles, CA 90095. (Email: akfletcher@ucla.edu)

Abstract—The generalized linear model (GLM), where a random vector x is observed through a noisy, possibly nonlinear, function of a linear transform output $z = Ax$, arises in a range of applications such as robust regression, binary classification, quantized compressed sensing, phase retrieval, photon-limited imaging, and inference from neural spike trains. When A is large and i.i.d. Gaussian, the generalized approximate message passing (GAMP) algorithm is an efficient means of MAP or marginal inference, and its performance can be rigorously characterized by a scalar state evolution. For general A, though, GAMP can misbehave. Damping and sequential-updating help to robustify GAMP, but their effects are limited. Recently, a "vector AMP" (VAMP) algorithm was proposed for additive white Gaussian noise channels. VAMP extends AMP's guarantees from i.i.d. Gaussian A to the larger class of rotationally invariant A. In this paper, we show how VAMP can be extended to the GLM. Numerical experiments show that the proposed GLM-VAMP is much more robust to ill-conditioning in A than damped GAMP.

I. INTRODUCTION

We consider the problem of estimating a random vector $x \in \mathbb{R}^N$ from observations $y \in \mathbb{R}^M$ generated as shown in Fig. 1, which is known as the *generalized linear model* (GLM) [1]. Under this model, x has a prior density p_x and y obeys a likelihood function of the form $p(y|x) = p_{y|z}(y|Ax)$, where $A \in \mathbb{R}^{M \times N}$ is a known linear transform and $z \triangleq Ax$ are hidden transform outputs. The conditional density $p_{y|z}$ can be interpreted as a probabilistic measurement channel that accepts a vector z and outputs a random vector y. Although we have assumed real-valued quantities for the sake of simplicity, it is straightforward to generalize the methods in this paper to complex-valued quantities.

A. The Generalized Linear Model

The GLM has many applications in statistics, computer science, and engineering. For example, in *statistical regression* [2], A and y contain experimental features and outcomes, respectively, and x are coefficients that best predict y from A. The relationship between y and the optimal scores $z = Ax$ is then characterized by $p_{y|z}$. In *imaging*-related inverse problems [3], x is an image to recover, A is often Fourier-based, and $p_{y|z}$ models the sensor(s). In *communications* problems
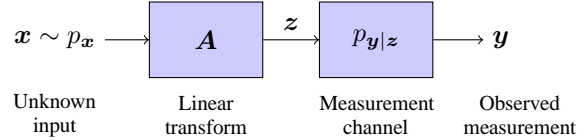
Fig. 1. Generalized Linear Model (GLM): An unknown random vector x is observed through a linear transform A followed by a probabilistic measurement channel $p_{y|z}$, yielding the measured vector y.

[4], x may be a vector of discrete symbols to recover, in which case A is a function of the modulation/demodulation scheme and the propagation physics. Or, x may contain propagation-channel parameters to recover, in which case A is a function of the modulation/demodulation scheme and the pilot symbols. In both cases, $p_{y|z}$ models receiver hardware and interference.

Below we give some examples of the measurement channels $p_{y|z}$ that are encountered in these applications.

- *Robust regression* [5] treats $y = z + w$, and so $p_{y|z}(y|z) = p_w(y - z)$, where p_w is the density of w. The "standard linear model" treats w as additive white Gaussian noise (AWGN) but is not robust to outliers. Robust methods use i.i.d. heavy-tailed models for w.
- *Binary linear classification* [6] can be modeled using $y_m = \text{sgn}(z_m + w_m)$, where $\text{sgn}(v) = 1$ for $v \geq 0$ and $\text{sgn}(v) = -1$ for $v < 0$, and w_m are i.i.d. errors. Gaussian w_m yields the "probit" model and logistic w_m yields the "logistic" model.
- *Quantized compressive sensing* [7] models $y_m = Q(z_m + w_m)$ with i.i.d. noise w_m. Here, $Q(\cdot)$ is a scalar quantizer.
- *Phase retrieval* [8] uses $y_m = |z_m + w_m|$ with $z_m, w_m \in \mathbb{C}$. When w_m is i.i.d. circular Gaussian, $p_{y|z}(y|z) = \prod_{m=1}^{M} p_{y|z}(y_m|z_m)$ with Rician $p_{y|z}(\cdot|z)$ [9].
- *Photon-limited imaging* [10] models the number of photons collected by the sensor, y_m, using a Poisson distribution with rate parameter z_m. Similar models are used when inferring parameters from *neural spike trains* [11].

B. Inference under the Generalized Linear Model

Our goal is to estimate the random vector $x \in \mathbb{R}^N$ from the observed measurements $y \in \mathbb{R}^M$. From the Bayesian viewpoint, there are two major options: *maximum a posteriori (MAP) estimation* or *approximate marginal inference*. The

