Sparse Regression Codes

Ramji Venkataramanan, University of Cambridge, ramji.v@eng.cam.ac.uk
Sekhar Tatikonda, Yale University, sekhar.tatikonda@yale.edu
Andrew Barron, Yale University, andrew.barron@yale.edu

1. Introduction

Developing computationally-efficient codes that approach the Shannon-theoretic limits for communication and compression has long been one of the major goals of information and coding theory. There have been significant advances towards this goal in the last couple of decades, with the emergence of turbo and sparse-graph codes in the '90s [1, 2], and more recently polar codes and spatially-coupled LDPC codes [3–5]. These codes are all primarily for discrete-alphabet sources and channels.

There are many channels and sources of practical interest where the alphabet is inherently continuous, e.g., additive white Gaussian noise (AWGN) channels, and Gaussian sources. A promising class of codes for Gaussian models is the recently proposed *Sparse Superposition Codes or Sparse Regression Codes* (SPARCs). This article provides a broad overview of SPARCs, covering theory, algorithms, and some practical implementation aspects. At the end, we discuss some open problems and future directions for research.

This survey is based on the tutorial on sparse regression codes presented at ISIT '16[1]. The discussion will be relatively informal. We paraphrase some of the technical results with the aim of providing intuition, and point the reader to references for an in-depth discussion.

To motivate the construction of SPARCs, let us begin with the standard AWGN channel with signal-to-noise ratio denoted by snr. The goal is to construct codes with computationally efficient encoding and decoding that *provably* achieve the channel capacity $C = (1/2)\log(1 + \text{snr})$ bits/transmission. In particular, we are interested in codes whose encoding and decoding complexity grows no faster than a low-order polynomial in the block length n.

Though it is well known that rates approaching C can be achieved with Gaussian codebooks, this has been largely avoided in practice due to the high decoding complexity of unstructured Gaussian codes. Instead, the popular approach has been to separate the design of the coding scheme into two steps: *coding* and *modulation*. State-of-the-art coding schemes for the AWGN channel such as coded modulation [6–8] use this two-step design, and combine binary error-correcting codes such as LDPC and turbo codes with standard modulation schemes such as Quadrature Amplitude Modulation (QAM). Though such schemes have good empirical performance, they have not been proven to be capacity-achieving for the AWGN channel.

With sparse regression codes, we step back from the coding/modulation divide and instead use a structured codebook to construct low-complexity, capacity-achieving schemes tailored to the AWGN channel. In a SPARC, codewords are sparse linear combinations of columns of a design matrix (see Fig. 1). The codewords are indexed by the locations of non-zeros in each section.

[1]The slides from the tutorial are available at https://goo.gl/8H8wrk

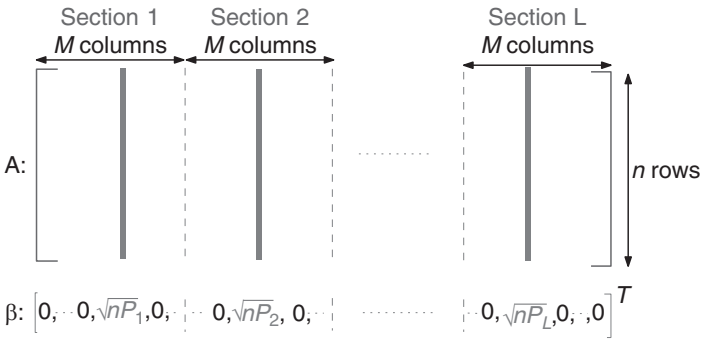


Figure 1: A Gaussian sparse regression codebook of block length *n*: *A* is a design matrix with independent Gaussian entries, and *β* is a sparse vector with one non-zero in each of *L* sections, where $L \sim (n/\log n)$. Codewords are of the form *Aβ*, i.e., linear combinations of the columns corresponding to the non-zeros in *β*. The message is indexed by the *locations* of the non-zeros, and the values $P_1, \ldots, P_L$ are fixed a priori.

We explain in Sec. 2 how the parameters of the design matrix determine the rate of the code, average power etc. In Sec. 3, we will see how the structure of the design matrix enables fast iterative decoding algorithms whose probability of decoding error decays rapidly with block length for rates $R < C$. Further, these codes also achieve the Shannon limit for lossy compression (Sec. 4), and can be easily combined to implement superposition and binning (Sec. 5). Thus sparse regression codes offer a way to construct low-complexity, rate-optimal codes for a variety of canonical models in network information theory.

We should mention that lattice codes are another class of structured codes for Gaussian channel and source models [9]. Several elegant coding schemes based on lattices have been proposed in the literature, e.g. [10–12], but we will not discuss these further in this article.

2. The Sparse Regression Codebook

As shown in Fig. 1, a SPARC is defined in terms of a 'dictionary' or design matrix A of dimension $n \times ML$, whose entries are *i.i.d.* $\mathcal{N}(0, \frac{1}{n})$. Here n is the block length, and M, L are integers whose values are specified below in terms of n and the rate R. We think of the matrix A as being composed of L sections with M columns each. Each codeword is a linear combination of L columns, with one column coming from each section. Formally, a codeword can be expressed as $A\beta$, where β is an $ML \times 1$ vector $(\beta_1, \ldots, \beta_{ML})$ with the following property: there is exactly one non-zero β_j for $1 \le j \le M$, one non-zero β_j for $M + 1 \le j \le 2M$, and so forth. The non-zero value of β in section $\ell \in [L]$ is set to $\sqrt{nP_\ell}$, where the positive constants P_ℓ satisfy $\sum_{\ell=1}^{L} P_\ell = P$. (We use the notation $[L]$ to denote the set $\{1, \ldots, L\}$.)

P is the average power per input symbol in the case of channel coding; in lossy compression it will be the variance of each codeword symbol.

Since each of the L sections contains M columns, the total number of codewords is M^L. To obtain a rate R code, we need

$$M^L = 2^{nR} \quad \text{or} \quad L \log M = nR. \tag{1}$$

(Throughout, we use log for logarithm with base 2, and ln for base e.) There are several choices for the pair (M, L) which satisfy (1). For example, $L = 1$ and $M = 2^{nR}$ recovers the Shannon-style random codebook in which the number of columns in A is 2^{nR}. For our constructions, we will choose M equal to L^a, for some constant $a > 0$. In this case, (1) becomes

$$aL \log L = nR. \tag{2}$$

Thus $L = \Theta(n/\log n)$, and the size of the design matrix A (given by $n \times ML = n \times L^{a+1}$) grows polynomially in n. In our numerical simulations, typical values for L are 512 or 1024.

We note that the SPARC is a non-linear code with pairwise dependent codewords. Two codewords $A\beta$ and $A\beta'$ are dependent whenever the underlying message vectors β, β' share one or more common non-zero entries.

