Deep Unfolded AMP for Short Block-Length SPARC over AWGN Channels

Sai Dinesh Kancharana*, VSV Sandeep†, Arun Pachai Kannu‡
Department of Electrical Engineering
Indian Institute of Technology Madras, Chennai - 600036, India
Email: *ee20d401@smail.iitm.ac.in,†ee22d004@smail.iitm.ac.in,‡arunpachai@ee.iitm.ac.in

***Abstract*—Sparse regression codes (SPARC) combined with approximate message passing (AMP) decoding provide a computationally efficient framework for reliable communication over additive white Gaussian noise (AWGN) channels. While SPARCs achieve capacity with AMP decoding at asymptotic block lengths, their performance degrades at short block lengths due to the mismatch between finite-length behaviour and the large-system assumptions underlying AMP decoding. In this work, we develop a deep unfolded version of AMP, referred to as learned-AMP (LAMP), for SPARC decoding in the short block-length regime. By considering AMP iterations as layers of neural network, the proposed approach introduces state evolution parameters and linear projections of classical AMP as trainable parameters, which are optimized using data-driven training. We investigate multiple LAMP variants that differ in training strategies and parametrisations. Simulation results demonstrate that short block-length SPARC achieve significantly lower block error rate (BLER) with LAMP decoding, particularly with incremental training, compared to classical AMP decoding at high code rate.**

***Index Terms*—Sparse regression codes, approximate message passing, deep unfolding, learned-AMP, short block lengths, AWGN channel.**

I. Introduction

The emergence of next-generation 6G wireless networks has brought renewed attention to the stringent reliability and latency requirements of time-critical applications. Applications such as autonomous vehicles, industrial control systems, and telemedicine demand communication links that operate with latencies as low as 0.1-1ms and with extremely high reliability, a paradigm referred to as hyper reliable low latency communication (HRLLC) [1]–[3]. Similarly, in massive machine type communication (mMTC) and vehicle-to-everything (V2X) scenarios, devices communicate with short, bursty payloads under tight delay constraints, making short block-length codes well-suited for such applications. Achieving both high reliability and strong finite block-length performance remains a key challenge for next-generation communication systems, motivating the exploration of alternative coding frameworks.

Sparse regression codes, also called SPARC, are a class of structured error control codes originally proposed for the AWGN channel [4], [5]. A SPARC codeword is formed as a sparse linear combination of columns drawn from a random Gaussian dictionary matrix, formulating decoding as a structured sparse recovery problem at the intersection of channel coding and compressed sensing. SPARC have been shown to exhibit strong finite block-length performance in AWGN channels [6], as well as in coherent [7] and non-coherent [8] flat-fading scenarios.

Approximate message passing [9], [10] is a practical decoder for SPARC, offering low per-iteration complexity and a precise asymptotic performance characterisation through state evolution. Authors in [11] demonstrated that SPARC with AMP decoding are capacity-achieving in the asymptotic block length regime under exponential power allocation.

However, the theoretical guarantees of AMP rely on assumptions such as the large system limit, and i.i.d Gaussian dictionary matrices that do not hold in practical short block length applications. State evolution predictions can deviate from empirical behaviour at finite lengths. Structured deterministic matrices, such as Hadamard, DFT or Gold sequence-based matrices with finite alphabet entries, which are employed to reduce the computational complexity, further violate the assumptions underpinning the Onsager correction in AMP. Consequently, the analytically prescribed parameters of classical AMP may be suboptimal for short block-length SPARC.

Deep unfolding [12] offers a principled approach to addressing these limitations. The iterations of an iterative algorithm, such as the AMP decoder, are interpreted as layers of a feed-forward neural network, and fixed algorithmic parameters are replaced with trainable counterparts that can be optimised directly from data. This paradigm was extended to AMP-based algorithms in [13] for general sparse linear inverse problems. In this work, we introduce the Learned-AMP (LAMP) framework for SPARC decoding and propose several training variants that account for finite block-length effects and non-i.i.d matrices. We demonstrate through simulations using Gold sequence-based SPARC that LAMP consistently and significantly outperforms classical AMP in short block-length, high code rate regimes.