MAP estimate is the posterior maximizer, i.e.,

$$\widehat{\boldsymbol{x}}_{\mathsf{map}} = \arg\max_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{y}) \stackrel{(a)}{=} \arg\max_{\boldsymbol{x}} \left\{ \ln p(\boldsymbol{y}|\boldsymbol{x}) + \ln p_{\boldsymbol{x}}(\boldsymbol{x}) \right\}$$

$$= \arg\max_{\boldsymbol{x}} \left\{ \ln p_{\boldsymbol{y}|\boldsymbol{z}}(\boldsymbol{y}|\boldsymbol{A}\boldsymbol{x}) + \ln p_{\boldsymbol{x}}(\boldsymbol{x}) \right\}, \quad (1)$$

where (a) is due to the monotonicity of the logarithm and Bayes rule, and (1) is due to the GLM. From (1), we see that MAP estimation is equivalent to solving an optimization problem of the form "$\arg\min_{\boldsymbol{x}} \left\{ l(\boldsymbol{x}) + r(\boldsymbol{x}) \right\}$," with loss function $l(\boldsymbol{x}) \triangleq -\ln p_{\boldsymbol{y}|\boldsymbol{z}}(\boldsymbol{y}|\boldsymbol{A}\boldsymbol{x})$ and regularizer $r(\boldsymbol{x}) \triangleq -\ln p_{\boldsymbol{x}}(\boldsymbol{x})$. Such problems are tractable when the loss and regularization are both convex. For example, with the AWGN channel $p(\boldsymbol{y}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{z}, \boldsymbol{I}/\gamma_w)$ and i.i.d. Laplacian prior $p(x_n) = 0.5\lambda\exp(-\lambda|x_n|)$, MAP estimation reduces to the LASSO [12] problem "$\arg\min_{\boldsymbol{x}} \left\{ \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \frac{\lambda}{\gamma_w}\|\boldsymbol{x}\|_1 \right\}$."

Tractable MAP optimization objectives, however, are often only surrogates for desired optimization objectives, such as minimizing the mean-squared error (MSE) on $\widehat{\boldsymbol{x}}$ or the classification error rate induced by the scores $\widehat{\boldsymbol{z}} = \boldsymbol{A}\widehat{\boldsymbol{x}}$. Likewise, MAP estimation returns a point estimate $\widehat{\boldsymbol{x}}_{\mathsf{map}}$, but reports nothing about the quality of that estimate. Such considerations motivate a different approach, known as *inference*, where the goal is to compute marginal posteriors like $p(x_n|\boldsymbol{y})$ and $p(z_m|\boldsymbol{y})$. If $p(x_n|\boldsymbol{y})$ was known, then the minimum MSE (MMSE) estimate of x_n and the MMSE itself are simply the mean and variance of $p(x_n|\boldsymbol{y})$ [13]. Exact marginal inference, however, is intractable for most problems of interest. Thus, one must usually settle for an approximation.

One well-known approach to approximate marginal inference is through *stochastic simulation* methods like MCMC [14]. But for high dimensional GLMs, such techniques can be computationally expensive and their convergence is difficult to assess. Another approach is *variational inference* [15]. There, the true posterior $p(\boldsymbol{x}|\boldsymbol{y})$ is approximated by a belief $b(\boldsymbol{x})$ that is restricted to a subset of densities $\mathcal{Q}$ chosen as a compromise between fidelity and tractability. For example, the standard "mean field" approach [16] assumes $b(\boldsymbol{x}) = \prod_{n=1}^{N} b_n(x_n)$ while the "expectation propagation" approach in [17] assumes $b(\boldsymbol{x}) = \prod_{m=1}^{M} b_m(\boldsymbol{a}_m^{\mathsf{T}}\boldsymbol{x})$, where $\boldsymbol{a}_m^{\mathsf{T}}$ is the mth row of $\boldsymbol{A}$. Additional constraints on the factors b_m are then needed, which restricts the choice of $p_{y|z}$ and p_x. Common examples include exponential-family, log-concavity, or Gaussian-scale-mixture constraints. Furthermore, high-quality variational inference often require the inversion of an $M \times M$ or $N \times N$ matrix at each iteration, which is impractical for large M, N.

The *approximate message passing* (AMP) algorithm [18], originally proposed for the *standard linear model* (SLM)

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w} \quad \text{with} \quad \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}/\gamma_w), \quad (2)$$

was extended to the GLM in [19]. The resulting *generalized AMP* (GAMP) algorithm is a computationally efficient approach to either MAP or marginal inference that places few restrictions on p_x and $p_{y|z}$. GAMP was originally formulated

assuming a separable prior and measurement channel, i.e.,

$$p_{\boldsymbol{x}}(\boldsymbol{x}) = \prod_{n=1}^{N} p_x(x_n) \quad \text{and} \quad p_{\boldsymbol{y}|\boldsymbol{z}}(\boldsymbol{y}|\boldsymbol{z}) = \prod_{m=1}^{M} p_{y|z}(y_m|z_m), \quad (3)$$

but extensions to non-identical factors and non-separable $p_{\boldsymbol{x}}$ and $p_{\boldsymbol{y}|\boldsymbol{z}}$ have been proposed (e.g., [20]–[23]). Most significantly, when $\boldsymbol{A}$ is large and i.i.d. zero-mean sub-Gaussian and the separability condition (3) holds, (G)AMP is rigorously characterized by a scalar state evolution whose fixed points, when unique, are Bayes-optimal [19,24]. However, (G)AMP can badly misbehave for other $\boldsymbol{A}$. For example, small mean perturbations and/or coefficient correlations in $\boldsymbol{A}$ can cause (G)AMP to diverge [25]. Although damping [25,26] and sequential-updating [27] strategies have been proposed to robustify (G)AMP, they are limited in their effect.

In this paper, we propose a new methodology for both MAP estimation and approximate inference under the GLM. Our method leverages the *vector AMP* (VAMP) [28] framework.