Power Allocation: The coefficients $\{P_\ell\}_{\ell=1}^L$, plays an important role in determining the performance of the code, both for channel coding and for lossy compression. We will consider allocations where $P_l = \Theta(1/L)$. Two examples are:

- Flat power allocation across sections: $P_\ell = \dfrac{P}{L}, \ell \in [L]$.
- Exponentially decaying power allocation: Fix parameter $\kappa > 0$. Then $P_\ell \propto 2^{-\kappa \ell/L}, \ell \in [L]$.

In Section 3, we discuss computationally efficient decoders which asymptotically achieve capacity with the exponentially decaying allocation (with $\kappa = 2C$). To improve decoding performance at practical block lengths, we explore different power allocation strategies in Section 3.3, and demonstrate that judicious power allocation can lead to dramatic improvements in decoding performance at finite block lengths. We also describe how decoding complexity can be reduced by replacing the Gaussian design matrix with a Hadamard-based design.

3. AWGN Channel Coding with SPARCs

The channel is described by the model

$$y_i = x_i + w_i, \quad i = 1, \ldots, n. \tag{3}$$

The noise variables w_i are *i.i.d.* $\sim \mathcal{N}(0, \sigma^2)$. There is an average power constraint P on the input: the codeword $x := (x_1, \ldots, x_n)$ should satisfy $\frac{1}{n}\sum_{i=1} x_i^2 \le P$. The signal-to-noise ratio P/σ^2 is denoted by snr.

Encoding: The encoder splits its stream of input bits into segments of log M bits each. A length ML message vector β_0 is indexed by L such segments – the decimal equivalent of segment ℓ determines the position of the non-zero coefficient in section ℓ of β_0. The input codeword is then computed as $x = A\beta_0$. Note that computing x simply involves adding L columns of A, weighted by the appropriate coefficients.

Optimal Decoding: Assuming that the codewords are equally likely, the optimal decoder produces

$$\hat{\beta}_{opt} = \arg\min_{\beta} \|y - A\beta\|^2,$$

where $y := (y_1, \ldots, y_n)$, and the minimum is over all the message vectors in the codebook.

Probability of Error: The performance of a SPARC decoder is measured by the *section error rate*, which is the fraction of sections decoded wrongly. The section error rate is denoted by $\mathcal{E}_{\text{sec}} := \frac{1}{L}\sum_{\ell=1}^{L} 1\{\hat{\beta}_\ell \neq \beta_{0\ell}\}$. For a given decoder, we will aim to bound the probability of the event $\{\mathcal{E}_{\text{sec}} > \epsilon\}$ for $\epsilon > 0$. Assuming that the mapping determining the non-zero location for each segment of log M input bits is generated uniformly at random, a section error will, on average, lead to half the bits corresponding to the section being decoded wrongly. Therefore, when a large number of segments are transmitted, the *bit error rate* of a SPARC decoder will be close to half its section error rate.

If we want the decoding error probability of the *message* β_0 to be small, we can use a concatenated code with the SPARC as the inner code and an outer Reed-Solomon (RS) code. (An RS code of rate $(1 - 2\epsilon)$ can correct upto a fraction ϵ of section errors in the SPARC; see [13] for details.)

We will not consider the outer RS code in the remainder of this article, and focus mostly on the section error rate (or bit error rate) of the SPARC.

Performance with Optimal Decoding: For rates $R < C$, the least-squares decoder was shown in [13] to have error probability decaying exponentially in the block length.

Theorem 1. [13] *Consider a SPARC with rate $R < C$, block length n, and equal power allocation, i.e, $P_\ell = (P/L), \ell \in [L]$. For any $\epsilon > 0$, the section error rate of the least-squares decoder satisfies*

$$\mathbb{P}(\mathcal{E}_{\text{sec}} > \epsilon) \le K e^{-\kappa n \min\{\epsilon, (C-R)^2\}}.$$

where κ, K are universal positive constants.

This result was extended to SPARCs with i.i.d. binary (± 1) design matrices in [14, 15]. The exponent κ in Theorem 1 is smaller than the Shannon-Gallager random coding exponent, but the result shows that SPARCs are essentially as good as Shannon-style random codes for the AWGN channel with maximum-likelihood decoding.

3.1. Feasible SPARC Decoders

In contrast to the least-squares decoder, the feasible decoders we discuss all use a decaying power allocation across sections. Thinking of the L sections of a SPARC as analogous to L users sharing a Gaussian multiple- access channel (MAC), leads to an exponentially decaying power allocation of the form $P_\ell \propto 2^{-2C\ell/L}, \ell \in [L]$. Indeed, consider the equal-rate point on the capacity region of a L-user Gaussian MAC where each user gets rate C/L. It is well-known [16, 17] that this rate point can be achieved with the above power allocation via successive cancellation decoding, where user 1 is first decoded, then user 2 is decoded after subtracting the codeword of user 1, and so on.

However, successive cancellation performs poorly for SPARC decoding. This is because L, the number of sections ("users") in the codebook, grows as $n/\log n$, while M, the number of codewords per user, only grows polynomially in n. An error in decoding one section affects the decoding of future sections, leading to a large number of section errors after L steps.

The first feasible SPARC decoder, proposed in [18], controls the accumulation of section errors using *adaptive* successive decoding. The idea is to not pre-specify the order in which sections are decoded, but to look across all the undecoded sections in each step, and adaptively decode columns which have a large inner product with the residual. The main ingredients of the algorithm are as follows. In the first step, the decoder computes the inner product of each column of the design matrix with the normalized channel output sequence $y/\|y\|$, and picks those columns for which this test statistic exceeds a pre-specified threshold; this gives the first estimate $\hat{\beta}_1$. In the second step, the test statistic is generated based on the *residual* $r^1 = y - A\hat{\beta}^1$: the decoder picks the columns (from the as yet undecoded sections) whose inner product with $r^1/\|r^1\|$ crosses the threshold; this gives $\hat{\beta}^2$. The algorithm continues in this fashion, decoding columns using the residual-based statistics in each step. The algorithm is run for a pre-specified number of steps, arranged to be of the order of $\log M$; it terminates earlier if at least one column has been selected from each section, or the test-statistics in any step are all below the threshold.

The performance of this decoder was analyzed in [18]. With power allocation $P_\ell \propto 2^{-2C\ell/L}$, it was shown that the probability of message decoding error decays as $\exp(-kL(C_M R)^2)$, where $C_M = C(1 - \frac{c}{\log M})$ for a constant $c > 0$, and R is the total rate (SPARC combined with an outer Reed-Solomon code).

Therefore the adaptive successive threshold decoder is capacity-achieving, and the gap to capacity is of order $1/\log M$. However, in practice, the section error rates at practically feasible block lengths are observed to be rather high for rates near capacity. The following two decoders improve the decoding performance by avoiding hard decisions about which columns to decode in each step.

