

Binary MIMO Detection via Homotopy Optimization and Its Deep Adaptation

Mingjie Shao and Wing-Kin Ma

[†]Department of Electronic Engineering, The Chinese University of Hong Kong,
Hong Kong SAR of China

February 4, 2021

Abstract

In this paper we consider maximum-likelihood (ML) MIMO detection under one-bit quantized observations and binary symbol constellations. This problem is motivated by the recent interest in adopting coarse quantization in massive MIMO systems—as an effective way to scale down the hardware complexity and energy consumption. Classical MIMO detection techniques consider unquantized observations, and many of them are not applicable to the one-bit MIMO case. We develop a new non-convex optimization algorithm for the one-bit ML MIMO detection problem, using a strategy called homotopy optimization. The idea is to transform the ML problem into a sequence of approximate problems, from easy (convex) to hard (close to ML), and with each problem being a gradual modification of its previous. Then, our attempt is to iteratively trace the solution path of these approximate problems. This homotopy algorithm is well suited to the application of deep unfolding, a recently popular approach for turning certain model-based algorithms into data-driven, and performance enhanced, ones. While our initial focus is on one-bit MIMO detection, the proposed technique also applies naturally to the classical unquantized MIMO detection. We performed extensive simulations and show that the proposed homotopy algorithms, both non-deep and deep, have satisfactory bit-error probability performance compared to many state-of-the-art algorithms. Also, the deep homotopy algorithm has attractively low computational complexity.

1 Introduction

MIMO detection has been, for decades, at the center of some very important developments in signal processing, communications and beyond. Its problem statement is simple—to detect a multitude of finite-alphabet variables from a noisy observation of their linearly entangled counterparts. The problem is, however, very challenging if the goal is to build an MIMO detection algorithm that can achieve optimal or near-optimal detection performance with a low implementation cost, particularly, for large problem sizes. The above goal has attracted generations of researchers' innovations, e.g., around the 1990's for multiuser code division multiple access, around the 2000's for multi-antenna systems, and recently for massive antenna arrays; and the consequence is a rich, enduring, collection of methods—such as sphere decoding methods [1, 2] for implementing the optimal maximum-likelihood (ML) detection via combinatorial search; convex relaxation methods [3–7] for efficient ML approximation via convex optimization; lattice reduction-aided methods [8] for improving the channel conditioning, and thereby performance, via basis design under integer coefficient

constraints; soft detection methods [9, 10] for posterior inference of the unknown finite-alphabet variables via various approximations; to name a few. These developments show intimate connections with studies in other areas, such as complexity theory, combinatorial and continuous optimization, statistical inference, etc. Also, recent research takes insight from approximate message passing in compressive sensing [11] to tackle MIMO detection [12]. Hence, MIMO detection is a topic of far-reaching implications. We refer the reader to the literature, e.g., [13], for a chronological account of the MIMO detection developments and a coverage of the numerous algorithms therein.

Lately, MIMO detection has been igniting new interest in two recent research directions, namely, coarsely quantized massive MIMO and deep learning for communications. We separately describe them in the following two subsections.

1.1 Coarsely Quantized Massive MIMO Detection

As a serious impediment to the real-world massive MIMO implementation, it is well-known that if every radio-frequency front end of a massive MIMO system is equipped with a high-resolution analog-to-digital converter (ADC) or digital-to-analog converter (DAC)—as in conventional MIMO, the system will be very expensive to implement in terms of hardware complexity and energy consumption. In view of this, there has been movement toward the use of low-resolution ADCs or DACs in massive MIMO. One arising problem in this context is MIMO detection under coarsely quantized observations—particularly one-bit, or sign-only, observations.

A challenge with coarsely quantized MIMO detection is that the classical MIMO detection methods we have studied for decades cannot be straightforwardly extended to the quantized case; this is owing to the different likelihood function in the quantized case. Some powerful classical methods, such as sphere decoding, semidefinite relaxation [5, 6] and lattice reduction, are inapplicable in the quantized case because they were designed to exploit the convex quadratic structure of the ML objective function in the unquantized case. Some researchers study the impacts of coarse quantizations on some classical MIMO detection methods (e.g., zero forcing) [14], while others redesign the MIMO detection methods to better harness the problem structure. For the latter, Studer and Durisi took inspiration from the classical linear minimum-mean-square-error (MMSE) detection to derive a variant in the quantized case [15]; Choi, Mo and Heath developed a convex sphere relaxation of one-bit ML MIMO detection [16]; Jeon *et al.* devised a sphere decoder for one-bit ML MIMO detection [17] (its appearance is very different from those we see in classical MIMO detection); approximate message passing was redesigned for quantized MIMO detection in [18, 19]; coding theory-inspired detection algorithms were introduced in [20–22]. Also, quantized MIMO detection for orthogonal frequency-division modulation (OFDM)-MIMO—a more realistic, but also computationally more challenging, scenario—was studied in [15, 23–25].

1.2 Deep Learning for MIMO Detection

Recently, the tremendous successes of deep neural networks in natural language processing, computer vision and machine learning have sparked widespread interest in the communications community. In particular, we have seen a variety of emerging deep network applications for communications, such as end-to-end communication system design [26], multiuser power control [27], and, as our main interest, MIMO detection [28–34]. Neural networks for MIMO detection was already considered in around the 1990’s [35, 36]; it is worth noting that the motivation at the time lies in neural networks’ ability to generate nonlinear decision regions, which the then-popular methods of

linear and decision-feedback detection have limitations. In the recent renewed interest, we have so far not seen a report on successfully training a deep neural network—specifically, that under a standard network architecture—that gives consistently near-optimal detection performance and good training stability for a broad range of MIMO settings. Instead, attention appears to have been drawn to the design approach of *deep unfolding*, which has a flavor of leveraging on both the existing model-based MIMO detection methods and the data-driven deep learning approach.

Deep unfolding was first introduced in the context of sparse coding by Gregor and LeCun [37], and it has recently received much attention [38, 39]. The rationale is to see an existing iterative algorithm, e.g., the iterative shrinkage thresholding algorithm (ISTA) in sparse coding (see [37]), as a deep network. It intends to learn a better algorithm than its predecessor algorithm by untying some parameters of the predecessor, and then by learning those parameters from data. Also, one can modify part of the structure of the predecessor to make it more general, and learn the new structure from data. The result is a structured deep network whose structures preserve some of the structures prescribed under a model-based framework. In the work by Gregor and LeCun, they showed that the learnt ISTA requires much less number of iterations, or layers, to achieve performance comparable to ISTA.

Deep unfolding was first applied to MIMO detection by Samuel, Diskin and Wiesel in 2017 [28]. The algorithm there, called DetNet, is a deep unfolding of a non-convex projected gradient algorithm for ML MIMO detection. We have seen growing interest with deep learning for MIMO detection since the DetNet work. For example, the works [30, 31] studied deep network structures similar to DetNet. The works [32, 33] studied the deep unfolding of approximate message passing. Apart from deep unfolding, the work [34] applied deep learning to learn the sphere radius in sphere decoding.

1.3 Present Contribution and Related Works

The present contribution considers one-bit ML MIMO detection under binary symbol constellations. We tackle the problem by a non-convex optimization strategy, namely, homotopy optimization; see [40–45] and the references therein. Also called continuation or graduated non-convexity, homotopy optimization is an idea that arose independently from a variety of applications in different fields, such as molecular conformation [40], sparse optimization [42] and neural network training [44]. Homotopy methods can be vastly different from one application to another, but they often follow a common principle. Specifically, it entails a problem transformation—called homotopy map—that has the flexibility of either making the transformed problem easier to solve (e.g., convex) but less accurate in approximating the original problem; or making the transformed problem harder but closer to the original. Then, the attempt is to iteratively trace the solution path of a sequence of such approximate problems, from easy to hard and in a gradual fashion. In the context of MIMO detection, our empirical experience is the following: tackling the ML MIMO detection problem directly by a straight application of a non-convex algorithm may result in poor detection performance, owing to convergence to poor local minima; but handling the problem indirectly via the aforementioned gradually easy-to-hard optimization principle may lead to better performance.

Homotopy optimization offers us a principle, not a numerical algorithm that fits all problems. We often need to find a suitable problem transformation for the application at hand. We employ the non-convex continuous reformulation of binary optimization in [46], as well as the efficient first-order algorithm design therein. This technique was previously proposed by us to tackle a one-bit MIMO precoding problem. In the present contribution, we incorporate this technique into the theme of

homotopy optimization and explore its potential in MIMO detection. Also we enrich the result by connecting homotopy optimization with Lagrangian dual relaxation (LDR), as will be shown in Section 3.2. It is worth noting that the LDR notion plays a vital role in ML MIMO detection; e.g., semidefinite relaxation and regularized lattice decoding can be interpreted as outcomes of LDR [47, 48].

In our development, we found that the homotopy algorithm has a structure favorable for deep unfolding. This motivates to consider deep unfolding of our homotopy algorithm. Our deep unfolding involves mild untying and structure modification, which means that the structure does not change a lot. By empirical experience, our deep homotopy network is easy to train. Also our deep homotopy network is trained to cater for different channels, rather than a fixed channel. Note that the former is more preferable since it allows us to have the expensive training done offline; the latter requires online training for every given channel, and the subsequent real-time computational overheads may be significant.

The contributions of this work are summarized as follows.

1. As a new attempt to tackle the challenge of efficient high-performance MIMO detection, we propose a non-convex homotopy optimization method for one-bit ML MIMO detection under binary symbol constellations. By extensive simulations, we show that the homotopy algorithm yields considerably better detection performance (specifically, bit-error rate performance) than some state-of-the-art algorithms. However it has a relatively high complexity requirement, compared to the state-of-the-art.
2. We apply deep unfolding to the proposed homotopy algorithm to learn a better algorithm. Simulation results show that the deep-unfolded homotopy algorithm has detection performance comparable to the original homotopy algorithm, and it does so with a much lower complexity requirement—20 layers, or iterations, in all of our tested MIMO settings. The deep-unfolded homotopy algorithm also runs faster than the state-of-the-art algorithms.
3. While our initial focus is on one-bit MIMO detection, the proposed homotopy method is also applicable to classical (unquantized) MIMO detection. Simulation results show that the homotopy algorithms (deep and non-deep) provide near-optimal detection performance, and the other empirical observations are similar to those in the one-bit case.