II. VAMP FOR THE STANDARD LINEAR MODEL

We first review the VAMP algorithm, which extends SLM-based AMP from i.i.d. sub-Gaussian $\boldsymbol{A}$ to "right-rotationally invariant" (RRI) $\boldsymbol{A}$. RRI random matrices are described by an SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathsf{T}}$ with $\boldsymbol{V}$ uniformly distributed over the group of orthogonal matrices, allowing arbitrary deterministic $\boldsymbol{U}$ and $\boldsymbol{S}$. It was shown in [28] that, with large RRI $\boldsymbol{A}$, VAMP can be rigorously characterized by a scalar state evolution whose fixed points agree with the replica prediction of MMSE. Numerical experiments in [28] suggest that VAMP performs very close to the replica prediction even at moderate dimensions and with strongly non-zero-mean or ill-conditioned $\boldsymbol{A}$. Such robust behavior is not observed with the S-AMP algorithm [29], which enjoys the same fixed points as VAMP but does not reliably converge to those fixed points.

The VAMP algorithm for the SLM (2) is specified in Algorithm 1. There, $\boldsymbol{g}_1(\cdot, \gamma) : \mathbb{R}^N \to \mathbb{R}^N$ is a "denoising" function identical to that used in the (G)AMP algorithm, and $\langle \boldsymbol{g}_1'(\boldsymbol{r}, \gamma) \rangle$ is its divergence at $\boldsymbol{r}$, i.e.,

$$\langle \boldsymbol{g}_i'(\boldsymbol{r}, \gamma) \rangle = \frac{1}{N} \operatorname{tr}\left\{ \frac{\partial \boldsymbol{g}_i(\boldsymbol{r}, \gamma)}{\partial \boldsymbol{r}} \right\} \quad \text{for} \quad i = 1, 2. \quad (4)$$

Under a separable prior, as in (3), VAMP could be configured for approximate marginal inference by choosing $\boldsymbol{g}_1$ as

$$[\boldsymbol{g}_1(\boldsymbol{r}, \gamma)]_n = \int_{\mathbb{R}} x_n \, b(x_n; r_n, \gamma) \, \mathrm{d}x_n \quad (5)$$

$$b(x_n; r_n, \gamma) \propto p_x(x_n)\mathcal{N}(x_n; r_n, 1/\gamma), \quad (6)$$

where $b(x_n; [\boldsymbol{r}_{1k}]_n, \gamma_{1k})$ is VAMP's iteration-k approximation of the marginal posterior $p(x_n|\boldsymbol{y})$. Likewise, VAMP can be configured for MAP inference by choosing $\boldsymbol{g}_1$ as

$$[\boldsymbol{g}_1(\boldsymbol{r}, \gamma)]_n = \arg\max_{x_n} b(x_n; r_n, \gamma). \quad (7)$$

Non-separable priors $p_{\boldsymbol{x}}$ are implicitly supported by Algorithm 1, although the simpler Monte-Carlo divergence approx-

Algorithm 1 VAMP for the SLM

Require: LMMSE estimator $g_2(r_{2k}, \gamma_{2k})$ from (10), denoiser $g_1(\cdot, \gamma_{1k})$, and number of iterations K.

1: Select initial r_{10} and $\gamma_{10} \geq 0$.
2: **for** $k = 0, 1, \ldots, K$ **do**
3:    // Denoising
4:    $\widehat{x}_{1k} = g_1(r_{1k}, \gamma_{1k}), \quad \alpha_{1k} = \langle g_1'(r_{1k}, \gamma_{1k}) \rangle$
5:    $r_{2k} = (\widehat{x}_{1k} - \alpha_{1k} r_{1k})/(1 - \alpha_{1k})$
6:    $\gamma_{2k} = \gamma_{1k}(1 - \alpha_{1k})/\alpha_{1k}$
7:    // LMMSE estimation
8:    $\widehat{x}_{2k} = g_2(r_{2k}, \gamma_{2k}), \quad \alpha_{2k} = \langle g_2'(r_{2k}, \gamma_{2k}) \rangle$
9:    $r_{1,k+1} = (\widehat{x}_{2k} - \alpha_{2k} r_{2k})/(1 - \alpha_{2k})$
10:    $\gamma_{1,k+1} = \gamma_{2k}(1 - \alpha_{2k})/\alpha_{2k}$
11: **end for**
12: Return $\widehat{x}_{1K}$.

Algorithm 2 VAMP for the GLM

Require: LMMSE estimators g_{x2} and g_{z2} from (15) or (16), denoisers g_{x1} and g_{z1}, and number of iterations K.

1: Select initial $r_{10}, p_{10}, \gamma_{10} > 0, \tau_{10} > 0$.
2: **for** $k = 0, 1, \ldots, K$ **do**
3:    // Denoising x
4:    $\widehat{x}_{1k} = g_{x1}(r_{1k}, \gamma_{1k}), \quad \alpha_{1k} = \langle g_{x1}'(r_{1k}, \gamma_{1k}) \rangle$
5:    $r_{2k} = (\widehat{x}_{1k} - \alpha_{1k} r_{1k})/(1 - \alpha_{1k})$
6:    $\gamma_{2k} = \gamma_{1k}(1 - \alpha_{1k})/\alpha_{1k}$
7:    // Denoising z
8:    $\widehat{z}_{1k} = g_{z1}(p_{1k}, \tau_{1k}), \quad \beta_{1k} = \langle g_{z1}'(p_{1k}, \tau_{1k}) \rangle$
9:    $p_{2k} = (\widehat{z}_{1k} - \beta_{1k} p_{1k})/(1 - \beta_{1k})$
10:    $\tau_{2k} = \tau_{1k}(1 - \beta_{1k})/\beta_{1k}$
11:    // LMMSE estimation of x
12:    $\widehat{x}_{2k} = g_{x2}(r_{2k}, p_{2k}, \gamma_{2k}, \tau_{2k}), \quad \alpha_{2k} = \langle g_{x2}'(\ldots) \rangle$
13:    $r_{1,k+1} = (\widehat{x}_{2k} - \alpha_{2k} r_{2k})/(1 - \alpha_{2k})$
14:    $\gamma_{1,k+1} = \gamma_{2k}(1 - \alpha_{2k})/\alpha_{2k}$
15:    // LMMSE estimation of z
16:    $\widehat{z}_{2k} = g_{z2}(r_{2k}, p_{2k}, \gamma_{2k}, \tau_{2k}), \quad \beta_{2k} = \langle g_{z2}'(\ldots) \rangle$
17:    $p_{1,k+1} = (\widehat{z}_{2k} - \beta_{2k} p_{2k})/(1 - \beta_{2k})$
18:    $\tau_{1,k+1} = \tau_{2k}(1 - \beta_{2k})/\beta_{2k}$
19: **end for**
20: Return $\widehat{x}_{1K}$.