3.2. Iterative Soft-decision Decoding

The key idea in the next two decoders is to iteratively update the posterior probabilities of each entry of β being the true non-zero in its section. The goal in both decoders is to iteratively generate test statistics that (in step t) have the form $\text{stat}_t \approx \beta + \tau_t Z_t$, where Z_t is standard normal and independent of β. In words, stat_t is essentially the message vector observed in independent additive Gaussian noise with known variance τ_t^2. Assuming this is true, the Bayes-optimal estimate for β in the next step is

$$\beta^{t+1}(\text{stat}_t) = \mathbb{E}\,[\beta | \beta + \tau_t Z_t = \text{stat}_t] = \eta_t(\text{stat}_t),$$

where the conditional expectation $\eta_t(\cdot)$ can be computed using the known prior on β (locations of non-zeros uniformly distributed within each section). For indices j in section ℓ of beta, we have

$$\eta_{t,j}(s) := \sqrt{nP_\ell}\,\frac{\exp(\sqrt{nP_\ell}\,s_j/\tau_t^2)}{\sum_{k\in \text{sec}_\ell}\exp(\sqrt{nP_\ell}\,s_k/\tau_t^2)}, \quad j \in \text{section } \ell, \ell \in [L]. \tag{4}$$

Note that $\eta_{t,j}(s)/\sqrt{nP_\ell}$ is the posterior probability (given stat_t) that term j is the non-zero coefficient in section ℓ of β.

In addition to stat_t having the desired distributional representation, we also want τ_t^2, the variance of the noise in the test statistic, to be computable iteratively from τ_{t-1}^2 as follows. Starting with $\tau_0^2 = \sigma^2 + P$, we define

$$\begin{aligned}\tau_t^2 &= \sigma^2 + \frac{1}{n}\mathbb{E}\|\beta - \mathbb{E}[\beta \mid \beta + \tau_{t-1}Z_{t-1}]\|^2 \\ &= \sigma^2 + \frac{1}{n}\mathbb{E}\|\beta - \eta_t(\beta + \tau_{t-1}Z_{t-1})]\|^2,\end{aligned} \tag{5}$$

where the expectation on the right is over β and the independent standard normal vector Z_t. In other words, we want the noise in the test statistic to have two independent Gaussian components: one component with variance σ^2 arising from the channel noise, and the other component arising from the error in the current estimate β^t. The recursion to generate τ_t^2 from τ_{t-1}^2 can be written as

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

where $x_t := x(\tau_{t-1})$ is an expectation of a function of ML standard normal random variables. The exact formula for $x(\tau_{t-1})$ can be found in [19, Sec. 3]. Compact asymptotic formulas for x_t, τ_t^2 are given in Lemma 1 below.

Therefore, under the assumed distribution for stat_t, we have $\frac{1}{n}\mathbb{E}\,\|\beta - \beta^t\|^2 = P(1 - x_t)$; it can also be shown that $\frac{1}{n}\mathbb{E}[\beta^T\beta^t] = \frac{1}{n}\mathbb{E}\|\beta^t\|^2 = Px_t$ [19, Prop. 3.1]. Thus the scalar x_t can be interpreted as the expected (power-weighted) success rate, and $P(1 - x_t)$ as the expected interference contribution to the noise variance τ_t^2 due to the undecoded sections. With this interpretation, for succesful decoding we want x_t to be very close to 1 when the algorithm terminates. Indeed, it can be verified that for all rates less than C and $P_\ell \propto 2^{-2C\ell/L}$, the iteration (6) has a fixed point with τ_t^2 close to σ^2, i.e., x_t is close to one. A more precise version of this statement in the large system limit is given in Lemma 1 below.

Finally, the key question is: how do we iteratively generate statistics stat_t that *in each step* are well-approximated as $\beta + \tau_t Z_t$, with τ_t^2 having the representation described above? The two decoders described below achieve this via seemingly very different approaches.

Adaptive Successive Soft-Decision Decoder [20–22]

The statistics for this decoder are defined using the fits $\text{Fit}_0 := Y, \text{Fit}_1 := A\beta^1, \ldots, \text{Fit}_t := A\beta^t$. With $G_0 := Y$, recursively define G_t to be the part of Fit_t that is orthogonal to $G_0, G_1, \ldots, G_{t-1}$. The ingredients of stat_t are the vectors $\mathcal{Z}_0, \ldots, \mathcal{Z}_t$, defined as

$$\mathcal{Z}_k = \sqrt{n}\,\frac{A^T G_k}{\|G_k\|}, \quad k \geq 0.$$

The test statistic is then defined as $\text{stat}_t = \tau_t \sum_{k=0}^t \lambda_k \mathcal{Z}_k + \beta^t$. The weights λ_k have to be carefully chosen in order for stat_t to be close enough in distribution to the desired form $\beta + \tau_t \mathcal{Z}_t$. The estimate β^{t+1} is generated as $\eta_t(\text{stat}_t)$, where $\eta_t(\cdot)$ is given by (4).

Two different ways to choose the weights $\lambda_k, 0 \leq k \leq t$, are proposed in [21, 22]. Each of these choices is based on a technical lemma [20, Lemma 1] characterizing the distribution of $\mathcal{Z}_k$, $\forall k$. The first choice of weights is deterministic, and given by

$$(\lambda_0, \lambda_1, \ldots, \lambda_t) : \tau_t\Big(\frac{1}{\tau_0}, -\sqrt{\frac{1}{\tau_1^2} - \frac{1}{\tau_0^2}}, \ldots, -\sqrt{\frac{1}{\tau_t^2} - \frac{1}{\tau_{t-1}^2}}\Big), \tag{7}$$

where $\tau_0,\ldots,\tau_t$ are given by (5). The analysis in [21] shows that this choice of weights makes stat_t close to the desired representation $\beta+\tau_t Z_t$, leading to the following concentration result.

Theorem 2. *[21, Lemma 7] Consider a SPARC with rate $R < C$, parameters (n, L, M) chosen according to* (1)*, and power allocation $P_\ell \propto 2^{-2C\ell/L}$. For $t \geq 1$, let*

$$\mathcal{A}_t := \Big\{\Big|\frac{1}{nP}\beta^T\beta^t - x_t\Big| > \varepsilon\Big\} \cup \Big\{\Big|\frac{1}{nP}\|\beta^t\|^2 - x_t)\Big| > \epsilon\Big\}.$$

Then, we have

$$\mathbb{P}\{\cup_{k=1}^t \mathcal{A}_k\} \lesssim \sum_{k=1}^t a_k \exp\Big(-k\frac{n}{(\log M)^{2k+1}}\epsilon^2\Big),$$

where $k, a_1, \ldots, a_t$ are universal constants depending on R,C.

The probability bound on the event $\mathcal{A}_t$ in Theorem 2 can be shown to imply a probability bound for the section-error rate exceeding $c\varepsilon$, where $c > 0$ is a constant. Thus the probability of decoding failure decays exponentially in $n/(\log n)^{2T^*+1}$, where T^* is the number of steps for which the algorithm is run. This is in contrast to the optimal decoder in Theorem 1 whose probability of decoding failure decays exponentially in n.

As an alternative to the deterministic weights in (7), weights $\{\lambda_k\}$ depending on the channel output y were also proposed in [21]. This choice is based on the Cholesky decomposition of a matrix generated from the estimates $\{\beta^1,\ldots,\beta^t\}$. A performance guarantee similar to Theorem 2 can be obtained for this set of weights as well; see [21, 22] for details.