We should discuss related works. In the decades of MIMO detection research, non-convex optimization was considered; see, e.g., [49, 50]. But the notion of homotopy optimization for attempting to avoid poor local minima, as well as our non-convex continuous reformulation for binary optimization, appear to have not been previously applied to MIMO detection. Also, our homotopy method for binary optimization appears to have not been seen in the prior homotopy optimization literature. Furthermore, the majority of the current deep MIMO detection studies consider the classical case. Our deep unfolding design covers both the classical and one-bit MIMO cases.

For the sake of reproducible research, we have released the source code at <http://www.ee.cuhk.edu.hk/~wkma/mimo/> or at <https://github.com/mjshao-cuhk/DeepHOTML>.

2 Problem Statement

2.1 Model and ML Detection

Let us begin by posing our problem in its generic abstract form. Consider a data model

$$\mathbf{y} = \text{sgn}(\mathbf{H}\mathbf{x} + \mathbf{v}), \quad (1)$$

where $\mathbf{y} \in \{-1, 1\}^M$ is an observed data vector; $\mathbf{x} \in \{-1, 1\}^N$ is a binary vector; sgn denotes the element-wise sign function; $\mathbf{H} \in \mathbb{R}^{M \times N}$ is a system matrix; $\mathbf{v} \in \mathbb{R}^M$ is element-wise independent and identically distributed (i.i.d.) Gaussian noise, with mean zero and variance σ^2 . Our task is to detect \mathbf{x} from \mathbf{y} , given that \mathbf{H} and σ are known. We do so by pursuing the maximum-likelihood (ML) detection approach—which guarantees the minimum error probability of detecting \mathbf{x} under the assumption of element-wise uniform i.i.d. \mathbf{x} . Under the model (1), the ML detector takes the form

$$\hat{\mathbf{x}}_{\text{ML}} = \arg \min_{\mathbf{x} \in \{-1, 1\}^N} f(\mathbf{x}) \triangleq - \sum_{i=1}^M \log \Phi \left(\frac{y_i \mathbf{h}_i^T \mathbf{x}}{\sigma} \right), \quad (2)$$

where $\Phi(t) = \int_{-\infty}^t \frac{1}{\sqrt{2\pi}} e^{-\tau^2/2} d\tau$, and \mathbf{h}_i denotes the i th row of \mathbf{H} ; see [16, 51]. The ML problem (2) is a discrete optimization problem due to the binary constraint $\mathbf{x} \in \{-1, 1\}^N$, and a method capable of finding $\hat{\mathbf{x}}_{\text{ML}}$ tractably—given any instance $(\mathbf{y}, \mathbf{H}, \sigma)$ —appears to be unavailable. In fact, it is computationally challenging to solve problem (2) exactly when the problem dimension N is large; e.g., complete enumeration of all points in $\{-1, 1\}^N$ is impossible for large N . Our problem is to find an efficient strategy to tackle problem (2).

The above problem arises in one-bit massive MIMO detection which has spurred great interest recently. To describe it, consider an uplink multiuser MIMO scenario where multiple single-antenna users simultaneously transmit information symbols to a massive MIMO base station (BS); and, to reduce the hardware cost, the BS employs low-resolution ADCs. Assuming flat fading channels, the signal model of the above scenario can be formulated as

$$\mathbf{y}_C = \mathcal{Q}(\mathbf{H}_C \mathbf{x}_C + \mathbf{v}_C), \quad (3)$$

where $\mathbf{y}_C \in \mathbb{C}^{M_C}$ is a receive vector whose i th element $y_{C,i}$ represents the complex baseband received signal at the i th antenna of the BS; $\mathbf{x}_C \in \mathcal{X}_C^{N_C}$ is a transmit vector whose i th element $x_{C,i}$ is the symbol transmitted by the i th user; M_C and N_C are the numbers of receive antennas and users, respectively; $\mathcal{X}_C \subset \mathbb{C}$ is the symbol constellation set; $\mathbf{H}_C \in \mathbb{C}^{M_C \times N_C}$ is the MIMO channel; \mathcal{Q} denotes an element-wise quantization function; $\mathbf{v}_C \in \mathbb{C}^{M_C}$ is element-wise i.i.d. circular Gaussian noise with mean zero and variance σ_C^2 . Let us consider the case of one-bit quantization $\mathcal{Q}(\mathbf{y}_C) = \text{sgn}(\Re\{\mathbf{y}_C\}) + j \cdot \text{sgn}(\Im\{\mathbf{y}_C\})$. Also we focus on an in-phase quadrature-phase binary constellation $\mathcal{X}_C = \{\pm 1 \pm j\}$, which is 4-ary quadratic amplitude modulation (QAM) or quaternary phase shift keying (QPSK) constellation. Under the aforementioned one-bit quantization and binary constellation, the complex-valued model (3) can be rewritten as the real-valued model (1) by setting $(M, N) = (2M_C, 2N_C)$, $\sigma^2 = \sigma_C^2/2$,

$$\mathbf{y} = \begin{bmatrix} \Re(\mathbf{y}_C) \\ \Im(\mathbf{y}_C) \end{bmatrix}, \mathbf{x} = \begin{bmatrix} \Re(\mathbf{x}_C) \\ \Im(\mathbf{x}_C) \end{bmatrix}, \mathbf{v} = \begin{bmatrix} \Re(\mathbf{v}_C) \\ \Im(\mathbf{v}_C) \end{bmatrix}, \mathbf{H} = \begin{bmatrix} \Re(\mathbf{H}_C) & -\Im(\mathbf{H}_C) \\ \Im(\mathbf{H}_C) & \Re(\mathbf{H}_C) \end{bmatrix}. \quad (4)$$

By the above relation, the ML detector (2) applies to the one-bit MIMO model (3).

2.2 Some Aspects

Some basic nature of the ML problem (2) should be noted. First, the objective function f of problem (2) is convex (this is mainly because $-\log \Phi(x)$ is convex [52]). The convexity of f can be leveraged to develop efficient approximate ML detectors. In [16], the authors considered a convex sphere relaxation of the ML problem (2); specifically,

$$\min_{\|\mathbf{x}\|^2 \leq N} f(\mathbf{x}), \quad (5)$$

where $\|\cdot\|$ denotes the Euclidean norm. Second, it is natural to question whether we can reliably perform MIMO detection from one-bit observations. From the one-bit model (1) and its ML detector (2), it is intuitively unobvious why or whether the true vector \mathbf{x} can be recovered from the heavily quantized observation \mathbf{y} . We provide a clue to this basic question by the following simple result.

Fact 1 *Suppose that*

- (a) $\mathbf{h}_1, \dots, \mathbf{h}_M$ are i.i.d.;
- (b) the probability density function of the \mathbf{h}_i 's, denoted by $q(\mathbf{h})$, is continuous on its support;
- (c) the support of $q(\mathbf{h})$ is \mathbb{R}^N .

Consider $M \rightarrow \infty$ such that

$$\frac{1}{M}f(\mathbf{x}) \xrightarrow{P} \tilde{f}(\mathbf{x}) \triangleq -\mathbb{E} \left[\log \Phi \left(\frac{y\mathbf{h}^T \mathbf{x}}{\sigma} \right) \right], \quad (6)$$

provided the expectation exists. Here, \xrightarrow{P} denotes convergence in probability; y and \mathbf{h} represent random variables associated with the realizations y_i 's and \mathbf{h}_i 's, respectively; the expectation $\mathbb{E}[\cdot]$ is taken with respect to y and \mathbf{h} . Then, the minimizer of $\tilde{f}(\mathbf{x})$ over \mathbb{R}^N is uniquely given by the true binary vector in the signal model (1). It follows that the ML problem

$$\min_{\mathbf{x} \in \{-1,1\}^N} \tilde{f}(\mathbf{x})$$

has its solution uniquely given by the true binary vector.

The result in Fact 1 was first reported in [51, Lemma 2] for the case of i.i.d. Gaussian \mathbf{h} and constant Euclidean-norm \mathbf{x} , and here we show a more general result. Before giving the proof, let us first discuss the implications. Fact 1 suggests that if the number of antennas at the BS is very large (true for massive MIMO), then the ML problem (2) may lead to correct recovery of the true binary vector. In fact, this large- M recovery result holds not only for the ML problem, but also for unconstrained relaxation of problem (2) (i.e., replacing the constraint $\{-1,1\}^N$ by \mathbb{R}^N) and the sphere relaxation in (5); it also holds if the true vector \mathbf{x} in the signal model (1) is drawn from a higher-order constellation set (or even \mathbb{R}^N). In addition, the requirement with the channel distribution is quite general; e.g., it works for i.i.d. Gaussian channels, correlated Gaussian channels (with respect to users), or other continuously distributed channels (with support \mathbb{R}^N).

Proof of Fact 1: Our proof is different from [51, Lemma 2], which uses stochastic orders. We employ a more basic proof, taking proof ideas from the consistency property of ML estimation.