---

imation from [23, Section V.B] has also been observed to work well in VAMP [30]. In general, $g_1(\cdot, \gamma)$ can be interpreted as "denoising" the AWGN-corrupted pseudo-measurement $r_{1k} = x + \mathcal{N}(0, I/\gamma_{1k})$ using prior knowledge of x.

The function $g_2(r_{2k}, \gamma_{2k}) : \mathbb{R}^N \to \mathbb{R}^N$ in line 8 of Algorithm 1 performs LMMSE estimation of x from the AWGN-corrupted measurements (2) under the pseudo-prior $x \sim \mathcal{N}(r_{2k}, I/\gamma_{2k})$, i.e.,

$$g_2(r_{2k}, \gamma_{2k}) := (\gamma_w A^\mathsf{T} A + \gamma_{2k} I)^{-1}(\gamma_w A^\mathsf{T} y + \gamma_{2k} r_{2k}) \quad (8)$$

$$\langle g_2'(r_{2k}, \gamma_{2k}) \rangle = \gamma_{2k} N^{-1} \operatorname{tr}\left[(\gamma_w A^\mathsf{T} A + \gamma_{2k} I)^{-1}\right] \quad (9)$$

The per-iteration matrix inverse in (8)-(9) can be avoided by precomputing the SVD $A = USV^\mathsf{T}$, after which

$$g_2(r_{2k}, \gamma_{2k}) = V D_k(\widetilde{y} + \gamma_{2k} V^\mathsf{T} r_{2k}) \quad (10)$$

$$\langle g_2'(r_{2k}, \gamma_{2k}) \rangle = \frac{1}{N} \sum_{n=1}^{N} \frac{\gamma_{2k}}{\gamma_w s_n^2 + \gamma_{2k}}, \quad (11)$$

where $\widetilde{y} = \gamma_w S^\mathsf{T} U^\mathsf{T} y$ and D_k is the $N \times N$ diagonal matrix with $[D_k]_{nn} = (\gamma_w s_n^2 + \gamma_{2k})^{-1}$. Since $\widetilde{y}$ can be precomputed, the complexity of VAMP is dominated by two matrix-vector multiplies per iteration, just like AMP.

III. VAMP FOR THE GENERALIZED LINEAR MODEL

Algorithm 1 applies VAMP to the SLM. We now show how a small modification allows its application to the GLM. Our approach exploits the equivalence relationship

$$z = Ax \iff 0 = \begin{bmatrix} A & -I \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} \iff \overline{y} = \overline{A} \overline{x} + \overline{w}, \quad (12)$$

where $\overline{y} \triangleq 0$, $\overline{A} \triangleq \begin{bmatrix} A & -I \end{bmatrix}$, $\overline{x} \triangleq \begin{bmatrix} x \\ z \end{bmatrix}$, and $\overline{w} \sim \mathcal{N}(0, I/\gamma_e)$ as $\gamma_e \to \infty$. Comparing (12) to (2), we see that our GLM can be expressed as an SLM where $\overline{x}$ has two sub-vectors, the first in $\mathbb{R}^N$ and the second in $\mathbb{R}^M$. Because these two sub-vectors can behave very differently, we propose a modified VAMP that separately tracks the precision of each. The result, shown in Algorithm 2, can be interpreted as an instance of

the more general "GEC" algorithm from [31] with a particular diagonalization operator.

In the sequel, we will use $\widehat{x}_{ik} \in \mathbb{R}^N$ and $\widehat{z}_{ik} \in \mathbb{R}^M$ to denote the two sub-vectors of the output of g_i at iteration k (for $i = 1, 2$), and we will use $r_{ik} \in \mathbb{R}^N$ and $p_{ik} \in \mathbb{R}^M$ to denote the two sub-vectors of the input to g_i. As in SLM-based VAMP, we will use the pseudo-measurement model $r_{1k} = x + \mathcal{N}(0, I/\gamma_{1k})$ when denoising x and the pseudo-prior $x \sim \mathcal{N}(r_{2k}, I/\gamma_{2k})$ for LMMSE estimation of x. Likewise, we will use pseudo-measurements $p_{1k} = z + \mathcal{N}(0, I/\tau_{1k})$ when denoising z and the pseudo-prior $z \sim \mathcal{N}(p_{2k}, I/\tau_{2k})$ for LMMSE estimation of z. A rigorous justification of these models is postponed for future work.

The independence between the random variables x and the random variables y conditioned on z implies that the function g_1 decouples across the two sub-vectors. That is, we can write $\widehat{x}_{1k} = g_{x1}(r_{1k}, \gamma_{1k})$ and $\widehat{z}_{1k} = g_{z1}(p_{1k}, \tau_{1k})$ for denoisers $g_{x1}(\cdot, \gamma_{1k}) : \mathbb{R}^N \to \mathbb{R}^N$ and $g_{z1}(\cdot, \tau_{1k}) : \mathbb{R}^M \to \mathbb{R}^M$. The construction of g_{x1} remains the same as described in Section II, and the construction of g_{z2} is similar but with $p_{y|z}(y|\cdot)$ replacing $p_x(\cdot)$. Lines 5-6 and 9-10 of Algorithm 2 follow directly from lines 5-6 of Algorithm 1.