Approximate Message Passing Decoder [19, 23]

Approximate message passing (AMP) refers to a class of algorithms [24–30] that are Gaussian or quadratic approximations of loopy belief propagation algorithms (e.g., min-sum, sum-product) on dense factor graphs. In its basic form [24, 27], AMP gives a fast iterative algorithm to solve the LASSO, i.e., to compute

$$\hat{\beta}_{LASSO} = \arg\min_{\hat{\beta}} \|y - A\hat{\beta}\|_2^2 + \lambda\|\hat{\beta}\|_1,$$

for any $\lambda > 0$. Recall that the decoding problem we wish to solve is

$\hat{\beta}_{SPARC} = \arg\min_{\hat{\beta}} \|y - A\hat{\beta}\|_2^2$ over $\hat{\beta}$ that are valid SPARC codewords.

One cannot directly use the LASSO-AMP of [24, 27] for SPARC decoding as it does not use the prior knowledge about β, i.e., the knowledge that β has exactly one non-zero value in each section, with the values of the non-zeros also being known.

An AMP decoder for SPARCs can be derived by writing down min-sum like updates for the SPARC decoding problem, and then approximating them using the recipe in [26]. This leads to a decoder with the following update rules [19]. Define $r^0 = y$, and for $t \geq 1$ compute:

$$r^t = y - A\beta^t + \frac{r^{t-1}}{\tau_{t-1}^2}\Big(P - \frac{\|\beta^t\|^2}{n}\Big),$$
$$\text{stat}_t = A^T r^t + \beta^t,$$
$$\beta^{t+1} = \eta_t(\text{stat}_t).$$

The coefficients τ_t^2 are recursively defined by (6), starting with $\tau_0^2 = \sigma^2 + P$. Following the terminology in [24, 26], we refer to this recursion as state evolution (SE). Recall that the SE equations are derived under the assumption that stat_t is distributed as $\beta + \tau_t Z_t$. The presence of the "Onsager" term $r^{t-1}/\tau_{t-1}^2\big(P - \frac{\|\beta^t\|^2}{n}\big)$ in the definition of the modified residual r^t is crucial to ensure that the distributional assumption is valid, at least asymptotically. Intuition about role of the Onsager term in the standard AMP algorithm can be found in [26, Section I-C].

We can derive a compact asymptotic formula for the SE recursion by taking the limit as $L, M, n \to \infty$ while satisfying (1). (This limit is denoted below by 'lim'.)

Lemma 1. *[19] For $t \geq 1$, the asymptotic value of τ_t^2, denoted by τ_t^2, is given by*

$$\tau_t^2 = \sigma^2 + P(1 - \bar{x}(\bar{\tau}_{t-1})),$$

where the function $\bar{x}(\cdot)$ is defined as follows. With $c_\ell := LP_\ell$*, we have*

$$\bar{x}(\tau) := \lim x(\tau) = \lim \sum_{\ell=1}^L \frac{P_\ell}{P} 1\{\lim c_\ell > 2(\ln 2)R\tau^2\}.$$

Recalling that x_{t+1} is the expected power-weighted fraction of correctly decoded sections after step $(t+1)$, for any power allocation $\{P_l\}$, Lemma 1 may be interpreted as follows: in the large system limit, for a section ℓ to be correctly decoded in step $(t+1)$, the limit of LP_l must exceed a threshold equal to $2(\ln 2)R\bar{\tau}_t^2$. All sections which satisfy this condition will be decodable in step $(t+1)$ (i.e., will have most of the posterior probability mass on the correct term). Conversely, any section whose power falls below the threshold will not be decodable in this step.

Lemma 1 can be used to quickly check whether a given power allocation is good by checking whether $\bar{x}(\tau_t)$ monotonically increases with t from 0 to 1. When applied to the exponentially decaying power allocation $P_\ell \propto 2^{-2C\ell/L}$, Lemma 1 gives

$$\bar{\tau}_t^2 = \sigma^2(1+\text{snr})^{1-\xi_{t-1}},\ \bar{x}_t = \frac{(1+\text{snr}) - (1+\text{snr})^{1-\xi_{t-1}}}{snr} \text{ for } t > 0, \quad (8)$$

where $\xi_{t-1} := 0$ and

$$\xi_t = \min\Big\{\Big(\Big(\frac{1}{2C}\Big)\log\Big(\frac{C}{R}\Big) + \xi_{t-1}\Big), 1\Big\}.$$

The constants $\{\xi_t\}_{t\geq 0}$ have a nice interpretation in the large system limit: for $R < C$, at the end of step $t+1$, the first ξ_t fraction of sections in β^{t+1} will be correctly decodable with high probability. An additional $\frac{1}{2C}\log(\frac{C}{R})$ fraction of sections become correctly decodable in each step until step $T^* = \lceil 2C/\log(C/R)\rceil$, when all the sections are correctly decodable with high probability.

The following theorem shows that the AMP decoder achieves capacity by showing that the above interpretation based on the SE equations is true in the large system limit.

Theorem 3. *For any rate $R < C$, consider a sequence of rate R SPARCs $\{S_n\}$ indexed by block length n and power allocation $P_\ell \propto 2^{-2C\ell/L}$. Then the section error rate of the AMP decoder (run for T^* steps, with the constants $\bar{\tau}_t^2$ given by (8)) converges to zero almost surely, i.e., for any $\epsilon > 0$,*

$$\lim_{n_0\to\infty} \mathbb{P}(\mathcal{E}_{\text{sec}}(S_n) < \varepsilon, \forall n \geq n_0) = 1.$$