Notation: Scalars are lower case letter x, vectors, and matrices are in bold lower case $\mathbf{x}$ and upper case $\mathbf{X}$, respectively. $\mathbf{x}_i$ represents the i^{th} column of the matrix $\mathbf{X}$. $(.)^T$ indicates transpose. $\|\mathbf{x}\|$ denotes the ℓ_2 norm of $\mathbf{x}$. $\mathcal{N}(\mu, \sigma^2)$ represents the Gaussian distribution with mean μ and variance σ^2. $[M]$ denotes the set $\{1, 2, \ldots . M\}$. $|\mathcal{X}|$ is the size of the set $\mathcal{X}$. $\lceil \cdot \rceil$ represents the ceil operator. $\mathbb{1}$ denotes indicator function. $\mathbb{E}[\cdot]$ denotes the expectation operator.

II. System model and background

A. SPARC Encoding

SPARC are defined by a dictionary matrix $\mathbf{A} \in \mathbb{R}^{N \times L}$ where $L > N$, with all columns being *unit norm*. The columns of $\mathbf{A}$ are partitioned into K disjoint sections, using an optimal section partitioning algorithm from [6]. The number of columns in section k is denoted by L_k (which is a power of 2), and the set of columns in section k is denoted by $\mathcal{A}_k$. $\mathrm{sec}(j)$ denotes the set of columns in the section containing the column $\mathbf{a}_j$. For example, if $\mathbf{a}_j$ belongs to section ℓ, then $\mathrm{sec}(j) = \mathcal{A}_\ell$. Based on the information bits, the encoder selects one column from each section $\mathcal{A}_k$, $\forall k \in [K]$, and the codeword is formed as a linear combination of the columns chosen from all sections. This is equivalent to multiplying the dictionary matrix $\mathbf{A}$ by a K-sparse vector β with non-zero values in the indices corresponding to the chosen columns. The nonzero value in section k is set to $\sqrt{NP_k}$, where $\{P_k\}$ is a power allocation satisfying $\sum_{k=1}^{K} P_k = P$. The SPARC codeword is formed as $\mathbf{A}\beta$ and transmitted over the AWGN channel, yielding the received vector

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

where $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbb{I}_N)$. Since each column in $\mathcal{A}_k$ can be indexed using $\log_2 L_k$ bits, the number of bits conveyed by the SPARC codeword is $N_b = \sum_{k=1}^{K} \log_2 L_k$. The dictionary matrix plays a crucial role in determining the sparse recovery performance. It is desirable to construct a dictionary matrix with a large number of columns that exhibit low pairwise correlation. This property is characterised by the mutual coherence of the matrix, defined as the maximum normalised absolute inner product between any two distinct columns of the matrix. In our work, we form the dictionary matrix based on binary Gold code sequences from [14], which are known to have low mutual coherence with $L = (N+1)^2$ columns. We use flat power allocation, i.e., $P_k = P/K, \forall k \in [K]$.

B. Classical AMP decoder

SPARC decoding can be viewed as a structured sparse recovery problem. The classical AMP algorithm iteratively estimates β through linear residual updates and non-linear denoising. The algorithm is characterised by *state evolution* parameters that predict the mean squared error (MSE) between the estimate and the true sparse signal at each iteration. The AMP recursion used for SPARC decoding [11] recovers β from $\mathbf{y}$ through iterative linear and non-linear updates. Let $\hat{\beta}^t$ denote the estimate at iteration t, with initialisation $\hat{\beta}^0 = \mathbf{0}$. Then,

$$\mathbf{z}^t = \mathbf{y} - \mathbf{A}\hat{\beta}^t + \frac{\mathbf{z}^{t-1}}{\tau_t^2}\left(P - \frac{\|\hat{\beta}^t\|^2}{n}\right), \tag{2}$$

$$\hat{\beta}^{t+1} = \eta_t\left(\hat{\beta}^t + \mathbf{A}^T\mathbf{z}^t;\, \tau_t^2\right), \tag{3}$$

where the last term in (2) is the *Onsager correction*, which asymptotically decorrelates the effective observation from the current estimate. Due to the Onsager correction term, the input to the denoiser is asymptotically distributed as true signal corrupted by Gaussian noise with variance equal to the state evolution parameters, i.e.,

$$\hat{\beta}^t + \mathbf{A}^T\mathbf{z}^t \overset{d}{\approx} \beta + \tau_t \mathbf{v}, \tag{4}$$

where τ_t^2 is the state evolution parameter at iteration t and $\mathbf{v} \sim \mathcal{N}(0, \mathbb{I})$. By exploiting the section structure, an MMSE estimator which acts element-wise can be used for the denoiser in (3), which is given by

$$\eta_i(\mathbf{s}) = \sqrt{nP_\ell}\,\frac{\exp\left(s_i\sqrt{nP_\ell}/\tau_t^2\right)}{\sum_{j \in \mathrm{sec}(i)} \exp\left(s_j\sqrt{nP_\ell}/\tau_t^2\right)}. \tag{5}$$