Lines 12-18 of Algorithm 2 implement LMMSE estimation of $\overline{x} = \begin{bmatrix} x \\ z \end{bmatrix}$ under the SLM in (12) and the pseudo-prior

$$\overline{x} = \begin{bmatrix} x \\ z \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} r_{2k} \\ p_{2k} \end{bmatrix}, \begin{bmatrix} I/\gamma_{2k} & \\ & I/\tau_{2k} \end{bmatrix}\right). \quad (13)$$

Because the likelihood and prior are both Gaussian, the

LMMSE estimate is equivalent to the MAP estimate

$$\arg\max_{\overline{\boldsymbol{x}}} p(\overline{\boldsymbol{x}}|\overline{\boldsymbol{y}}) = \arg\min_{\overline{\boldsymbol{x}}} \{-\ln p(\overline{\boldsymbol{y}}|\overline{\boldsymbol{x}}) - \ln p(\overline{\boldsymbol{x}})\} \qquad (14)$$
$$= \arg\min_{\boldsymbol{x},\boldsymbol{z}} \gamma_e\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{z}\|_2^2 + \gamma_{2k}\|\boldsymbol{r}_{2k} - \boldsymbol{x}\|_2^2 + \tau_{2k}\|\boldsymbol{p}_{2k} - \boldsymbol{z}\|_2^2.$$

Zeroing the gradients w.r.t. $\boldsymbol{x}$ and $\boldsymbol{z}$, taking $\gamma_e \to \infty$, and substituting the SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\mathsf{T}$ into the result, we get

$$\boldsymbol{g}_{x2}(\boldsymbol{r}_{2k}, \boldsymbol{p}_{2k}, \gamma_{2k}, \tau_{2k}) = \boldsymbol{V}\boldsymbol{D}_k(\tau_{2k}\boldsymbol{S}^\mathsf{T}\boldsymbol{U}^\mathsf{T}\boldsymbol{p}_{2k} + \gamma_{2k}\boldsymbol{V}^\mathsf{T}\boldsymbol{r}_{2k})$$
$$\boldsymbol{g}_{z2}(\boldsymbol{r}_{2k}, \boldsymbol{p}_{2k}, \gamma_{2k}, \tau_{2k}) = \boldsymbol{A}\boldsymbol{g}_{x2}(\boldsymbol{r}_{2k}, \boldsymbol{p}_{2k}, \gamma_{2k}, \tau_{2k}), \qquad (15)$$

where $\boldsymbol{D}_k$ is an $N \times N$ diagonal matrix such that $[\boldsymbol{D}_k]_{nn} \triangleq (\tau_{2k}s_n^2 + \gamma_{2k})^{-1}$. An alternative expression for $\boldsymbol{g}_{x2}$ is

$$\boldsymbol{g}_{x2}(\boldsymbol{r}_{2k}, \boldsymbol{p}_{2k}, \gamma_{2k}, \tau_{2k})$$
$$= \boldsymbol{r}_{2k} + \boldsymbol{V}\boldsymbol{S}^\mathsf{T}\left(\frac{\gamma_{2k}}{\tau_{2k}}\boldsymbol{I} + \boldsymbol{S}\boldsymbol{S}^\mathsf{T}\right)^{-1}(\boldsymbol{U}^\mathsf{T}\boldsymbol{p}_{2k} - \boldsymbol{S}\boldsymbol{V}^\mathsf{T}\boldsymbol{r}_{2k}). \quad (16)$$

Both (15) and (16) are derived in the Appendix.

Recalling the definition of the divergence in (4), we see that α_{2k} from line 12 of Algorithm 2 equals N^{-1} times the trace of the Jacobian $\partial\boldsymbol{g}_{x2}/\partial\boldsymbol{r}_{2k} = \gamma_{2k}\boldsymbol{V}\boldsymbol{D}_k\boldsymbol{V}^\mathsf{T}$, and so (16) gives

$$\alpha_{2k} = \langle\boldsymbol{g}'_{x2}(\boldsymbol{r}_{2k}, \boldsymbol{p}_{2k}, \gamma_{2k}, \tau_{2k})\rangle = \frac{1}{N}\sum_{n=1}^{N}\frac{\gamma_{2k}}{\tau_{2k}s_n^2 + \gamma_{2k}}. \quad (17)$$

Similarly, β_{2k} from line 16 of Algorithm 2 is M^{-1} times the trace of the Jacobian $\partial\boldsymbol{g}_{z2}/\partial\boldsymbol{p}_{2k} = \tau_{2k}\boldsymbol{S}\boldsymbol{D}_k\boldsymbol{S}^\mathsf{T}$, and so

$$\beta_{2k} = \langle\boldsymbol{g}'_{z2}(\boldsymbol{r}_{2k}, \boldsymbol{p}_{2k}, \gamma_{2k}, \tau_{2k})\rangle \qquad (18)$$
$$= \frac{1}{M}\sum_{n=1}^{N}\frac{\tau_{2k}s_n^2}{\tau_{2k}s_n^2 + \gamma_{2k}} = \frac{M}{N}(1 - \alpha_{2k}). \qquad (19)$$

The above explains lines 12 and 16 of Algorithm 2. Lines 13-14 and 17-18 of Algorithm 2 follow directly from lines 9-10 of Algorithm 1.