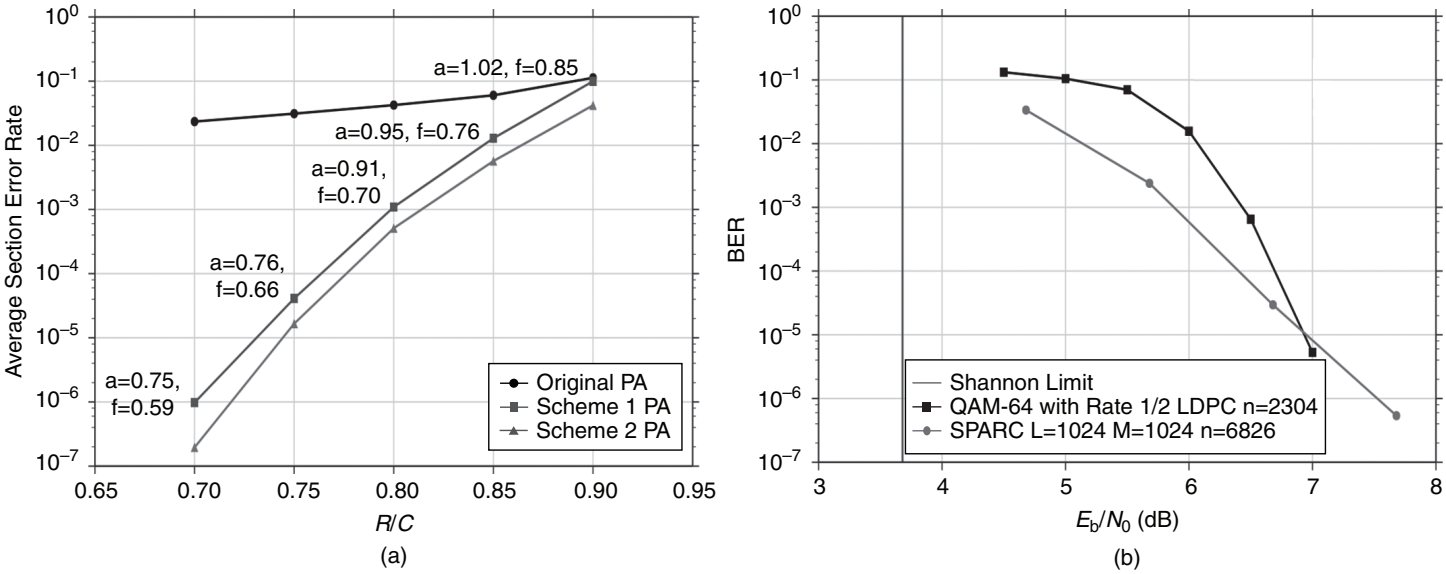


Figure 2: Performance with AMP decoding: a) Section error rate vs. *R* with snr = 15, C = 2 bits, SPARC parameters M = 512, L = 1024; b) Bit error rate of SPARC vs. E_b/N_0 at *R* = 1.5, compared with coded modulation at information rate = 1.5 bit/dimension.

In recent work, we have used the the finite-sample AMP analysis techniques of [31] to refine the asymptotic result of Theorem 3 and obtain a large-deviations bound similar to Theorem 2 for the probability of the section error rate exceeding ϵ.

Computational Complexity

With a Gaussian design matrix, the running time and memory requirement of both the adaptive successive soft-decision decoder and the AMP decoder are of the same order: $O(nML)$. However, in practice the AMP decoder is faster as no orthonormalization or Cholesky decomposition is required to compute its test statistic in each step. Further, as described in [19], choosing the design matrix by uniformly sampling n rows of the $ML \times ML$ Hadamard matrix reduces the AMP running time to $O(ML \log M)$. The Hadamard-based design matrix does not need to be stored, hence there is also a large saving in required memory. Finally, the partitioned structure of the SPARC could be exploited to design parallelized or pipelined implementations of the above decoders.

3.3. Empirical performance at practical blocklengths

Though all three decoders theoretically have section error-rate decaying to zero with increasing block length for any fixed $R < C$, the soft-decision decoders have much better empirical performance [19, 22]. In the following, we illustrate the performance of the AMP decoder for block lengths of the order of a few thousands. All the simulation results are obtained using Hadamard-based designs.

Fig. 2a illustrates the performance for a SPARC with $M = 512$, $L = 1024$, snr = 15 at various values of rate R. The block length n is determined by R according to (1). For example, we have $n = 7680$ for $R = 0.6C$, and $n = 5120$ for $R = 0.9C$. The top curve shows the average section error rate of the AMP (over 1000 runs) with the power $P_\ell \propto 2^{-2C\ell/L}$ allocation . The bottom two curves are obtained with two alternative power allocation (PA) schemes, discussed below. Though $P_\ell \propto 2^{-2C\ell/L}$ is the optimal PA for rates very close to C, it is clear that as we back off from capacity, a carefully chosen PA can reduce the error rate by several orders of magnitude.

PA Scheme 1: The PA is determined by two parameters a, f. For $a > 0$ and $f \in [0, 1]$, let

$$P_\ell = \begin{cases} \kappa \cdot 2^{-a2C\ell/L}, & 1 \le \ell \le fL \\ \kappa \cdot 2^{-a2Cf}, & fL + 1 \le \ell \le L, \end{cases}$$

where κ is a normalizing constant chosen so that $\sum_\ell P_\ell = P$. The parameter a controls the decay of the exponential. Increasing a increases the power allocated to the initial sections which makes them more likely to decode correctly, which in turn helps by decreasing the effective noise variance in subsequent AMP iterations. However, if a is too large, the final sections may have too little power to decode correctly – this is why the standard PA with $a = 1$ performs poorly for rates that are not close to capacity. Thus we want the parameter a to be large enough to ensure that the AMP gets started on the right track, but not much larger.

The parameter f controls the *flattening* of the PA. The exponentially decaying PA may leave too little power for the final sections. To address this issue, the idea is to have an exponential PA only for a fraction f of the sections, and allocate the remaining power equally among the rest of the sections. The middle curve in Fig. 2a shows the performance of the AMP with numerically optimized (a, f) values for each rate. As expected, the optimal values of a, f decrease as we back off from capacity.

PA Scheme 2: Optimizing the parameters (a, f) is computationally intensive and has to be done separately for each rate and snr value of interest. To address this, we have recently developed a simple PA algorithm based on Lemma 1. If the AMP decoder is run for T^* steps, the goal (in the large system limit) is to have the first $1/T^*$ fraction of sections be decodable in the first step; the second $1/T^*$ fraction be decodable in the second step, and so on. Starting with $\tilde{\tau}_0^2 = \sigma^2 + P$, Lemma 1 lets us calculate the minimum power required for a

section to be decodable in the first step. We allocate approximately this power to each of the first L/T^* sections. Then compute $\bar{\tau}_1^2$, and from Lemma 1, the minimum power required for a section to now be decodable; allocate approximately this amount of power to the next L/T^* sections. Repeat this process sequentially for each set of L/T^* sections, with the following caveat: at any stage if the minimum power prescribed by Lemma 1 is less than what could be obtained by allocating the available power equally among the remaining sections, then choose the latter and complete the power allocation.

The bottom curve at the bottom in Fig. 2a shows the decoding performance of the AMP with this PA scheme. Clearly, the performance is at least as good as the first scheme, without having to optimize over the parameters (a, f). We used the second PA scheme to compare the performance of the SPARC with that of coded modulation schemes which combine QAM constellations with a powerful binary LDPC code. Fig 2b illustrates the bit-error performance of a SPARC vs. coded modulation at rate 1.5 bit/dim. at various values of E_b/N_0. (For the SPARC, E_b/N_0 can be calculated as $E_b/N_0 = \text{snr}/(2R)$.) The coded modulation scheme consists of a 64-QAM constellation with a rate 1/2 LDPC code. The LDPC code is specified in the WiMAX standard 802.16e and was implemented using the Coded Modulation Library [32]. We see that the SPARC with AMP decoding achieves a BER of 10^{-4} at snr around 2.5 dB from the Shannon limit.