The behaviour of the AMP algorithm can be characterised using *scalar* state evolution parameters, which track the per-iteration MSE in the large-system limit. A rigorous mathematical proof has been given in [15]. For the SPARC AMP decoder, they are calculated as,

$$\tau_{t+1}^2 = \sigma^2 + P(1 - x_{t+1}), \tag{6}$$

where x_{t+1} is the expected fraction of correctly decoded signal power at iteration $t+1$. These parameters are computed offline via Monte Carlo simulation. The total number of iterations is set to

$$T^* = \left\lceil \frac{2C}{\log_2(C/R)} \right\rceil + 1, \tag{7}$$

with $C = \frac{1}{2}\log_2(1 + P/\sigma^2)$ denoting the AWGN channel capacity. After T^* iterations, a hard decision selects the maximum-magnitude index in each section. It has been shown in [11] that the state evolution converges to a fixed point after T^* iterations.

C. Limitations at Short Block Lengths

AMP decoding for SPARC has a low per-iteration complexity and an elegant asymptotic characterisation, which help SPARC achieve capacity in the large system limit. However, the performance of SPARC deteriorate significantly at the short block length of interest. This degradation arises because the classical AMP recursion relies on two strong assumptions [11],

- A large system limit, where the codeword length and number of sections of the dictionary matrix are assumed to be asymptotically large, in which empirical quantities concentrate sharply.
- An i.i.d random Gaussian dictionary matrix, under which the Onsager correction helps decouple the estimates across iterations.

However, for practical low latency applications, the codeword lengths are small, and dictionary matrices with finite alphabet elements, such as Gold or Hadamard sequences, are used for low complexity implementations. Hence, the effective input to the denoiser and the Onsager terms may not follow the ideal Gaussian assumptions. These observations motivate a data-driven refinement of the AMP algorithm, using a deep unfolding approach, where the AMP iterations are interpreted as layers of a neural network. In particular, by learning the state-evolution parameters and linear projection matrix, the decoder can compensate for deviations from the ideal AMP assumptions while preserving the interpretability and low-complexity structure of the classical AMP algorithm.

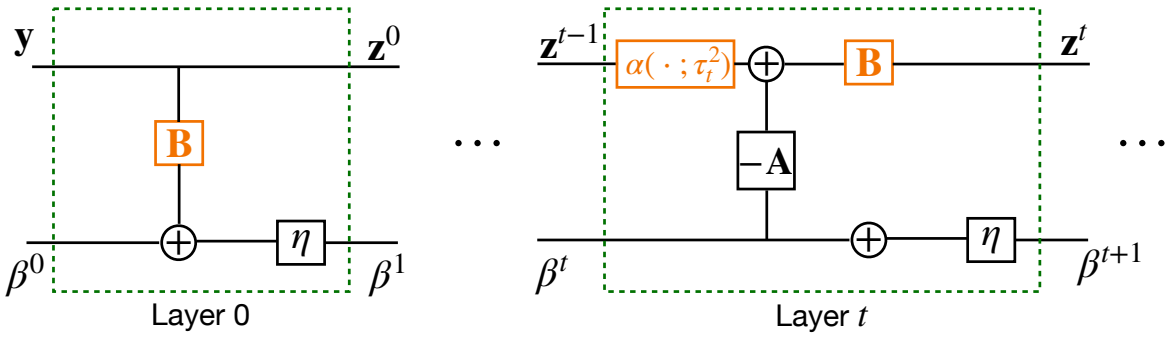

Fig. 1. Learned-AMP architecture.

III. Learned AMP (LAMP) for SPARC

In this section, we propose a deep unfolded AMP decoder, called the Learned-AMP (LAMP), obtained by interpreting the $T = T^*$ AMP iterations as layers of a feed-forward neural network and replacing selected parameters with trainable quantities optimised end-to-end via backpropagation. Fig. 1 illustrates the proposed LAMP architecture obtained by unfolding the classical AMP recursion in (2)-(3) into a layered neural network. Each layer mimics one AMP iteration. The current residual is updated using the received vector and the previous estimate, followed by a linear projection through $\mathbf{B}^{(t)}$, and then a section-wise denoising operation parametrised by τ_t^2. In this manner, the decoder preserves the model-based structure of AMP while allowing selected parameters to be learned from data, thereby improving robustness in short block-length regimes.