To avoid notation overlap, re-denote the true binary vector in the signal model (1) as \mathbf{x}_0 . The probability mass function of y_i given \mathbf{h}_i and \mathbf{x}_0 is

$$p(y_i|\mathbf{h}_i, \mathbf{x}_0) = \Phi\left(\frac{y_i \mathbf{h}_i^T \mathbf{x}_0}{\sigma}\right), \quad (7)$$

which can be shown from (1). We see that, for any $\mathbf{x} \neq \mathbf{x}_0$,

$$\begin{aligned} \tilde{f}(\mathbf{x}_0) - \tilde{f}(\mathbf{x}) &= \mathbb{E} \left[\log \left(\frac{p(y|\mathbf{h}, \mathbf{x})}{p(y|\mathbf{h}, \mathbf{x}_0)} \right) \right] \\ &= \sum_{y \in \{-1, 1\}} \int_{\mathbb{R}^N} \log \left(\frac{p(y|\mathbf{h}, \mathbf{x})}{p(y|\mathbf{h}, \mathbf{x}_0)} \right) p(y|\mathbf{h}, \mathbf{x}_0) q(\mathbf{h}) d\mathbf{h} \\ &\leq \sum_{y \in \{-1, 1\}} \int_{\mathbb{R}^N} \left(\frac{p(y|\mathbf{h}, \mathbf{x})}{p(y|\mathbf{h}, \mathbf{x}_0)} - 1 \right) p(y|\mathbf{h}, \mathbf{x}_0) q(\mathbf{h}) d\mathbf{h} \\ &= 0, \end{aligned}$$

where the inequality is due to $\log(t) \leq t - 1$. The above inequality implies that \mathbf{x}_0 is a minimizer of $\tilde{f}(\mathbf{x})$ over \mathbb{R}^N . To show that \mathbf{x}_0 is the unique minimizer, note that equality in the above equation holds if and only if $p(y|\mathbf{h}, \mathbf{x}) = p(y|\mathbf{h}, \mathbf{x}_0)$ for every $y \in \{-1, 1\}$ and $\mathbf{h} \in \mathbb{R}^N$. The latter condition implies

$$\mathbf{h}^T \mathbf{x} = \mathbf{h}^T \mathbf{x}_0, \quad \text{for every } \mathbf{h} \in \mathbb{R}^N;$$

this is seen from (7) (the monotonicity of Φ should also be noted). Clearly the above equation does not hold for any $\mathbf{x} \neq \mathbf{x}_0$, which means that there does not exist $\mathbf{x} \neq \mathbf{x}_0$ such that $\tilde{f}(\mathbf{x}) = \tilde{f}(\mathbf{x}_0)$. The proof is complete. \blacksquare

3 A Homotopy Optimization Method

3.1 The Main Idea

Our strategy for tackling the ML problem (2) hinges on a very recently introduced penalty approach [46]. Consider a penalty, and possibly approximate, formulation of problem (2)

$$(P_\lambda) \quad \min_{\mathbf{x} \in [-1, 1]^N} F_\lambda(\mathbf{x}) \triangleq f(\mathbf{x}) - \lambda \|\mathbf{x}\|^2, \quad (8)$$

for a given parameter $\lambda \geq 0$. Intuitively, the idea is to encourage large value with every x_i^2 by imposing the penalty term $-\lambda \|\mathbf{x}\|^2$ in the objective, on the one hand, and limit $x_i^2 \leq 1$ on the other hand. In doing so, the optimal solution to problem (8) should be forced to $x_i^2 = 1$, or $x_i \in \{-1, 1\}$, if we apply a large λ . In fact, under a mild assumption, this intuition is correct:

Theorem 1 (a rephrased version of Theorem 2 in [46]) *Let f be a twice differentiable function, not necessarily the one in (2), and consider the corresponding problems in (2) and (8). Let $L_f > 0$ be a Lipschitz constant of the gradient of f on $[-1, 1]^N$, which must exist for twice differentiable f . For any $\lambda > L_f/2$, it holds that¹*

¹As a technical remark, Theorem 2 in [46] only showed statement (a) of Theorem 1. Statements (b)–(c) are straightforward corollaries of statement (a).

- (a) any locally optimal solution to problem (8) lies in $\{-1, 1\}^N$;
- (b) any globally optimal solution to problem (8) is also that to problem (2);
- (c) if \mathbf{x} is a stationary point of problem (8) and \mathbf{x} does not lie in $\{-1, 1\}^N$, then \mathbf{x} must be either a local maximum or a saddle point.

Hence, for a sufficiently large λ , we can employ problem (8) as an equivalent formulation of the ML problem (2).

As the main benefit, the penalty formulation (8) turns the discrete ML problem into a continuous, and convex constrained, optimization problem. As a result, we can use methods, such as the descent-based methods, to efficiently compute a stationary point of problem (8) (and hopefully a locally optimal solution to it). But we should also note that the penalty term $-\lambda\|\mathbf{x}\|^2$ in problem (8) makes the problem non-convex, and a descent-based algorithm can converge to a poor local minimum.

Algorithm 1 A homotopy strategy for tackling problem (2)

- 1: **given** an initial penalty parameter $\lambda_0 \geq 0$ and a starting point \mathbf{x}^0
 - 2: $k = 0$
 - 3: **repeat**
 - 4: $k = k + 1$
 - 5: set λ_k as increased version of λ_{k-1}
 - 6: run a descent-based algorithm, with \mathbf{x}^{k-1} as the starting point, to compute a stationary point of problem (P_{λ_k}) in (8), and store the solution obtained as \mathbf{x}^k
 - 7: **until** λ_k is larger than a pre-specified threshold.
 - 8: **output** \mathbf{x}^k
-

As an attempt to circumvent local minima, we consider the following strategy. Problem (8) for $\lambda = 0$, or problem (P_0) , is convex. This gives an intuition that problem (P_λ) in (8) should be easy for small λ , and hard for large λ . It is therefore natural to consider the optimization strategy in Algorithm 1, where we progressively increase λ , or the difficulty of the problem. Also, and just as important, we use the previous solution \mathbf{x}^{k-1} in an effort to find a good solution to problem (P_{λ_k}) . Imagine this: If \mathbf{x}^{k-1} is a globally optimal solution to problem $(P_{\lambda_{k-1}})$ and the change of λ_k relative to λ_{k-1} is small, then the optimization landscape may undergo only mild changes, and a descent-based algorithm starting with \mathbf{x}^{k-1} may “easily” descend to a close-by globally optimal solution to problem (P_{λ_k}) . Thus we may be able to find the ML solution by tracing a solution path of problem (P_λ) , from small λ to large λ . We give the reader more insight by pictorially illustrating the above described idea in Fig. 1, and by displaying the landscape of a 2D instance of problem (P_λ) in Fig. 2.

To demonstrate the benefits of the above optimization strategy, we show a simulation result here. We consider one-bit MIMO detection with $(M, N) = (256, 48)$. We perform detection by tackling the ML problem (2) via formulation (8), either with a fixed λ or with the progressively increasing λ strategy in Algorithm 1. The results are shown in Fig. 3. We observe that the progressively increasing λ strategy leads to better bit-error rate performance than fixing λ . In particular, using a large fixed λ yields unsatisfactory results, most likely due to convergence to poor local minima. Using a smaller fixed λ mitigates the undesirable effects, but doing so also weakens its approximation accuracy relative to the ML.

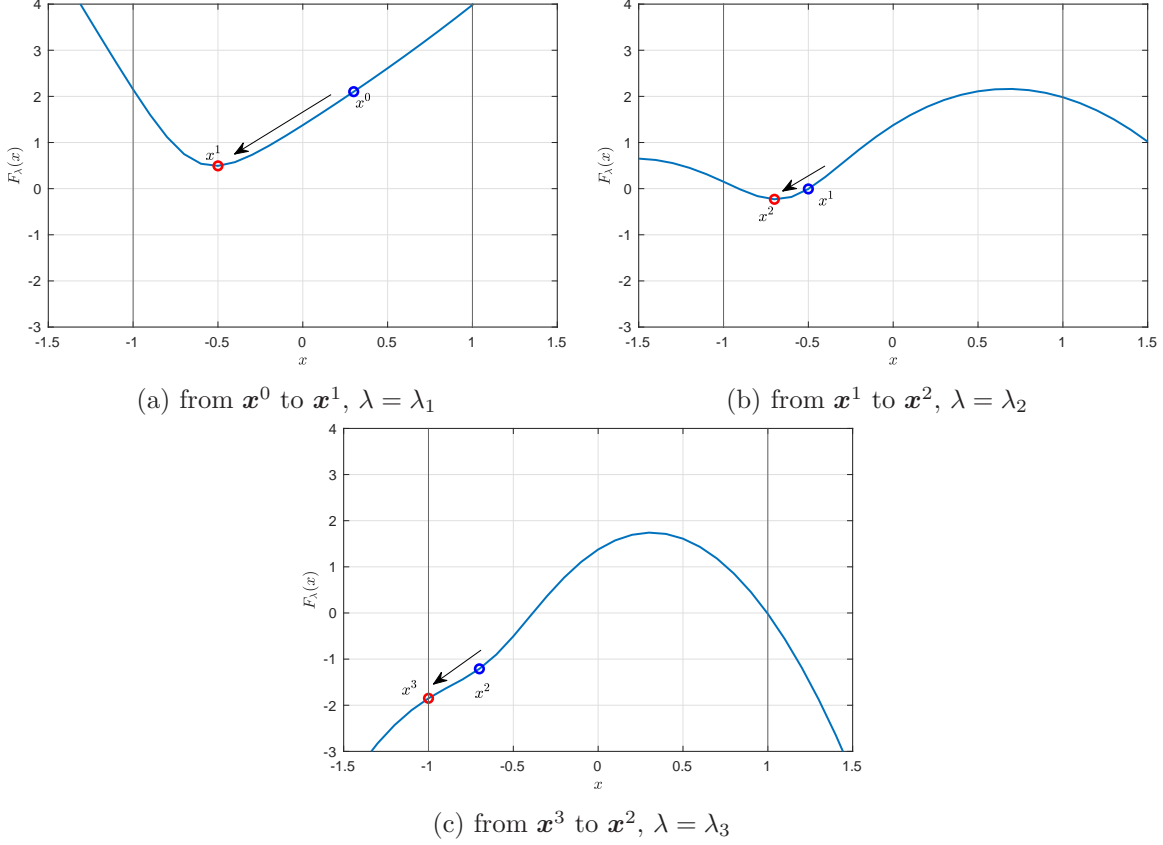


Figure 1: Illustration of how the homotopy method works.

Algorithm 1 may be taxonomized into the class of homotopy optimization methods; see [40–45] and the references therein. The principle of homotopy optimization is to first find a transformation, or a homotopy map, that maps the original problem to another problem. That transformation depends on a parameter, say, λ . For large λ , the transformed problem is close to the original problem but is hard to solve directly. For small λ , the transformed problem is easy to solve but poorly approximates the original problem. Then, we seek to find a solution (possibly approximate) to the original problem by attempting to trace the solution path of a sequence of such transformed problems, from easy to hard and in a gradually changing fashion, e.g., by the routine we saw in Algorithm 1 which is typical in homotopy methods.