IV. Numerical Experiments

We now show the results of a numerical experiment on *one-bit compressed sensing*, where the goal was to recover the sparse signal $\boldsymbol{x} \in \mathbb{R}^N$ from measurements

$$y_m = \mathrm{sgn}([\boldsymbol{A}\boldsymbol{x} + \boldsymbol{w}]_m) \quad \text{for} \quad m = 1, \ldots, M. \qquad (20)$$

For our experiment, we drew $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}/\gamma_w)$ and we constructed $\boldsymbol{x}$ with 16 non-zero coefficients whose amplitudes were drawn i.i.d. $\mathcal{N}(0, 1)$ and whose indices were drawn independently and uniformly at random. Also, we used $N = 512$ and $M = 2048$, and we adjusted γ_w to achieve a signal-to-noise ratio $\mathrm{E}\{\|\boldsymbol{A}\boldsymbol{x}\|^2\}/\mathrm{E}\{\|\boldsymbol{w}\|^2\} = 40$ dB.

Following [25], we constructed $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ from the singular value decomposition (SVD) $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\mathsf{T}$, where orthogonal matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ were drawn uniformly with respect to the Haar measure. That is, $\boldsymbol{A}$ was rotationally invariant. The singular values s_n were a geometric series, i.e., $s_n/s_{n-1} = \rho \; \forall n > 1$, with ρ and s_1 chosen to achieve a desired condition number $\kappa(\boldsymbol{A}) \triangleq s_1/s_{\min(M,N)}$ with $\|\boldsymbol{A}\|_F^2 = N$. It was shown in [25,26] that standard AMP (and even damped AMP) diverges when the matrix $\boldsymbol{A}$
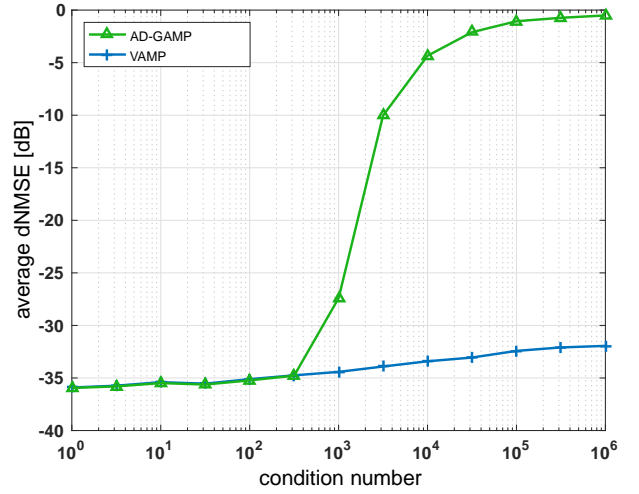

Fig. 2. Debiased NMSE versus condition number $\kappa(\boldsymbol{A})$ at the final algorithm iteration, averaged over 500 realizations.

has a sufficiently high condition number. Thus, this matrix-generation model provides an effective test for the stability of AMP methods. Recovery performance was assessed using "debiased" normalized mean-squared error (dNMSE), $\min_{c\in\mathbb{R}} \|c\widehat{\boldsymbol{x}} - \boldsymbol{x}\|^2/\|\boldsymbol{x}\|^2$. The debiasing was used because the measurement channel discards amplitude information.

Figure 2 plots the average dNMSE achieved by VAMP and by the adaptively damped (AD) GAMP algorithm from [25] versus condition number $\kappa(\boldsymbol{A})$. The dNMSE was evaluated for $\kappa(\boldsymbol{A})$ ranging from 1 (i.e., row-orthogonal) to 10^6 (i.e., highly ill-conditioned $\boldsymbol{A}$), and averaged over 500 independent draws of $\boldsymbol{A}$, $\boldsymbol{x}$, and $\boldsymbol{w}$. For this experiment, VAMP perfectly knew the prior $p_{\boldsymbol{x}}$ and measurement-channel $p_{\boldsymbol{y}|\boldsymbol{z}}$ (although if not the technique in [32] could be used for automatic tuning) and it was initialized using $\boldsymbol{r}_{10} = \boldsymbol{0}$, $\boldsymbol{p}_{10} = \boldsymbol{0}$, $\gamma_{10} = 10^{-8}$, and $\tau_{10} = 10^{-8}$. The figure shows that AD-GAMP accurately recovered $\boldsymbol{x}$ for $\kappa(\boldsymbol{A}) < 10^3$ but failed at higher condition numbers. By contrast, VAMP accurately recovered $\boldsymbol{x}$ over the full tested range of $\kappa(\boldsymbol{A})$.

Figure 3 plots the average dNMSE versus iteration for condition numbers $\kappa(\boldsymbol{A}) \in \{1, 316, 10^6\}$. The figures show that, for the range of $\kappa(\boldsymbol{A})$ where AD-GAMP accurately recovers $\boldsymbol{x}$, VAMP converges faster: in about 10 iterations compared to 30-40 for AD-GAMP. Meanwhile, at the extreme case of $\kappa(\boldsymbol{A}) = 10^6$, VAMP converges in less than 20 iterations. Thus, these experiments suggest that the convergence speed of VAMP is relatively insensitive to the condition number of large, rotationally invariant $\boldsymbol{A}$.