A. LAMP Architecture

Let $T = T^*$ denote the number of unfolded layers, where T^* is chosen according to (7) from classical AMP. The LAMP network is initialised with $\hat{\beta}^0 = \mathbf{0}$, and at each layer $t \in \{0, \dots, T-1\}$, the LAMP network computes:

$$\mathbf{z}^t = \mathbf{y} - \mathbf{A}\hat{\beta}^t + \alpha^t \mathbf{z}^{t-1}, \tag{8}$$

$$\mathbf{s}^t = \hat{\beta}^t + \mathbf{B}^{(t)}\mathbf{z}^t, \tag{9}$$

$$\hat{\beta}^{t+1} = \eta(\mathbf{s}^t; \tau_t^2), \tag{10}$$

where $\alpha^t = \frac{1}{\tau_t^2}\left(P - \frac{\|\hat{\beta}^t\|^2}{n}\right)$ is the Onsager correction coefficient, $\eta(\cdot; \tau_t^2)$ is the SPARC denoiser in (5), and $\mathbf{B}^{(t)}$ is a trainable projection matrix initialised to $\mathbf{A}^T$. The trainable parameter set is

$$\Theta = \left\{\tau_t^2,\ \mathbf{B}^{(t)}\right\}_{t=0}^{T-1}. \tag{11}$$

Trainable state evolution τ_t^2: In classical AMP, state evolution parameters τ_t^2 are interpreted as the variance of the effective Gaussian noise corrupting the true signal at the input to the denoiser. At short block lengths, this interpretation is only approximate, and learning $\tau_t^2, \forall t \in [T]$ allows the denoiser to adapt to the actual effective noise level observed at finite dimensions.

Trainable projection $\mathbf{B}^{(t)}$: Learning $\mathbf{B}^{(t)}$ relaxes the assumption that the transpose $\mathbf{A}^T$ is the best linear projection at every iteration. In practical low-latency systems, structured dictionaries such as Gold sequence or Hadamard based matrices are preferred due to their low implementation complexity. Allowing $\mathbf{B}^{(t)}$ to be learned enables the decoder to compensate for deviations from the ideal i.i.d. Gaussian assumption.

B. Training Objective

Let $\phi(\cdot)$ denote the hard section-wise decision that selects the index with the maximum value in each section. The block error rate is $P_{\text{BLER}} = \Pr\big(\phi(\hat{\beta}^T) \neq \beta\big)$. Since the hard decision is non-differentiable, we train by minimising the MSE between the soft decoder output and the true sparse vector:

$$\mathcal{L}(\Theta) = \mathbb{E}_{\beta,\mathbf{y}}\left[\|\hat{\beta}^T(\mathbf{y}, \Theta) - \beta\|_2^2\right]. \tag{12}$$

The expectation is approximated by mini-batch sampling of synthetically generated pairs $(\beta, \mathbf{y})$, where $\mathbf{y} = \mathbf{A}\beta + \mathbf{w}$, using the Adam optimiser.

C. Training Variants

We study four variants of increasing complexity:

1) τ^2 only learning: In the simplest variant, only the T-dimensional state evolution vector $\{\tau_t^2\}_{t=0}^{T-1}$ is learned. The projection is fixed at $\mathbf{B}^{(t)} = \mathbf{A}^\top$ for all t. The vector is initialised to the analytical state evolution sequence obtained from offline Monte Carlo simulation [11]. This variant has very low training complexity and isolates the gain obtained purely from correcting the mismatch in the effective noise parameters.

2) Joint $\mathbf{B}^{(t)}$ and τ_t^2 training: In this variant, both the state evolution parameters and the projection matrix are learned jointly. Two parameter sharing strategies are considered. In the tied architecture, a single projection matrix $\mathbf{B}$ is shared across all layers, resulting in $NL + T$ trainable parameters, including the state evolution parameters. In contrast, in the untied architecture, each layer learns a layer-specific $\mathbf{B}^{(t)}$, which increases model expressiveness, with the total number of parameters given by $T(NL + 1)$.