Remark 1 (Related Work) We close this subsection by discussing the relationship of the proposed homotopy formulation and the existing methods. For $\lambda = 0$, formulation (8) reduces to convex box relaxation [3, 4, 7]. Thus, formulation (8) may be regarded as a non-convex enhancement of box relaxation. Moreover, the negative square penalty $-\lambda\|\mathbf{x}\|^2$ used in formulation (8) also appears in other contexts, such as low-density parity-check decoding [53] and one-bit precoding [54], to force the solution closer to a binary vector. A subtle but important difference is that the aforementioned studies often employ a fixed penalty parameter λ , while we use the homotopy optimization strategy to progressively adjust λ .

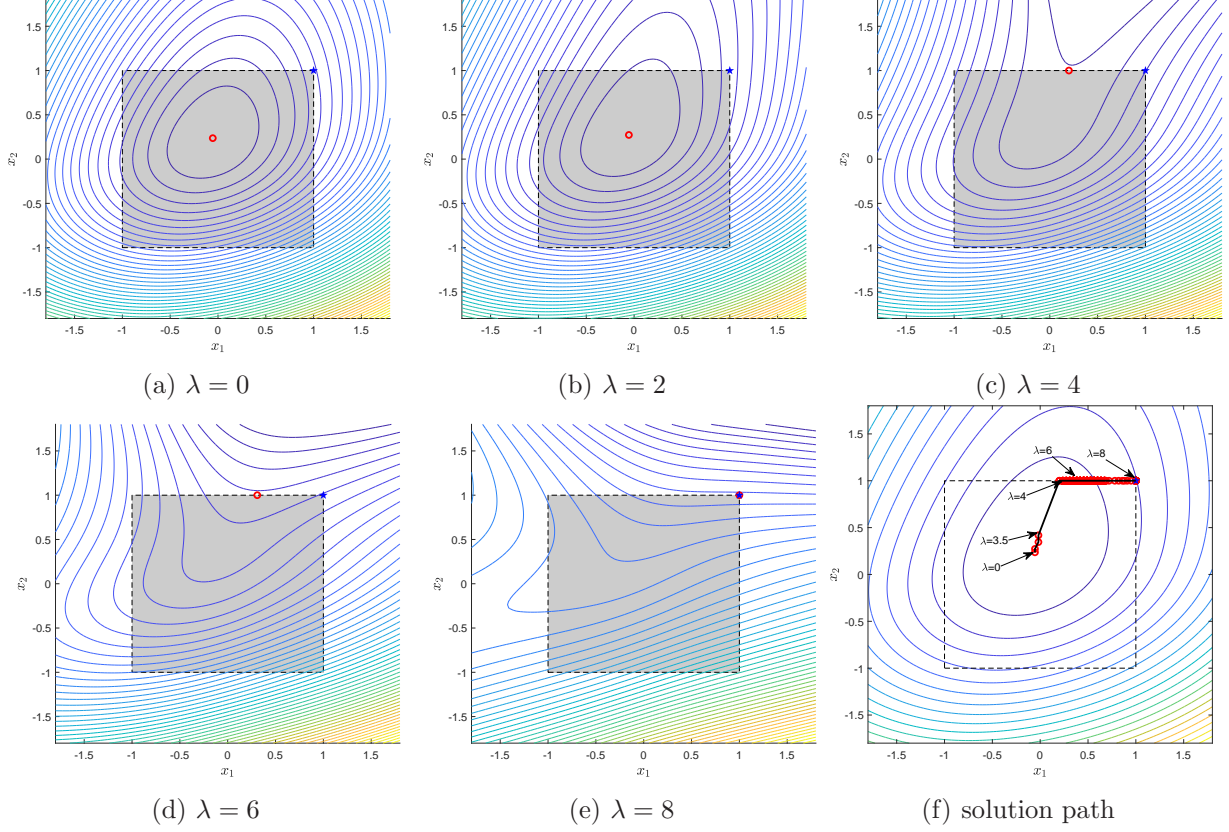


Figure 2: An instance of the landscape of problem (P_λ) in (8), with $N = 2$. The contour is $F_\lambda(\mathbf{x})$; the gray area is the feasible set; the red circle is the optimal solution to problem (P_λ) ; the blue star is the optimal ML solution; the instance is a randomly generated one for one-bit MIMO detection, with $M = 10, \sigma = 1$.

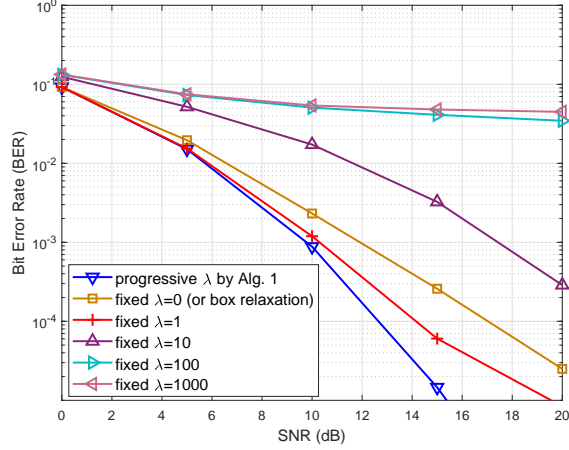


Figure 3: Performance of using formulation (8) to tackle the one-bit ML MIMO detection problem.

3.2 How to Choose the Penalty Sequence $\{\lambda_k\}$?

In the homotopy strategy in Algorithm 1, a crucial question is how the penalty parameter sequence $\{\lambda_k\}$ should be chosen. Having a small increase with λ_k at each iteration may allow us to follow the solution path better, but it may take too many iterations to do so. Having a large increase with λ_k , on the other hand, reduces the number of iterations but may also increase the risk of losing track of the solution path. By our experience, a heuristically chosen $\{\lambda_k\}$ works well—empirically. For example, we may choose $\lambda_k = \lambda_{k-1}c$ for some constant $c > 1$. But it is tempting to see whether some theory-guided selection rule exists. Here we show one such rule by drawing a connection with Lagrangian dual relaxation (LDR).

We start with studying an equivalent formulation of the ML problem (2)

$$\begin{aligned} f^* &= \min_{\mathbf{x} \in \mathcal{D}} f(\mathbf{x}) \\ \text{s.t. } &\|\mathbf{x}\|^2 \geq N \end{aligned} \quad (9)$$

where $\mathcal{D} = [-1, 1]^N$ denotes the problem domain, and f^* the optimal value. We are interested in the Lagrangian dual of the formulation in (9). Let

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda(N - \|\mathbf{x}\|^2)$$

be the Lagrangian of problem (9), and let

$$d(\lambda) = \min_{\mathbf{x} \in \mathcal{D}} L(\mathbf{x}, \lambda) \quad (10)$$

be the dual function. The dual problem of problem (9) is

$$d^* = \max_{\lambda \geq 0} d(\lambda) \quad (11)$$

where d^* denotes the optimal value of problem (11). At this point, note that the minimization problem in (10) is equivalent to the penalty formulation (8). Also, recall from the basic concepts of Lagrangian duality that the dual problem (11) is convex, and the weak duality inequality $f^* \geq d^*$ holds.

LDR is a method that exploits the dual problem to approximate the original problem; see, e.g., [47, 48]. It works by first finding the optimal solution to the dual problem (11), denoted herein by λ^* , and then finding the solution \mathbf{x} to the minimization problem in (10) for $\lambda = \lambda^*$. The duality gap $f^* - d^*$ provides strong indication of how well LDR approximates the original problem; the smaller the gap is, the better LDR should be. For the problem at hand, we show that the LDR in (11) is tight.

Theorem 2 *Consider the same problem settings in Theorem 1. The primal-dual problem pair (9) and (11) achieves zero duality gap $f^* - d^* = 0$.*

Proof: Let $\bar{\lambda}$ be any constant such that $\bar{\lambda} > L_f/2$. We have $d^* \geq d(\bar{\lambda}) = f^*$, where the equality above is due to Theorem 1. This, together with $f^* \geq d^*$, implies that $f^* = d^*$. ■

There is a connection between the preceding homotopy strategy and the LDR here. Consider finding the optimal solution to the dual problem (11) via the projected subgradient method [55], given by

$$\lambda_k = \max\{0, \lambda_{k-1} + \mu_k g(\lambda_{k-1})\}, \quad k = 1, 2, \dots \quad (12)$$

Here, $g(\lambda)$ denotes a subgradient of the dual function d at λ ; μ_k is a step size (see the literature [55] for step-size rules that guarantee that λ_k will converge to an optimal solution); λ_0 , the starting point, is assumed to be non-negative. By the subgradient calculus [55], $g(\lambda)$ is given by

$$g(\lambda) = N - \|\hat{\mathbf{x}}_\lambda\|^2, \quad (13)$$

where

$$\hat{\mathbf{x}}_\lambda \in \arg \min_{\mathbf{x} \in \mathcal{D}} L(\mathbf{x}, \lambda). \quad (14)$$

Putting (13)–(14) into (12), we can simplify the projected subgradient method (12) to

$$\lambda_k = \lambda_{k-1} + \mu_k(N - \|\hat{\mathbf{x}}_{\lambda_{k-1}}\|^2), \quad k = 1, 2, \dots \quad (15)$$

where we have used $\lambda_0 \geq 0$ and $\|\mathbf{x}\|^2 \leq N$ for any $\mathbf{x} \in \mathcal{D}$. The above projected subgradient method is seen to be closely related to the homotopy strategy in Algorithm 1: the former also increases λ_k at each iteration, and it requires us to solve problem (14)—an equivalent of the penalty formulation (8)—at each iteration. The challenge with realizing the projected subprojected method is that problem (14) is non-convex. But we can substitute the globally optimal solution $\hat{\mathbf{x}}_{\lambda_k}$ with an approximate solution, obtained by using $\hat{\mathbf{x}}_{\lambda_{k-1}}$ to warm-start a descent-based algorithm. The resulting (approximate) projected subgradient method will then be identical to the homotopy strategy in Algorithm 1, with a clearly defined penalty update rule as shown in (15).