Another way to improve the empirical performance of SPARCs is via *spatially coupled* design matrices, as demonstrated in [23, 33, 34]. Here the idea is to have a band-diagonal structure for the design matrix, with overlapping Hadamard blocks near the diagonal and zeros elsewhere. The idea is to have some extra channel outputs to reliably determine the first few sections of β; this kick-starts a decoding progression due to the overlapping structure of the design matrix.

4. Lossy Compression with SPARCs

In this section we show that SPARCs are useful for lossy compression of continuous alphabet sources with squared-error distortion criterion. For any ergodic source with variance v^2, the goal is to develop computationally efficient codes that achieve a target distortion D with a rate R as close as possible to the Gaussian rate-distortion bound $R^*(D) = v^2 e^{-2R}$ nats. (For this section alone, it will be convenient to use natural logarithms and measure rate in nats.)

The sparse regression codebook is exactly as described in Section 2, with codewords of the form $A\beta$ where β has one non-zero entry in each section. The only difference is that the values of the non zeros, $\{\sqrt{nP_\ell}\}$, do not have to satisfy a power constraint; they can be chosen in any way to help the compression encoder.

Optimal Encoding: Given a source sequence $s := (s_1, \ldots, s_n)$, the optimal (least-squares) encoder determines $\hat{\beta}_{opt} := \text{argmin} \|s - A\hat{\beta}\|^2$, where the minimization is over all $\hat{\beta}$ with the SPARC structure. The positions of the non-zeros in $\hat{\beta}_{opt}$ are conveyed using R nats/sample to the decoder, which produces the reconstruction $\hat{s} = A\hat{\beta}_{opt}$.

The following result characterizes the probability of excess distortion with optimal encoding.

Theorem 4. [35, 36] *Let $s := (s_1, \ldots, s_n)$ be drawn from an ergodic source with mean zero and variance σ^2. Let $D \in (0, \sigma^2)$, $R > \frac{1}{2}\ln\frac{\sigma^2}{D}$, and $\gamma^2 \in (\sigma^2, De^{2R})$. Let $P_\ell = (P/L) \forall \ell$ and let the SPARC parameters determined by* (1) *satisfy $M = L^a$ for $a > a^*$, where the constant a^* depends only on R and γ^2/D. Then for all sufficiently large n,*

$$\mathbb{P}\Big(\frac{1}{n}\|s - A\hat{\beta}_{opt}\|^2 > D\Big) \le \mathbb{P}\Big(\frac{\|s\|^2}{n} > \gamma^2\Big) + \exp(-\kappa n^{1+c}), \qquad (9)$$

where κ, c are strictly positive constants.

A few remarks about the two terms on the right-hand side of (9). The first term is the probability of the source sequence being atypical, i.e., the probability that its second moment is significantly greater than σ^2. The second term is the probability that the SPARC does not contain a codeword within distortion D of a typical source sequence. Note that the second term decays super-exponentially in n. Thus, if the probability of observing an atypical source sequence decays exponentially in n (e.g., as for an i.i.d. Gaussian source), it is the first term that dominates the excess distortion probability. The phenomenon of source atypicality being the dominant error event can also be observed in the analysis of the optimal excess-distortion exponent for memoryless discrete and Gaussian sources [37, 38].

An immediate corollary of Theorem 4 is that SPARCs with least-squares encoding achieve the *optimal* excess-distortion exponent for memoryless Gaussian sources derived in [38]. This result should be contrasted with the AWGN channel coding result (Theorem 1), where SPARCs with optimal decoding have probability of error decreasing exponentially in n, but the error exponent is smaller than the Shannon-Gallager random coding exponent [13].

The proof of Theorem 4 uses some techniques recently developed to characterize thresholds for random graph coloring and random constraint satisfaction problems [39, 40]. Denote the number of codewords that are within distortion D of the source sequence by Z. We need to upper bound the probability of the event $Z = 0$. Due to the dependence structure of the codewords, the techniques we use boils the analysis down to showing that $\mathbb{E}Z^2$ is of the same order as $(\mathbb{E}Z)^2$. Curiously, for distortions $D \gtrsim 0.2v^2$, the required condition is true only for rates greater than a threshold which is *strictly larger* than $R^*(D)$ [35]. To prove that Theorem 4 holds for all distortions, we use a refined second-moment analysis in [36] that excludes design matrix realizations that give rise to an atypically large number of solutions. This approach is inspired by a similar idea used to obtain improved thresholds for the problem of coloring random hypergraphs [39], and could potentially be useful in other probabilistic settings where one needs to count the number of (dependent) solutions.

Feasible Encoding: A simple SPARC compression encoder based on successive cancellation was proposed in [41]. The encoder starts with $\beta^0 = 0$, and sequentially encodes the position of the non-zero in each section of β. The non-zero location in section ℓ corresponds to the column in the ℓth section of A that maximizes the inner product with the residual $S - A\beta^{\ell-1}$. The update β^l is then generated by setting the non-zero value in section ℓ to $\sqrt{2(\ln M)v^2(1-\frac{2R}{L})^{\ell-1}}$. After the non-zero location in the final section is chosen, the codeword is computed as $A\beta^L$.

Theorem 5. [4I] *For an ergodic source S with mean 0 and variance v^2, the encoding algorithm produces a* codeword $A\hat{\beta}$ *that satisfies the following for sufficiently large M, L:*

$$\mathbb{P}\Big(\frac{1}{n}\|S - A\hat{\beta}\|^2 > v^2 e^{-2R} + \Delta\Big) < e^{-\kappa n\left(\Delta - \frac{c\ln\ln M}{\ln M}\right)}$$

where κ, c are universal *positive* constants.

We can view the encoder as successively refining the source over over $L \sim (n/\log n)$ stages, with each stage being a rate-distortion code of rate R/L. The first stage is an optimal code of rate R/L for an *i.i.d.* $\mathcal{N}(0,\nu^2)$ source. This implies that the residual $r_1 = s - A\beta^1$ satisfies $\|r_1\|^2/n \approx \nu^2 e^{-2R/L} \approx \nu^2(1-\frac{2R}{L})$. The residual r_1 acts as the 'source' sequence for the second stage, which is an optimal rate-distortion code for source variance $\nu^2 e^{-2R/L}$. At the end of the second stage, we have the residual r_2, which gets refined by the third stage, and so on. Each stage of refinement reduces the variance of the incoming residual by a factor of approximately $(1-{}^{2R}/_{L})$. Therefore, we expect that the final distortion $\|r_L\|^2/n \approx \nu^2(1-\frac{2R}{L})^L \leq \nu^2 e^{-2R}$.

However, since the rate R/L is infinitesimal, the deviations from the expected distortion in each stage can be significant. The essence of the proof of Theorem 5 is in analyzing these deviations, and showing that the final distortion $\|r_L\|^2/n$ is close to the typical value $\nu^2 e^{-2R}$. We note that such a "hard-decision" successive cancellation approach does not work well for AWGN channel decoding, i.e., the section error rate would decay much slower than exponentially with block length n. One explanation for this is that in channel coding, there is a unique codeword that the decoder has to determine, whereas in lossy compression, the number of good codewords is exponential in n when the rate is larger than the rate-distortion function.