Appendix

To derive (15)-(16), we zero the gradient of the cost in (14) w.r.t. $\boldsymbol{x}$ and $\boldsymbol{z}$ at $\widehat{\boldsymbol{x}}_{2k}$ and $\widehat{\boldsymbol{z}}_{2k}$, yielding the equations

$$\boldsymbol{0} = \gamma_e\boldsymbol{A}^\mathsf{T}(\boldsymbol{A}\widehat{\boldsymbol{x}}_{2k} - \widehat{\boldsymbol{z}}_{2k}) + \gamma_{2k}(\widehat{\boldsymbol{x}}_{2k} - \boldsymbol{r}_{2k}) \qquad (21)$$
$$\boldsymbol{0} = \gamma_e(\widehat{\boldsymbol{z}}_{2k} - \boldsymbol{A}\widehat{\boldsymbol{x}}_{2k}) + \tau_{2k}(\widehat{\boldsymbol{z}}_{2k} - \boldsymbol{p}_{2k}), \qquad (22)$$
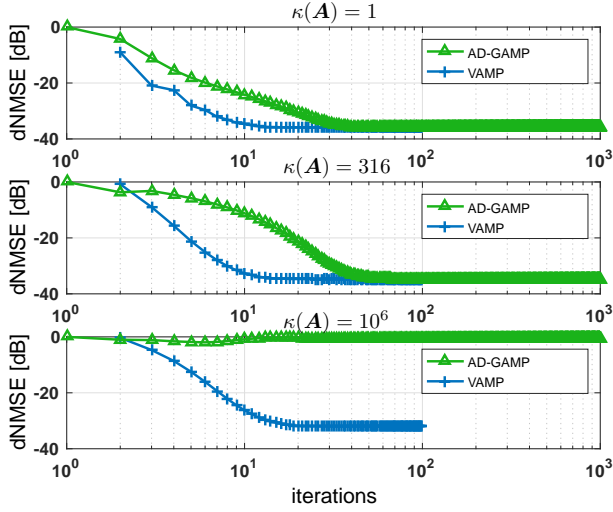
Fig. 3. Debiased NMSE versus iteration k at several condition numbers $\kappa(\boldsymbol{A}) = 1$ in (a), $\kappa(\boldsymbol{A}) = 316.23$ in (b), and $\kappa(\boldsymbol{A}) = 10^6$ in (c), averaged over 500 realizations.

which can be rewritten as

$$\begin{bmatrix} \gamma_{2k}\boldsymbol{r}_{2k} \\ \tau_{2k}\boldsymbol{p}_{2k} \end{bmatrix} = \begin{bmatrix} \gamma_e\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} + \gamma_{2k}\boldsymbol{I} & -\gamma_e\boldsymbol{A}^{\mathsf{T}} \\ -\gamma_e\boldsymbol{A} & (\tau_{2k}+\gamma_e)\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{x}}_{2k} \\ \widehat{\boldsymbol{z}}_{2k} \end{bmatrix}. \quad (23)$$

Inverting the block matrix in (23) via the Schur complement $\boldsymbol{Q} \triangleq \gamma_e\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} + \gamma_{2k}\boldsymbol{I} - \frac{\gamma_e^2}{\tau_{2k}+\gamma_e}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} = \frac{\gamma_e\tau_{2k}}{\tau_{2k}+\gamma_e}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} + \gamma_{2k}\boldsymbol{I}$ gives (after temporarily suppressing the "k" index)

$$\begin{bmatrix} \widehat{\boldsymbol{x}}_2 \\ \widehat{\boldsymbol{z}}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}^{-1} & \frac{\gamma_e}{\tau_2+\gamma_e}\boldsymbol{Q}^{-1}\boldsymbol{A}^{\mathsf{T}} \\ \frac{\gamma_e}{\tau_2+\gamma_e}\boldsymbol{A}\boldsymbol{Q}^{-1} & \frac{1}{\tau_2+\gamma_e}(\boldsymbol{I} + \frac{\gamma_e^2}{\tau_2+\gamma_e}\boldsymbol{A}\boldsymbol{Q}^{-1}\boldsymbol{A}^{\mathsf{T}}) \end{bmatrix} \begin{bmatrix} \gamma_2\boldsymbol{r}_2 \\ \tau_2\boldsymbol{p}_2 \end{bmatrix}.$$

Taking $\gamma_e \to \infty$ then gives $\boldsymbol{Q} = \tau_{2k}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} + \gamma_{2k}\boldsymbol{I}$ and

$$\begin{bmatrix} \widehat{\boldsymbol{x}}_{2k} \\ \widehat{\boldsymbol{z}}_{2k} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}^{-1} & \boldsymbol{Q}^{-1}\boldsymbol{A}^{\mathsf{T}} \\ \boldsymbol{A}\boldsymbol{Q}^{-1} & \boldsymbol{A}\boldsymbol{Q}^{-1}\boldsymbol{A}^{\mathsf{T}} \end{bmatrix} \begin{bmatrix} \gamma_{2k}\boldsymbol{r}_{2k} \\ \tau_{2k}\boldsymbol{p}_{2k} \end{bmatrix} \quad (24)$$

$$= \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{A} \end{bmatrix} \left(\tau_{2k}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} + \gamma_{2k}\boldsymbol{I} \right)^{-1} \left(\gamma_{2k}\boldsymbol{r}_{2k} + \tau_{2k}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{p}_{2k} \right). \quad (25)$$

Plugging the SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathsf{T}}$ into (25) yields (15). An alternative expression results from the matrix inversion lemma:

$$\widehat{\boldsymbol{x}}_{2k} = \boldsymbol{r}_{2k} + \boldsymbol{A}^{\mathsf{T}} \left(\frac{\gamma_{2k}}{\tau_{2k}}\boldsymbol{I} + \boldsymbol{A}\boldsymbol{A}^{\mathsf{T}} \right)^{-1} \left(\boldsymbol{p}_{2k} - \boldsymbol{A}\boldsymbol{r}_{2k} \right), \quad (26)$$

and plugging the SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathsf{T}}$ into (26) yields (16).