Let us summarize. In the homotopy strategy in Algorithm 1, we may choose λ_k by

$$\lambda_k = \lambda_{k-1} + \mu_k(N - \|\mathbf{x}^{k-1}\|^2), \quad (16)$$

where $\{\mu_k\}$ is a standard step-size sequence in the subgradient method. It is interesting to observe that if \mathbf{x}^{k-1} is closer to $\{-1, 1\}^N$, which implies a smaller $N - \|\mathbf{x}^{k-1}\|^2$, then the increase of λ_k will be slowed down—which seems to make sense intuitively.

3.3 The Descent-Based Algorithm Used

We complete our design by specifying the descent-based algorithm in line 6 of Algorithm 1. The constraint of problem (8) has a simple structure, and one can exploit such structure for efficient optimization via structured optimization methods—such as the projected gradient method. Here we employ a more advanced version of the projected gradient method which was numerically found to be very fast in a different application [46].

The aforementioned method, called *gradient extrapolated majorization-minimization* (GEMM), entails concepts from majorization-minimization and the accelerated project gradient method. Here we concisely state the method, and the reader is referred to [46] for detailed descriptions such as its intuitions and stationary-point convergence guarantee. Let $\nabla f(\mathbf{x})$ denote the gradient of a differentiable function f at \mathbf{x} . Given a starting point \mathbf{x}^0 , the GEMM method iteratively runs

$$\mathbf{x}^{t+1} = \Pi(\mathbf{z}^t + \beta_t \nabla G_\lambda(\mathbf{z}^t | \mathbf{x}^t)), \quad t = 0, 1, 2, \dots \quad (17)$$

Here, Π denotes the projection onto $[-1, 1]^N$, which has a closed form $\Pi(\mathbf{x}) = [\Pi(x_1), \dots, \Pi(x_N)]^T$,

$$\Pi(x) = \begin{cases} -1, & x < -1 \\ x, & -1 \leq x \leq 1 \\ 1, & x > 1 \end{cases};$$

$G_\lambda(\cdot|\bar{\mathbf{x}})$ is a convex differentiable majorant of F_λ at $\bar{\mathbf{x}}$; $\nabla G_\lambda(\mathbf{x}|\bar{\mathbf{x}})$ is the gradient of $G_\lambda(\cdot|\bar{\mathbf{x}})$ at \mathbf{x} ; β_t is a step size; \mathbf{z}^t is an extrapolated point of \mathbf{x}^t and takes the form

$$\mathbf{z}^t = \mathbf{x}^t + \alpha_t(\mathbf{x}^t - \mathbf{x}^{t-1}) \quad (18)$$

for some pre-specified extrapolation sequence $\{\alpha_t\}$ (note that $\mathbf{x}^{-1} = \mathbf{x}^0$). Simply speaking, the GEMM method in (17) is an inexact majorization-minimization scheme that uses a one-iteration accelerated projected gradient update as its inexact update. It reduces to the projected gradient method if we set $\alpha_t = 0$ and replace $\nabla G_\lambda(\cdot|\mathbf{x}^t)$ with $\nabla F_\lambda(\cdot)$, i.e., no extrapolation and no majorization, respectively.

Let us examine the algorithmic details. We choose

$$G_\lambda(\mathbf{x}|\bar{\mathbf{x}}) = f(\mathbf{x}) - 2\lambda\langle\bar{\mathbf{x}}, \mathbf{x} - \bar{\mathbf{x}}\rangle - \lambda\|\bar{\mathbf{x}}\|^2 \quad (19)$$

as our convex differentiable majorant (note that $\|\mathbf{x}\|^2 \geq \|\bar{\mathbf{x}}\|^2 + 2\langle\bar{\mathbf{x}}, \mathbf{x} - \bar{\mathbf{x}}\rangle$, and f is convex differentiable). The gradient of the above majorant is

$$\nabla G_\lambda(\mathbf{x}|\bar{\mathbf{x}}) = -\sum_{i=1}^M \frac{1}{\Phi(\mathbf{g}_i^T \mathbf{x})} \frac{e^{-(\mathbf{g}_i^T \mathbf{x})^2/2}}{\sqrt{2\pi}} \mathbf{g}_i - 2\lambda\bar{\mathbf{x}} \quad (20)$$

where

$$\mathbf{g}_i = \frac{y_i}{\sigma} \mathbf{h}_i, \quad i = 1, \dots, M. \quad (21)$$

The step size β_t is chosen such that the sufficient descent condition

$$G_\lambda(\mathbf{x}^{t+1}|\mathbf{x}^t) \leq G_\lambda(\mathbf{z}^t|\mathbf{x}^t) + \langle\nabla G_\lambda(\mathbf{z}^t|\mathbf{x}^t), \mathbf{x}^{t+1} - \mathbf{z}^t\rangle + \frac{1}{2\beta_t}\|\mathbf{x}^{t+1} - \mathbf{z}^t\|^2 \quad (22)$$

is satisfied. We use the backtracking line search [56] to find such a β_t . The extrapolation sequence $\{\alpha_t\}$ is chosen as that of FISTA [56], given by

$$\alpha_t = \frac{\xi_{t-1} - 1}{\xi_t}, \quad \xi_t = \frac{1 + \sqrt{1 + 4\xi_{t-1}^2}}{2}, \quad \xi_{-1} = 1. \quad (23)$$

3.4 A Numerical Issue and Its Fix

We should mention a numerical issue with the gradient-based algorithm in the preceding subsection. In the gradient expression in (20), observe that the term $e^{-x^2/2}$ vanishes as $|x|$ is very large. We found that the occurrence of such vanishing instances can substantially slow down the convergence speed of the gradient-based algorithm, and it may even lead to numerical instability. As a practical trick, we get this around by over-estimating the noise variance, specifically, by replacing the original σ with $\sigma + \sigma_0$ for some fixed $\sigma_0 > 0$. We found that this trick works empirically. On the other hand, doing so means that we actually implement a mismatched ML detector

$$\min_{\mathbf{x} \in \{-1, 1\}^N} -\sum_{i=1}^M \log \Phi\left(\frac{y_i \mathbf{h}_i^T \mathbf{x}}{\hat{\sigma}}\right), \quad (24)$$

for some mismatched noise variance $\hat{\sigma}^2$. By our numerical experience, we do not see noticeable performance degradation with the mismatched ML detector. We justify this by the following fact.

Fact 2 Consider the same settings as in Fact 1, but with the objective function f replaced by that in (24). Suppose that the probability density function of \mathbf{h} is Gaussian with mean zero and covariance $\rho\mathbf{I}$ for some $\rho > 0$. Then, the minimizer of the “large- M ” objective function $\tilde{f}(\mathbf{x})$ in (6) over $\|\mathbf{x}\|^2 = N$ is uniquely given by the true binary vector. It follows that the mismatched ML problem

$$\min_{\mathbf{x} \in \{-1,1\}^N} \tilde{f}(\mathbf{x})$$

has its solution uniquely given by the true binary vector. The above results hold for any $\hat{\sigma}^2 > 0$.

Fact 2 is a corollary of Lemma 2 in [51], which considers the no-mismatch case $\hat{\sigma}^2 = \sigma^2$. Fact 2 suggests that, if the number of antennas at the BS is very large, the ML detector may be insensitive to the mismatch of the noise variance.

Proof of Fact 2: First, we recall a basic result in stochastic orders [57]. Let ξ, v be two continuous random variables, and let $\phi : \mathbb{R} \rightarrow \mathbb{R}$ be a strictly increasing function. If

$$\text{Prob}\{\xi \geq t\} \leq \text{Prob}\{v \geq t\}, \quad \text{for all } t, \quad (25)$$

ξ and v have unequal probability distributions, then we have $\mathbb{E}[\phi(\xi)] > \mathbb{E}[\phi(v)]$.

Second, we apply the above result to the problem at hand. Re-denote the true binary vector in the signal model (1) as \mathbf{x}_0 . Let $\mathbf{x} \in \mathbb{R}^N$ be any vector such that $\|\mathbf{x}\|^2 = N$ and $\mathbf{x} \neq \mathbf{x}_0$, and let

$$\xi = \mathbf{h}^T \mathbf{x}_0, \quad v = \mathbf{h}^T \mathbf{x}, \quad \phi(t) = \log \Phi(t/\hat{\sigma}).$$

Note that ϕ is strictly increasing. It was shown in Lemma 2 in [51] that (25) holds, and ξ and v have unequal probability distributions. Consequently we have

$$-\tilde{f}(\mathbf{x}_0) = \mathbb{E}[\phi(\mathbf{h}^T \mathbf{x}_0)] > \mathbb{E}[\phi(\mathbf{h}^T \mathbf{x})] = -\tilde{f}(\mathbf{x}),$$

and the minimum of $\tilde{f}(\mathbf{x})$ over $\|\mathbf{x}\|^2 = N$ is attained at, and only at, \mathbf{x}_0 . The proof is complete. ■

4 Making the Homotopy Algorithm a Deep Learnt One

As mentioned in the Introduction, lately there has been growing interest in deep learning for MIMO detection via the deep unfolding approach [28, 29, 37]. To employ deep unfolding, one first needs to start with an existing algorithm that is iterative by nature. The deep unfolding approach sees each iteration of the algorithm as a network layer, and it seeks to find a better network by untying some of the existing algorithm’s parameters and learning those parameters from data. Additionally, or alternatively, one may make the structure of each iteration more general, with the new structure controlled by some parameters; and then we learn those parameters from data.