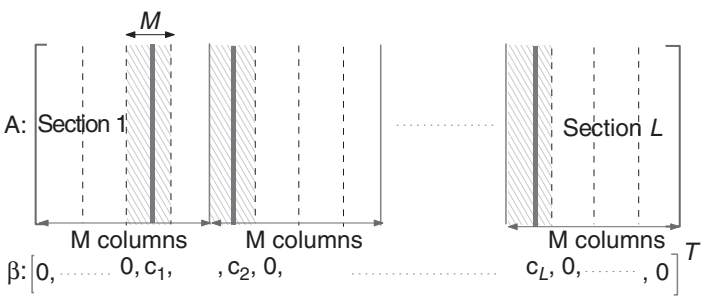
With a Gaussian design matrix, the running time and memory required for the successive cancellation encoder are $O(nML)$. As in channel coding, implementing the compressor with a Hadamard-based design matrix can lead to significant speedup and memory savings.

5. Multi-terminal Source and Channel Coding with SPARCs

Coding schemes that achieve the optimal rate-regions for several multi-terminal source and channel coding models often use the following ingredients: i) rate-optimal point-to-point source and channel codes, and ii) combining or splitting these point-to-point codes via superposition or binning [17].

Superposition with SPARCs: Superposition is easy to implement with SPARCs since the structure of the code itself is motivated by the idea of superposition! Indeed, to construct a superposition codebook with rates R_1 and R_2, use two design matrices A_1, A_2 with rates R_1, R_2, each with block length n. Then the concatenated SPARC defined by the matrix $A := [A_1, A_2]$ defines a superposition codebook with sum-rate $R_1 + R_2$. The message vector β is $[\beta_1, \beta_2]^T$, with β_1 and β_2 being the messages corresponding to the rate R_1 and rate R_2 SPARCs, respectively.

Binning with SPARCs [42]: We now describe how to bin a rate R_1 SPARC (with 2^{nR_1} codewords) into 2^{nR} bins, where $R < R_1$. Fix the parameters M, L, n of the design matrix A such that $L \log M = nR_1$.



As shown above, divide each section of A into sub-sections consisting of M' columns each. Then each *bin* is indexed by picking one sub-section from each section. For example, the collection of shaded sub-sections in the figure together forms one bin. The key observation is that each bin is a sub-matrix of A that defines a rate $(R_1 - R)$ SPARC with parameters (n, L, M'). Since we have (M/M') sub-section choices in each of the L sections, the total number of bins is $(M/M')^L$. Choosing M' such that $L \log M' = n(R_1 - R)$, we have 2^{nR} bins as required.

We have divided a higher rate SPARC of rate R_1 into 2^{nR} bins, each of which is a rate $(R_1 - R)$ SPARC. Note that the sub-matrices defining the bins have overlapping sub-sections, just like the SPARC codewords have overlapping columns. It was shown in [42], this superposition and binning constructions described above let us construct rate-optimal SPARCs for a variety of Gaussian multi-terminal models including broadcast channels, multiple-access channels, as well as source/channel coding with decoder/encoder side-information.

6. Open Questions

We conclude with a list of open questions in each of the topics discussed in this survey.

AWGN Channel Coding

- Theoretical guarantees for Hadamard designs: Current analysis techniques for both the AMP decoder and the adaptive successive decoder depend on the Gaussianity of A. Empirically, Hadamard-based SPARCs have very similar error performance to Gaussian ones (and much lower complexity), but there are no existing theoretical bounds.

- Getting closer to C at moderate block lengths: One idea in this direction is to combine power allocation techniques with spatially-coupled design matrices to boost empirical performance at rates close to C.

- Polynomial gap from C: A major open problem is to design a feasible SPARC decoder whose gap from capacity (for a fixed error probability) provably shrinks as $O(\frac{1}{n^a})$ for some $a \in (0, \frac{1}{2})$? With optimal decoding, the analysis in [13] shows that the gap from capacity for SPARCs is close to order The current analysis for the feasible decoders proposed so far suggests that the gap from capacity is of order $1/(\log n)^c$, where the constant $c \gtrsim 1$ varies according to the decoder.

- Generalizing the sparse regression construction: An interesting direction is to extend SPARCs to models such as fading channels and MIMO channels. A more general question is to construct efficient codes for other memoryless channels: There has been some recent work in this direction [43].

Lossy Compression

- Smaller gap from $D^*(R)$: Can we design feasible encoders with better compression performance, i.e., whose gap from $D^*(R)$ is smaller than $O(\log\log n/\log n)$? In particular, can we design soft-decision based encoders, e.g., an AMP encoder?

- Bernoulli dictionaries: Can one extend the results for optimal encoding and successive cancellation encoding (which are proved for Gaussian dictionaries) to dictionaries with *i.i.d.* ± 1 entries? For AWGN channels, SPARC ML decoding with *i.i.d.* ± 1 design matrices has been analyzed in [14].

- Finite-alphabet lossy compression: Can one use SPARC-like constructions to compress to finite alphabet sources, e.g., binary sources with Hamming distortion?

Multi-terminal models

- Approaching the Shannon limits with feasible encoding and decoding: The construction in Sec. 5 implements binning by nesting a lower-rate source/channel code inside a higher-rate channel/source code. Since the optimal power allocation for feasible encoding/decoding will depend on the rate, we may not be able to ensure that the SPARC power allocation is simultaneously optimal for both the high-rate and low-rate codes.

  An open question is: how to do we design good PA schemes for problems that require binning so that both the high-rate and low-rate codes are close to their Shannon limits, thereby ensuring that the overall rate is also near-optimal? We note that PA is not an issue when we use optimal encoding and decoding, as flat power-allocation is sufficient at all rates.

- Implementing SPARCs at near-optimal rates for basic models with binning (such as Gaussian Wyner-Ziv and 'writing on dirty paper') will pave the way to construct low-complexity, rate-optimal codes for a variety of Gaussian multi-terminal models such as multiple descriptions, distributed lossy compression, and relay channels.

Acknowledgements

This survey is based on work done in collaboration with Sanghee Cho, Adam Greig, Antony Joseph, Cynthia Rush, and Tuhin Sarkar. The work was supported in part by the National Science Foundation under Grant CCF-1217023, and by a Marie Curie Career Integration Grant (GA No. 631489) from the European Commission.

References

[1] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Transactions on Communications*, vol. 44, pp. 1261–1271, Oct 1996.

[2] T. Richardson and R. Urbanke, *Modern Coding Theory.* Cambridge University Press, 2008.

[3] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, pp. 3051–3073, July 2009.

[4] S. Korada and R. Urbanke, "Polar codes are optimal for lossy source coding," *IEEE Trans. Inf. Theory*, vol. 56, pp. 1751–1768, April 2010.

[5] S. Kudekar, T. Richardson, and R. L. Urbanke, "Spatially coupled ensembles universally achieve capacity under belief propagation," *IEEE Trans. Inf. Theory*, vol. 59, pp. 7761–7813, December 2013.