3) Incremental training: Inspired by [16], we employ layer-wise incremental training in which the network is grown progressively. At generation g, only the first g layers are optimised, using the parameters from generation $g - 1$ as initialisation. After all T layers are active, a final end-to-end fine-tuning pass is performed with a reduced learning rate and increased batch size. This strategy mitigates vanishing-gradient

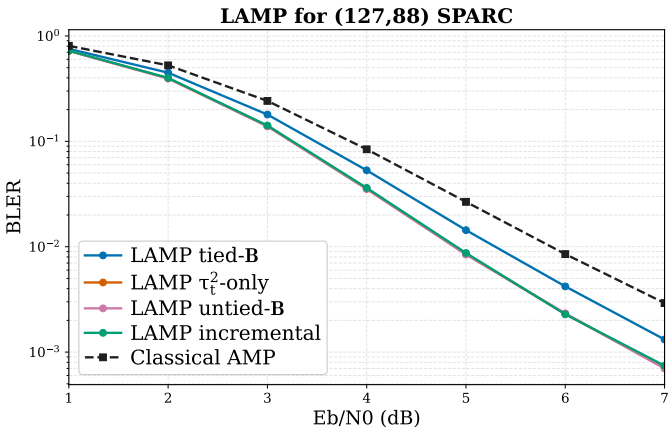

Fig. 2. Learned-AMP for (127,88) SPARC.

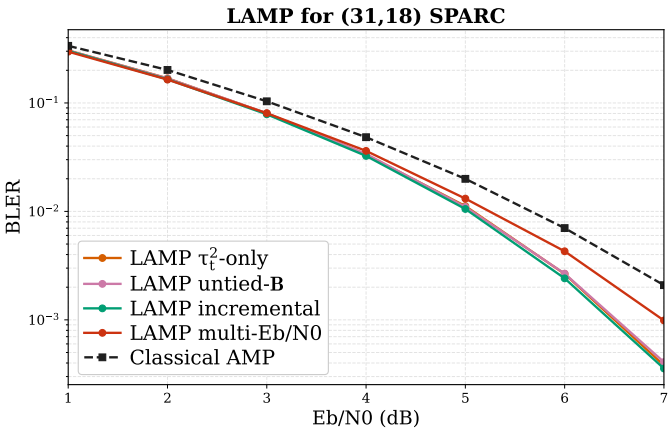

Fig. 3. Learned-AMP for (31,18) SPARC.

difficulties that arise when training deep unfolded networks jointly from random initialisation, particularly at high code rates.

4) Multi-E_b/N_0 training: In the preceding variants, a single LAMP network is trained at a fixed E_b/N_0 and evaluated across the entire SNR range. In this variant, a separate network is trained and evaluated at each target E_b/N_0 point, with the state evolution sequence and the number of layers T^* initialised accordingly. This allows the learned parameters to specialise to a particular noise level, at the cost of maintaining multiple trained models.

IV. Simulation Results

We evaluate the performance of the proposed LAMP decoder in terms of block error rate (BLER) over the AWGN channel and compare it with classical AMP in the short block-length regime. SPARC codewords are generated using Gold sequence based matrices with block lengths N = 31 and 127, and with $L = (N+1)^2$ columns. The BLER performance is evaluated as a function of E_b/N_0(in dB) with receiver noise variance $N_0 = \sigma_v^2$. The expected energy per bit is computed as $E_b = E_s/N_b$, where N_b is the number of information bits and $E_S = K$ is the expected codeword energy, with the codeword being generated from a picked column from each of the K sections of the dictionary matrix. For an (N, N_b) SPARC, the code rate is given by N_b/N bits per real dimension, or channel uses (bpcu). The empirical BLER is computed over N_{test} test codewords as

$$\hat{P}_{\text{BLER}} = \frac{1}{N_{\text{test}}} \sum_{m=1}^{N_{\text{test}}} \mathbb{1}\left\{\phi\left(\hat{\beta}^T(\mathbf{y}^{(m)})\right) \neq \beta^{(m)}\right\}. \quad (13)$$

Fig. 2 compares the LAMP variants against classical AMP for the $(127, 88)$ SPARC ($K = 8$ sections) at higher code rate. All LAMP variants outperform the classical AMP baseline, with the gains becoming more pronounced at higher E_b/N_0. Among the variants, LAMP incremental achieves the lowest BLER, followed closely by the τ^2 only and untied-$\mathbf{B}$ variants, which exhibit nearly identical performance. The tied-$\mathbf{B}$ variant improves over classical AMP but lags behind the other LAMP configurations, indicating that sharing a single projection matrix across layers limits the network's ability to adapt to per-layer statistics.