The homotopy algorithm in the preceding section is very suitable for deep unfolding. To put into context, we first note that the homotopy algorithm in Algorithm 1, with the descent-based algorithm given by the one in Section 3.3, contains two loops—one for the update of the penalty parameter λ_k , another for the iterative process of the descent-based algorithm. But we can unfold the two loops into one and write

$$\mathbf{x}^{k+1} = \Pi(\mathbf{z}^k + \beta_k \nabla G_{\lambda_k}(\mathbf{z}_k | \mathbf{x}^k)), \quad (26)$$

for some appropriate $\alpha_k, \beta_k, \lambda_k$; cf. Algorithm 1 and (17). By putting the derivation of $\nabla G_\lambda(\mathbf{x}|\bar{\mathbf{x}})$ in (20) into (26), and recalling the expression of \mathbf{z}^k , we can represent the whole algorithm by

$$\mathbf{x}^{k+1} = \Pi(\mathbf{z}^k + \beta_k \mathbf{G}^T \mathbf{u}^k + 2\beta_k \lambda_k \mathbf{x}^k), \quad (27a)$$

$$\mathbf{u}^k = \Psi(\mathbf{G} \mathbf{z}^k), \quad (27b)$$

$$\mathbf{z}^k = \mathbf{x}^k + \alpha_k (\mathbf{x}^k - \mathbf{x}^{k-1}), \quad (27c)$$

where $\mathbf{G} = [\mathbf{g}_1, \dots, \mathbf{g}_M]^T$; \mathbf{g}_i is defined in (21); $\Psi(\mathbf{x}) = [\Psi(x_1), \dots, \Psi(x_N)]^T$,

$$\Psi(x) = \frac{1}{\Phi(x)} \frac{1}{\sqrt{2\pi}} e^{-x^2/2}.$$

As mentioned, deep unfolding sees each iteration in (27) as a network layer, and the whole iterative process a deep network. Fig. 4(a) illustrates (27) in a network form. We see that Π and Ψ appear as nonlinear activation functions; \mathbf{G} , $\alpha_k, \beta_k, \lambda_k$ serve as weights.

We apply deep unfolding to (27) by untying $\alpha_k, \beta_k, \lambda_k$, which are the extrapolation coefficient, step size, and penalty parameter, respectively. This means that we want to learn, from data, how the penalty parameters are adjusted. We should recall that the challenge with the homotopy method lies in the selection of the penalty parameters, and now we use data-driven learning to tackle the challenge. Similarly, we use data-driven learning to decide the step sizes and extrapolation. In addition, we make the structure more general by adding element-wise weights and biases in (27b). The network structure arising from the aforementioned untying and modification is given by

$$\mathbf{x}^{k+1} = \Pi(\mathbf{z}^k + \beta_k \mathbf{G}^T \mathbf{u}^k + \gamma_k \mathbf{x}^k), \quad (28a)$$

$$\mathbf{u}^k = \Psi(\mathbf{w}_k \odot (\mathbf{G} \mathbf{z}^k) + \mathbf{b}_k), \quad (28b)$$

$$\mathbf{z}^k = \mathbf{x}^k + \alpha_k (\mathbf{x}^k - \mathbf{x}^{k-1}), \quad (28c)$$

where γ_k is the untied parameter of $2\beta_k \lambda_k$; $\mathbf{w}_k, \mathbf{b}_k \in \mathbb{R}^M$ are a weight and bias, respectively; \odot denotes the element-wise, or Hadamard, product. Fig. 4(b) illustrates the network structure of (28). The set of parameters to learn at each layer is $\boldsymbol{\theta}_k \triangleq \{\alpha_k, \beta_k, \gamma_k, \mathbf{w}_k, \mathbf{b}_k\}$.

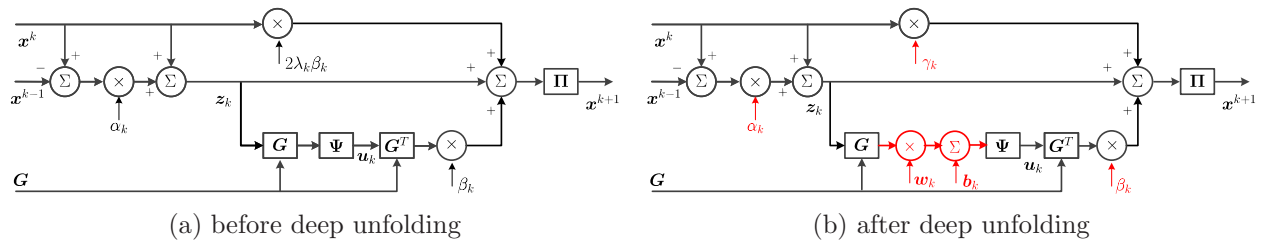


Figure 4: Illustration of the network structure.

In the above deep-unfolded homotopy algorithm, we do not give it a starting point \mathbf{x}^0 . Instead we learn a starting point. We generate \mathbf{x}^0 by

$$\mathbf{x}^0 = \Pi(\mathbf{W}_0 \mathbf{y} + \mathbf{b}_0), \quad (29)$$

where $\mathbf{W}_0 \in \mathbb{R}^{N \times M}$ is a weight matrix and $\mathbf{b}_0 \in \mathbb{R}^N$ is a bias; both are to be learnt. We let $\boldsymbol{\theta}_0 = \{\mathbf{W}_0, \mathbf{b}_0\}$.

Our training process is rather standard. The generative model $\mathbf{y} = \text{sgn}(\mathbf{H}\mathbf{x} + \mathbf{v})$ in (1), or more precisely, the complex model (3) and the subsequent complex-to-real conversion (4), is used to generate a large number of training samples $(\mathbf{x}_i, \mathbf{H}_i, \mathbf{y}_i, \sigma_i)_{i=1}^R$; note that R is the number of samples, the \mathbf{x}_i 's are uniform i.i.d. generated, and the \mathbf{H}_i 's are generated from a statistical channel model such as the Rayleigh. The network parameters $\boldsymbol{\theta}$'s are obtained by applying stochastic gradient descent to the empirical risk

$$\sum_{i=1}^R \|\mathbf{x}_i - \varphi(\mathbf{y}_i, \mathbf{H}_i, \sigma_i; \boldsymbol{\Theta})\|^2, \quad (30)$$

where the function $\varphi(\mathbf{y}, \mathbf{H}, \sigma; \boldsymbol{\Theta})$ represents the input-output relationship of the deep network, parameterized by $\boldsymbol{\Theta} = (\boldsymbol{\theta}_k)_{k=0}^K$; K is the number of layers.

Remark 2 We should also describe the big-O complexity of the deep-unfolded homotopy algorithm in (28). In addition to the usual operations of floating-point $+$, $-$, \times , $/$, \log , \exp , etc., the algorithm also requires us to compute Φ ; recall that $\Phi(t) = \int_{-\infty}^t \frac{1}{\sqrt{2\pi}} e^{-\tau^2/2} d\tau$ is the cumulative distribution of a standard Gaussian distribution. The computation of Φ takes definitely more than one floating-point operations, although there exist highly specialized algorithms (e.g., `erfc` in MATLAB) for efficient computation of Φ . In real-world implementations, one may build a dedicated operation for Φ to efficiently implement the algorithm in (28). It can be verified that the per-iteration complexity of the algorithm in (28) is $\mathcal{O}(M)$ for calling Φ , and $\mathcal{O}(MN)$ for the usual operations. Moreover, the non-deep counterpart of (28) has the same big-O per-iteration complexity.

5 Variation for Classical MIMO Detection

The preceding concepts for one-bit MIMO detection can also be applied to classical MIMO detection. The latter considers the unquantized model

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{v},$$

and the corresponding ML detector

$$\hat{\mathbf{x}}_{\text{ML}} = \arg \min_{\mathbf{x} \in \{-1, 1\}^N} \frac{1}{2} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2. \quad (31)$$

The homotopy algorithm in Section 3 directly applies by changing f to $\frac{1}{2} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2$. Recalling the algorithm representation in (26) and (28c), the homotopy algorithm for the classical MIMO case has the form

$$\mathbf{x}^{k+1} = \Pi(\mathbf{z}^k - \beta_k \mathbf{H}^T \mathbf{H} \mathbf{z}^k + \beta_k \mathbf{H}^T \mathbf{y} + 2\beta_k \lambda_k \mathbf{x}^k), \quad (32a)$$

$$\mathbf{z}^k = \mathbf{x}^k + \alpha_k (\mathbf{x}^k - \mathbf{x}^{k-1}); \quad (32b)$$

(we obtain (32a) by applying $\nabla f(\mathbf{x}) = \mathbf{H}^T \mathbf{H} \mathbf{x} - \mathbf{H}^T \mathbf{y}$ to (19) and (26)). The step size β_k can be chosen as $\beta_k = 1/\|\mathbf{H}\|_2^2$ where $\|\cdot\|_2$ denotes the spectral norm; this is a standard method for fulfilling the sufficient descent condition (22) when f is convex quadratic [56]. For the deep-unfolded homotopy algorithm, we modify (32) as

$$\mathbf{x}^{k+1} = \Pi(\mathbf{z}^k - \beta_k \mathbf{H}^T \mathbf{H} \mathbf{z}^k + \omega_k \mathbf{H}^T \mathbf{y} + \gamma_k \mathbf{x}^k), \quad (33a)$$

$$\mathbf{z}^k = \mathbf{x}^k + \alpha_k (\mathbf{x}^k - \mathbf{x}^{k-1}), \quad (33b)$$

where $\alpha_k, \beta_k, \omega_k, \gamma_k$ are the parameters to learn. Note that we train only four parameters at each layer (without counting the zeroth layer in (29)). The remaining details are essentially identical. The per-iteration complexity of the algorithm in (33) is $\mathcal{O}(N^2)$.

6 Simulation Results

Table 1: Summary of tested algorithms.

name	algorithm, reference	1-bit or classical	formulation, approach
HOTML	homotopy algorithm in Sections 3 and 5	both	ML, homotopy optimization
DeepHOTML	deep-unfolded homotopy algorithm in Sections 4 and 5	both	ML, deep unfolding of homotopy opt.
ZF	zero-forcing detector	both	linear
nML	near-ML detector [16]	1-bit	ML, convex relaxation
nML, two-stage	near-ML detector, then local exhaustive search [16]	1-bit	ML, convex relaxation, local search
GAMP	generalized approximate message passing [18, 19]	1-bit	Bayesian, approximate message passing
MMSE DF	minimum-mean-square-error decision-feedback detector	classical	linear, decision feedback
LRA MMSE DF	lattice reduction-aided MMSE DF detector [8]	classical	linear, lattice reduction
SD	sphere decoder, Schnorr-Euchner implementation [1]	classical	ML, exact branch-and-bound search
SDR	semidefinite relaxation, implemented by the row-by-row method [58]	classical	ML, convex relaxation
DetNet	detection network [29]	classical	ML, deep unfolding
LAMA	large MIMO approximate message passing [12], damping [59]	classical	Bayesian, approximate message passing

In this section we show an extensive collection of numerical results to examine the performance of the homotopy algorithm and its deep-unfolded adaptation proposed in the preceding sections, and to provide benchmarking with the state-of-the-art algorithms.