[6] G. D. Forney and G. Ungerboeck, "Modulation and coding for linear gaussian channels," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2384–2415, 1998.

[7] A. Guillén i Fàbregas, A. Martinez, and G. Caire, *Bit-interleaved coded modulation.* Now Publishers Inc, 2008.

[8] G. Böcherer, F. Steiner, and P. Schulte, "Bandwidth efficient and rate-matched low-density parity-check coded modulation," *IEEE Transactions on Communications*, vol. 63, no. 12, pp. 4651–4665, 2015.

[9] R. Zamir, *Lattice Coding for Signals and Networks: A Structured Coding Approach to Quantization, Modulation, and Multiuser Information Theory*. Cambridge University Press, 2014.

[10] U. Erez and R. Zamir, "Achieving 1/2 log(1 + *snr)* on the AWGN channel with lattice encoding and decoding," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2293–2314, 2004.

[11] N. Sommer, M. Feder, and O. Shalvi, "Low-density lattice codes," *IEEE Trans. Inf. Theory*, vol. 54, no. 4, pp. 1561–1585, 2008.

[12] Y. Yan, L. Liu, C. Ling, and X. Wu, "Construction of capacity-achieving lattice codes: Polar lattices," *arXiv preprint arXiv:1411.0187*, 2014.

[13] A. Barron and A. Joseph, "Least squares superposition codes of moderate dictionary size are reliable at rates up to capacity," *IEEE Trans. on Inf. Theory*, vol. 58, pp. 2541–2557, Feb. 2012.

[14] Y. Takeishi, M. Kawakita, and J. Takeuchi, "Least squares superposition codes with Bernoulli dictionary are still reliable at rates up to capacity," *IEEE Trans. Inf. Theory*, vol. 60, pp. 2737–2750, May 2014.

[15] Y. Takeishi and J. Takeuchi, "An improved upper bound on block error probability of least squares superposition codes with unbiased bernoulli dictionary," in *Proc. IEEE Int. Symp. Inf. Theory*, pp. 1168–1172, 2016.

[16] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley and Sons, 2012.

[17] A. El Gamal and Y.-H. Kim, *Network Information Theory.* Cambridge University Press, 2011.

[18] A. Joseph and A. R. Barron, "Fast sparse superposition codes have near exponential error probability for $R < C$,"*IEEE Trans. Inf. Theory*, vol. 60, pp. 919–942, Feb. 2014.

[19] C. Rush, A. Greig, and R. Venkataramanan, "Capacity-achieving sparse superposition codes via approximate message passing decoding," *arXiv:1501.05892*, 2015. (Shorter version appeared in ISIT '15).

[20] A. R. Barron and S. Cho, "High-rate sparse superposition codes with iteratively optimal estimates," in *Proc. IEEE Int. Symp. Inf. Theory*, 2012.

[21] S. Cho and A. Barron, "Approximate iterative bayes optimal estimates for high-rate sparse superposition codes," *in Sixth Workshop on Information-Theoretic Methods in Science and Engineering, 2013.*

[22] S. Cho, *High-dimensional regression with random design, including sparse superposition codes.* PhD thesis, Yale University, 2014.

[23] J. Barbier and F. Krzakala, "Approximate message-passing decoder and capacity-achieving sparse superposition codes," *arXiv:1503.08040, 2015.*

[24] 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.

[25] A. Montanari, "Graphical models concepts in compressed sensing," in *Compressed Sensing* (Y. C. Eldar and G. Kutyniok, eds.), pp. 394–438, Cambridge University Press, 2012.

[26] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," *IEEE Trans. Inf. Theory,* pp. 764–785, 2011.

[27] M. Bayati and A. Montanari, "The LASSO risk for Gaussian matrices," *IEEE Trans. Inf. Theory,* vol. 58, no. 4, pp. 1997–2017, 2012.

[28] F. Krzakala, M. Mézard, F. Sausset, Y. Sun, and L. Zdeborová, "Probabilistic reconstruction in compressed sensing: algorithms, phase diagrams, and threshold achieving matrices," *Journal of Statistical Mechanics: Theory and Experiment*, no. 8, 2012.

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

[30] D. L. Donoho, A. Javanmard, and A. Montanari, "Information-theoretically optimal compressed sensing via spatial coupling and approximate message passing," *IEEE Trans. Inf. Theory,* pp. 7434–7464, Nov. 2013.

[31] C. Rush and R. Venkataramanan, "Finite sample analysis of approximate message passing," in *Proc. IEEE Int. Symp. Inf. Theory,* 2016. Full version: https://arxiv.org/abs/1606.01800.

[32] "Coded modualtion library." Online: http://www.iterativesolutions.com/Matlab.htm.

[33] J. Barbier, C. Schülke, and F. Krzakala, "Approximate message-passing with spatially coupled structured operators, with applications to compressed sensing and sparse superposition codes," *Journal of Statistical Mechanics: Theory and Experiment*, no. 5, 2015.

[34] J. Barbier, M. Dia, and N. Macris, "Proof of threshold saturation for spatially coupled sparse superposition codes," in *Proc. IEEE Int. Symp. Inf. Theory,* 2016.

[35] R. Venkataramanan, A. Joseph, and S. Tatikonda, "Lossy compression via sparse linear regression: Performance under minimum-distance encoding," *IEEE Trans. Inf. Thy*, vol. 60, pp. 3254–3264, June 2014.

[36] R. Venkataramanan and S. Tatikonda, "The rate-distortion function and error exponent of sparse regression codes with optimal encoding," *arXiv:1401.5272, 2014.* (Shorter version appeared in ISIT '14).

[37] K. Marton, "Error exponent for source coding with a fidelity criterion," *IEEE Trans. Inf. Theory,* vol. 20, pp. 197–199, Mar 1974.

[38] S. Ihara and M. Kubo, "Error exponent for coding of memoryless Gaussian sources with a fidelity criterion," IEICE Trans. Fundamentals, vol. E83-A, pp. 1891–1897, Oct. 2000.

[39] A. Coja-Oghlan and L. Zdeborová, "The condensation transition in random hypergraph 2-coloring," in *Proc. 23rd Annual ACM-SIAM Symp. on Discrete Algorithms (SODA),* pp. 241–250, 2012.

[40] A. Coja-Oghlan and D. Vilenchik, "Chasing the k-colorability threshold," in *Proc. IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS),* pp. 380–389, 2013.

[41] R. Venkataramanan, T. Sarkar, and S. Tatikonda, "Lossy compression via sparse linear regression: Computationally efficient encoding and decoding," *IEEE Trans. Inf. Theory,* vol. 60, pp. 3265–3278, June 2014.

[42] R. Venkataramanan and S. Tatikonda, "Sparse regression codes for multi-terminal source and channel coding," in *50th Allerton Conf. on Commun., Control, and Computing,* 2012.

[43] J. Barbier, M. Dia, and N. Macris, "Threshold saturation of spatially coupled sparse superposition codes for all memoryless channels," in *Proc. Inf. Theory Workshop,* 2016.