REFERENCES

[1] P. McCullagh and J. A. Nelder, *Generalized Linear Models*. London: Chapman & Hall/CRC, 2nd ed., 1989.

[2] A. Gelman and J. Hill, *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, 2006.

[3] A. Ribés and F. Schmitt, "Linear inverse problems in imaging," *IEEE Signal Process. Mag.*, vol. 25, no. 4, pp. 84–99, 2008.

[4] F. Hlawatsch and G. Matz, *Wireless Communications over Rapidly Time-Varying Channels*. New York, NY: Academic, 2011.

[5] P. J. Huber and E. M. Ronchetti, *Robust Statistics*. Somerset: Wiley, 2nd ed., 2009.

[6] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2007.

[7] U. S. Kamilov, V. K. Goyal, and S. Rangan, "Message-passing dequantization with applications to compressed sensing," *IEEE Trans. Signal Process.*, vol. 60, pp. 6270–6281, Dec. 2012.

[8] R. P. Millane, "Recent advances in phase retrieval," *The Int. Soc. Optical Eng.*, vol. 6316, 2006.

[9] P. Schniter and S. Rangan, "Compressive phase retrieval via generalized approximate message passing," *IEEE Trans. Signal Process.*, vol. 63, pp. 1043–1055, Feb. 2015.

[10] R. M. Willett, R. F. Marcia, and J. M. Nichols, "Compressed sensing for practical optical imaging systems: A tutorial," *Optical Eng.*, vol. 50, July 2011.

[11] A. K. Fletcher, S. Rangan, L. R. Varshney, and A. Bhargava, "Neural reconstruction with approximate message passing (NeuRAMP)," in *Proc. Neural Inform. Process. Syst. Conf.*, pp. 2555–2563, 2011.

[12] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist. Soc. B*, vol. 58, no. 1, pp. 267–288, 1996.

[13] H. V. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer, 2nd ed., 1994.

[14] M. Pereyra, P. Schniter, E. Chouzenoux, J.-C. Pesquet, J.-Y. Tourneret, A. Hero, and S. McLaughlin, "A survey of stochastic simulation and optimization methods in signal processing," *IEEE J. Sel. Topics Signal Process.*, vol. 10, pp. 1–14, 2016.

[15] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Found. Trends Mach. Learn.*, vol. 1, May 2008.

[16] G. Parisi, *Statistical Field Theory*. Reading, MA: Addison-Wesley, 1988.

[17] M. W. Seeger, S. Gerwinn, and M. Bethge, "Bayesian inference for sparse generalized linear models," in *Proc. European Conf. on Mach. Learning*, pp. 298–309, 2007.

[18] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing," *Proc. Nat. Acad. Sci.*, vol. 106, pp. 18914–18919, Nov. 2009.

[19] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," in *Proc. IEEE Int. Symp. Inform. Thy.*, pp. 2168–2172, Aug. 2011. (full version at *arXiv:1010.5141*).

[20] P. Schniter, "Turbo reconstruction of structured sparse signals," in *Proc. Conf. Inform. Science & Syst.*, (Princeton, NJ), pp. 1–6, Mar. 2010.

[21] S. Rangan, A. K. Fletcher, V. K. Goyal, and P. Schniter, "Hybrid generalized approximate message passing with applications to structured sparsity," in *Proc. IEEE Int. Symp. Inform. Thy.*, pp. 1236–1240, July 2012. (full version at *arXiv:1111.2581*).

[22] M. Borgerding, P. Schniter, J. Vila, and S. Rangan, "Generalized approximate message passing for cosparse analysis compressive sensing," in *Proc. IEEE Int. Conf. Acoust. Speech & Signal Process.*, 2015.

[23] C. A. Metzler, A. Maleki, and R. G. Baraniuk, "From denoising to compressed sensing," *IEEE Trans. Inform. Theory*, vol. 62, no. 9, pp. 5117–5144, 2016.

[24] A. Javanmard and A. Montanari, "State evolution for general approximate message passing algorithms, with applications to spatial coupling," *Inform. Inference*, vol. 2, no. 2, pp. 115–144, 2013.

[25] 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.*, pp. 2021–2025, 2015.

[26] S. Rangan, P. Schniter, and A. Fletcher, "On the convergence of generalized approximate message passing with arbitrary matrices," in *Proc. IEEE Int. Symp. Inform. Thy.*, pp. 236–240, July 2014. (full version at *arXiv:1402.3210*).

[27] A. Manoel, F. Krzakala, E. W. Tramel, and L. Zdeborová, "Swept approximate message passing for sparse estimation," in *Proc. Int. Conf. Mach. Learning*, pp. 1123–1132, 2015.

[28] S. Rangan, P. Schniter, and A. K. Fletcher, "Vector approximate message passing," *arXiv:1610.03082*, 2016.

[29] B. Çakmak, O. Winther, and B. H. Fleury, "S-AMP: Approximate message passing for general matrix ensembles," in *Proc. Inform. Theory Workshop*, pp. 192–196, 2014.

[30] P. Schniter, S. Rangan, and A. K. Fletcher, "Denoising-based vector approximate message passing," in *Proc. Intl. Biomed. Astronom. Signal Process. (BASP) Frontiers Workshop*, 2017.

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

[32] A. K. Fletcher and P. Schniter, "Learning and free energies for vector approximate message passing," *arXiv:1602.08207*, 2016.