6.1 Simulation and Algorithm Settings

We first describe how the simulations were prepared. The procedure is standard: We randomly generate the signal according to the complex-valued MIMO model (3) for the one-bit MIMO case, and its unquantized counterpart $\mathbf{y}_C = \mathbf{H}_C \mathbf{x}_C + \mathbf{v}_C$ for the classical MIMO case. Then, we apply the complex-to-real conversion in (4) to obtain an instance of $(\mathbf{y}, \mathbf{H}, \mathbf{x})$. Unless otherwise specified, the complex-valued channel \mathbf{H}_C is generated following the element-wise i.i.d. circular Gaussian distribution with mean zero and unit variance (i.e., the Rayleigh channel). The transmit vector \mathbf{x}_C is generated following the element-wise i.i.d. uniform distribution on the QPSK constellation set

$\{\pm 1 \pm j\}$. We define the SNR as

$$\text{SNR} = \mathbb{E}[\|\mathbf{H}_C \mathbf{x}_C\|^2] / \mathbb{E}[\|\mathbf{v}_C\|^2],$$

and we use the above formula to determine the noise variance σ_C^2 (or its real counterpart $\sigma^2 = \sigma_C^2/2$) when the SNR is given. We use 100,000 independently generated instances to evaluate the bit error rates (BERs) of the algorithms under test. We also evaluate the computational complexities by numerically counting the floating point operations (FLOPs) and also by measuring the actual runtimes on our computer. For fairness, the evaluations were done on the same platform and on the same computer; specifically, MATLAB 8.5, and a desktop computer with Intel i7-4770 processor and 16GB memory.

Next, we provide the implementation details with our proposed algorithms. For convenience, we name the homotopy algorithm in Section 3 *Homotopy Optimization ML (HOTML)*, and the deep-unfolded homotopy algorithm in Section 4 *DeepHOTML*; the same convention applies to their classical MIMO counterparts in Section 5. Let us first consider HOTML based on the representation in Algorithm 1. We initialize the penalty parameter as $\lambda_0 = 0.01$. We randomly initialize the algorithm by uniformly generating \mathbf{x}^0 on $[-1, 1]^N$. The update rule of λ_k is (16), with $\mu_k = 0.1/k$ for the one-bit MIMO case and $\mu_k = 1/k$ for the classical MIMO case (a standard step-size rule in subgradient methods). We stop Algorithm 1 when $|\lambda_k - \lambda_{k-1}| \leq 10^{-4}$. The descent-based algorithm inside Algorithm 1, which is described in Section 3.3, is stopped when $\|\mathbf{x}^{t+1} - \mathbf{x}^t\| \leq 10^{-4}$ or when the number of iterations exceeds a limit; that limit is 300 for the one-bit MIMO case, and 100 for the classical MIMO case. For the remedy of the numerical issue discussed in Section 3.4, we set $\sigma_0 = 0.5$.

The implementation details with DeepHOTML, which are mostly with training, are as follows. Unless otherwise specified, the number of network layers is $K = 20$. We implement the training process by Tensorflow for Python [60]. The generation of training samples is exactly the same as the signal generation described above. We use the ADAM stochastic gradient descent optimizer [61], with exponentially decaying step sizes, for training: specifically, the initial step size is 0.001; the step size decays by 0.9 for the one-bit case, and 0.95 for the classical MIMO case, once every 500 training iterations; each training iteration contains a batch of 500 training samples; we train the network with 10,000 iterations. In addition, for the one-bit MIMO case, we do the following: every training sample has its SNR randomly drawn from a range of 5dB to 22dB, by uniform distribution; the network parameters \mathbf{w}_i 's and \mathbf{b}_i 's are initialized as i.i.d. Gaussian random vectors with mean zero and variance 0.01; we initialize $\beta_k = 0.01, \gamma_k = 0.001, \alpha_k = 0.5$ for all k . For the classical MIMO case the settings are similar, with the following differences: the SNR range is changed to 0dB to 18dB; we initialize $\beta_k = 0.01, \omega_k = -0.01, \gamma_k = 0.001, \alpha_k = 0.5$ for all k .

The algorithms we benchmarked should also be mentioned. Table 1 provides a summary of the tested algorithms. These algorithms either are some of the *de facto* standards in MIMO detection, or are emerging methods. Some additional implementation details should be mentioned. We modify nML [16] by changing its step-size rule to the backtracking line search, which is also used in HOTML for one-bit MIMO detection; we found that this modification improves the performance of nML. The number of iterations of GAMP is set to 20, as recommended in [18]. The number of iterations of the row-by-row method [58] for SDR is 200. The number of iterations of LAMA is 100, the same one used in [12]; also we stop LAMA if the difference of successive iterates is less than 10^{-4} . DetNet [29] has its source code available at <https://github.com/neevesamuel/LearningToDetect>. We use that open source code to train DetNet under the recommended settings of 30 network layers, 50,000 training iterations, and 3,000 training samples for each training iteration.

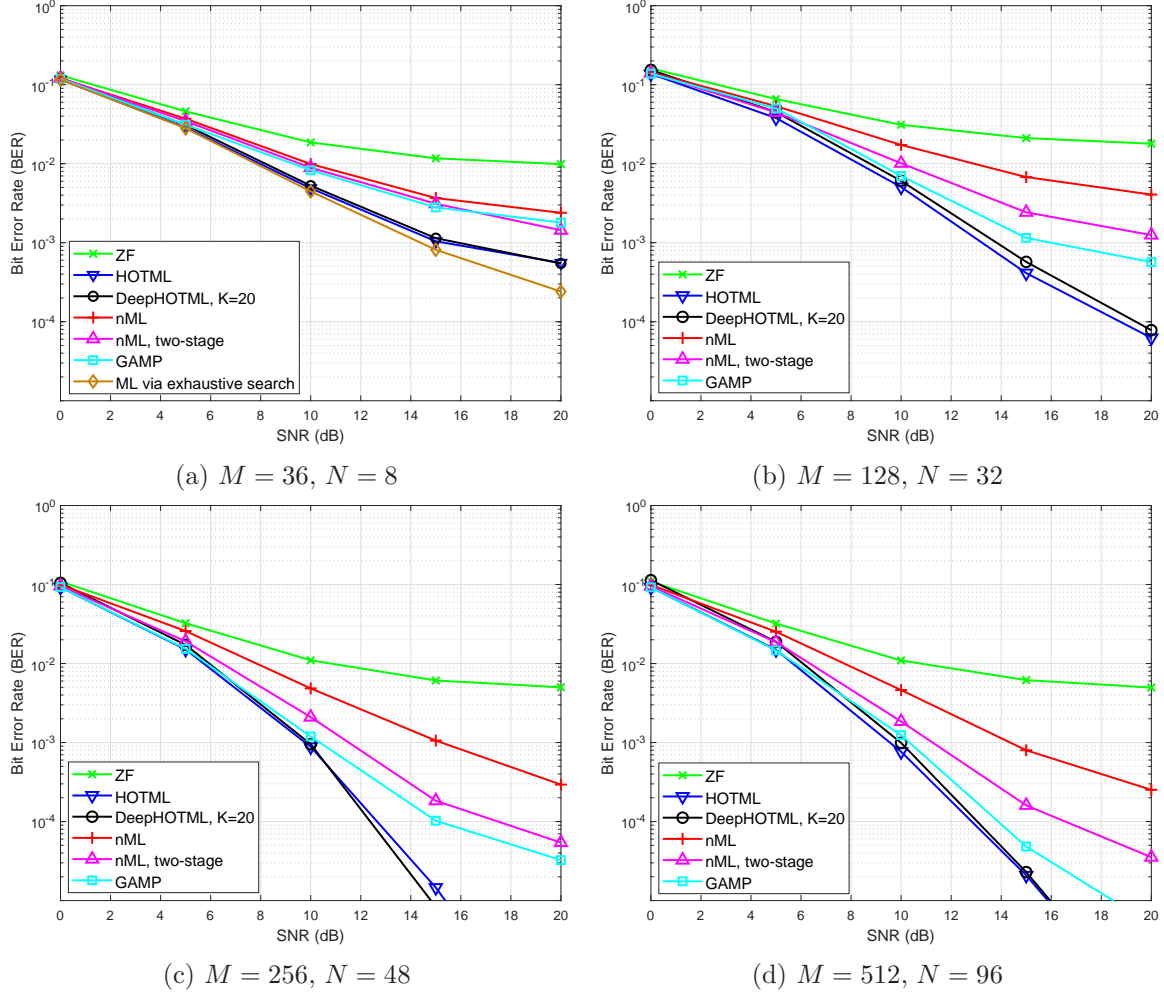


Figure 5: One-bit MIMO detection performance.

Table 2: Complexity comparison with the one-bit MIMO detection algorithms.

$M \times N$		HOTML	DeepHOTML	nML	GAMP
128×32	FLOPs	1.8862×10^7	3.5187×10^5	5.6806×10^5	3.9269×10^5
	no. of Φ calc.	1.4555×10^5	2.560×10^3	4.5408×10^3	2.560×10^3
	time (Sec.)	0.0589	0.0008	0.0027	0.0013
256×48	FLOPs	5.1287×10^7	1.03801×10^6	1.4811×10^6	1.1714×10^6
	no. of Φ calc.	2.6557×10^5	5.120×10^3	8.2565×10^3	5.120×10^3
	time (Sec.)	0.0936	0.0014	0.0053	0.0021
512×96	FLOPs	9.8408×10^7	4.0913×10^6	8.1409×10^6	4.2355×10^6
	no. of Φ calc.	3.8483×10^5	1.024×10^4	2.0065×10^4	1.024×10^4
	time (Sec.)	0.1308	0.0019	0.0088	0.0029

6.2 One-Bit MIMO Detection

In this subsection we examine the one-bit MIMO case. Fig. 5 shows the BERs of the tested algorithms under different problem sizes (M, N) . The benchmarked algorithms are the ZF detector, which is the direct application of zero forcing in classic MIMO detection; nML and its two-stage variant, which are based on sphere relaxation of the ML problem (2) (cf., problem (5)); and GAMP which is the application of approximate message passing to one-bit MIMO detection. For the small problem size case of $(M, N) = (36, 8)$ in Fig. 5(a), we also provide a performance baseline by evaluating the performance of the optimal ML detector via exhaustive search. We see from Fig. 5 that HOTML and DeepHOTML generally yield similar BER performance and achieve much better BER performance than all the other benchmarked algorithms except the exhaustive search. It is also observed that the performance of GAMP improves as the problem size increases.