Fig. 3 presents results for the $(31, 18)$ SPARC ($K = 2$ sections) at moderate code rate for very short block lengths. All LAMP variants deliver a clear BLER improvement over classical AMP across the entire E_b/N_0 range. At high SNR, incremental training and the untied-$\mathbf{B}$ variant achieve the best performance. The multi-E_b/N_0 variant, which trains a separate network at each E_b/N_0 point, performs comparably to τ^2 only variant at low-to-moderate SNR but falls slightly behind at higher E_b/N_0. Notably, the τ^2 only variant performs comparably to the richer models, suggesting that at very short block lengths the dominant source of mismatch is the state evolution parameters rather than the projection matrix.

V. Conclusion

In this work, we developed a deep unfolded version of the classical AMP decoder, referred to as learned-AMP (LAMP), for SPARC decoding over AWGN channels. By introducing trainable parameters within the AMP framework, the proposed approach adapts to finite-dimensional effects and practical sensing matrices, overcoming the limitations of state evolution in short block-length regimes. Our simulations demonstrated that LAMP outperforms classical AMP in terms of BLER, particularly at high code rates and very short block lengths. Among the considered learning variants, incremental training was found to be especially effective for deep architectures, while learning only the state evolution parameters provides competitive performance with lower complexity. As future work, the LAMP decoder can be adapted to more complex wireless channel models, including fading and multi-user settings, where fast-varying channel conditions and multi-access interference pose additional challenges.

References

[1] T. Tao, Y. Wang, D. Li, Y. Wan, P. Baracca, and A. Wang, "6G hyper reliable and low-latency communication – requirement analysis and proof of concept," in *2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*, pp. 1–5, 2023.

[2] R. Liu, M. Hua, K. Guan, X. Wang, L. Zhang, T. Mao, D. Zhang, Q. Wu, and A. Jamalipour, "6G enabled advanced transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 9, pp. 10564–10580, 2024.

[3] A. A. Shamsabadi, A. Yadav, Y. Gadallah, and H. Yanikomeroglu, "Exploring the 6G potentials: Immersive, hyper reliable, and low-latency communication," 2025.

[4] A. R. Barron and A. Joseph, "Toward fast reliable communication at rates near capacity with Gaussian noise," in *2010 IEEE International Symposium on Information Theory*, pp. 315–319, 2010.

[5] A. R. Barron and S. Cho, "High-rate sparse superposition codes with iteratively optimal estimates," in *2012 IEEE International Symposium on Information Theory Proceedings*, pp. 120–124, 2012.

[6] M. K. Sinha and A. P. Kannu, "Generalized sparse regression codes for short block lengths," *IEEE Transactions on Communications*, 2024.

[7] A. Kumar, S. D. Kancharana, M. K. Sinha, and A. P. Kannu, "Sparse regression codes for coherent simo channels," in *2026 18th International Conference on COMmunication Systems and NETworks (COMSNETS)*, pp. 28–32, IEEE, 2026.

[8] S. D. Kancharana, M. K. Sinha, and A. Pachai Kannu, "Sparse regression codes for non-coherent simo channels," *IEEE Transactions on Communications*, vol. 74, pp. 337–349, 2026.

[9] D. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[10] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 18914–18919, 2009.

[11] C. Rush, A. Greig, and R. Venkataramanan, "Capacity-Achieving Sparse Superposition codes via Approximate Message Passing Decoding," *IEEE Transactions on Information Theory*, vol. 63, no. 3, pp. 1476–1500, 2017.

[12] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (Madison, WI, USA), p. 399–406, Omnipress, 2010.

[13] M. Borgerding, P. Schniter, and S. Rangan, "AMP-inspired deep networks for sparse linear inverse problems," *IEEE Transactions on Signal Processing*, vol. 65, no. 16, pp. 4293–4308, 2017.

[14] R. Gold, "Optimal binary sequences for spread spectrum multiplexing (corresp.)," *IEEE Transactions on Information Theory*, vol. 13, no. 4, pp. 619–621, 1967.

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

[16] D. Ito, S. Takabe, and T. Wadayama, "Trainable ISTA for sparse signal recovery," *IEEE Transactions on Signal Processing*, vol. 67, no. 12, pp. 3113–3125, 2019.