Next, we compare the computational complexities of the tested algorithms. As mentioned in Remark 2, HOTML and DeepHOTML require calling the function $\Phi(t) = \int_{-\infty}^t \frac{1}{\sqrt{2\pi}} e^{-\tau^2/2} d\tau$. Hence we divide the operation counts into two, one as the FLOPs of the usual operations ($+$, $-$, \times , $/$, \log , \exp), the other as the number of times Φ is called. Note that nML and GAMP also require calling Φ ², and their operational counts are done by the same way as above.

The complexity results are shown in Table 2. We use the same settings as in Fig. 5, with the SNR fixed at 15dB. The fastest algorithm, as revealed by both the actual runtimes and operation counts, is DeepHOTML. GAMP and nML are the second and third best, respectively; HOTML is the slowest. It is worth noting that the computations of DeepHOTML are similar to those of a gradient descent method with a fixed number of iterations (or layers) of 20—this is why DeepHOTML is fast. The computations of HOTML are also similar to those of a gradient descent method. However, as suggested by Table 2, HOTML uses considerably more iterations than DeepHOTML; in fact, HOTML’s line search routine for determining the step sizes also costs a non-negligible amount of computations.

It is also interesting to examine the BER performance of DeepHOTML under different number of network layers. Fig. 6 shows the results; again, the settings are identical to those in Fig. 5. Note that when we change the number of layers K , we train DeepHOTML for that particular K . We observe from Fig. 6 is that the BER performance improves as the number of layers increases. Also, it is encouraging to see that DeepHOTML with 10 layers can achieve BER performance similar to that with 20 layers; DeepHOTML with 5 layers also looks good.

6.3 Classical MIMO Detection

Now we turn to the classical MIMO case. Fig. 7 shows the BERs of the tested algorithms for the relatively easier instances of overdetermined MIMO channels (i.e., $M > N$). In the figures, “lower bound” is the BER performance when there is no interference between symbols. Note that SD is an optimal ML algorithm using some smart branch-and-bound search method, but still it is computationally prohibitive when N is large. In our simulation, we only tested SD for $N = 40$. Some observations are as follows. LAMA suffers from error floor effects for the case of $(M, N) = (60, 40)$, but it achieves near-optimal performance when the problem size increases to $(M, N) = (120, 80)$. LAMA appears to be favorable for large MIMO regimes, and we will see the

²GAMP requires computing the expectation and variance of some posterior distribution, and this can be done by computing Φ [19].

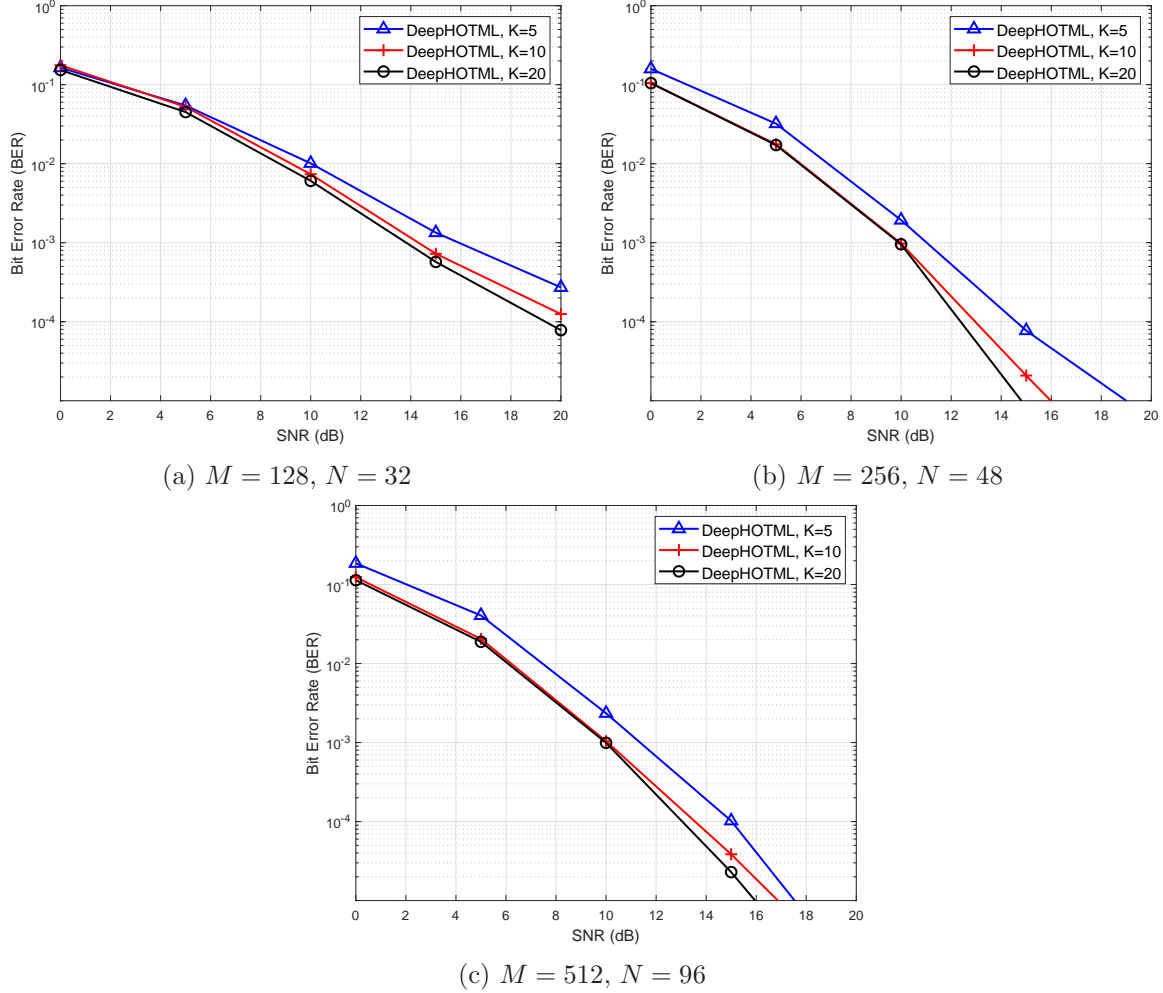


Figure 6: One-bit MIMO detection performance of DeepHOTML under different numbers of layers.

same behaviors later. Also, SDR, DetNet, HOTML and DeepHOTML all achieve near-optimal performance.

In the BER plots in Fig. 8, we examine the more challenging instances of critically determined channels (i.e., $M = N$). It is worth noting that DetNet, in its original paper [28,29], did not consider the critically determined cases. We tried our best to train DetNet in critically determined cases, but we were not too successful with obtaining satisfactory results as in the preceding overdetermined cases. In the case of $M = N = 400$, we were even unable to complete the training; we found that the training (implemented by Tensorflow) drew a very substantial amount of computational resources. Fortunately, for DeepHOTML, we did not encounter the same problem; this is likely because our DeepHOTML network has only 4 parameters per layer (without counting the 0th layer in (29)). Also, note that we did not run SDR for the case of $M = N = 400$; although SDR is known to have polynomial-time complexity, it is still too expensive to run SDR when the problem size is very large. We observe that DetNet does not perform well, LAMA performs well only when the problem sizes are large, and SDR, HOTML and DeepHOTML consistently show near-optimal performance; DeepHOTML performs slightly worse for the small size $M = N = 40$ case.

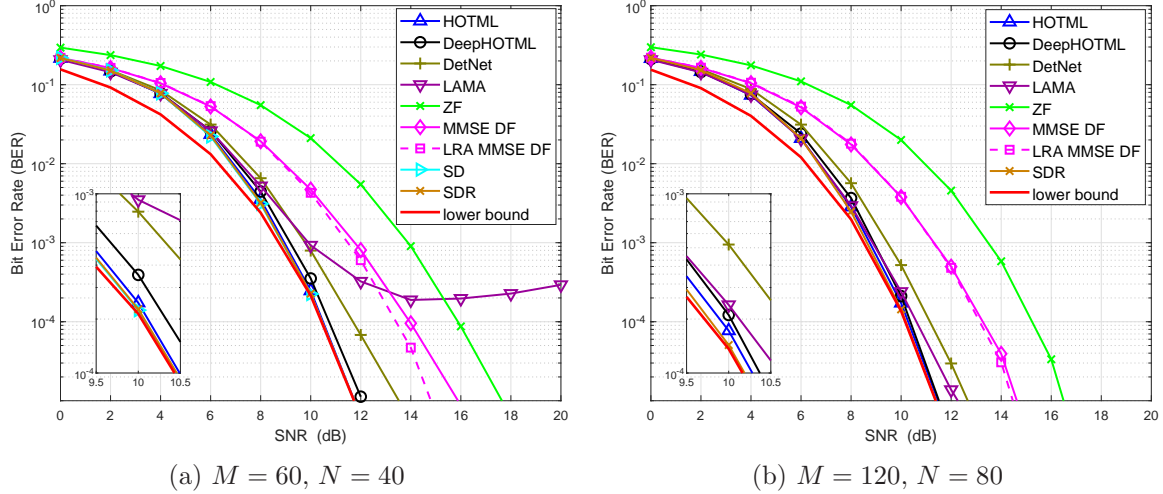


Figure 7: Classical MIMO detection performance for overdetermined channel cases.

In addition to the i.i.d. Gaussian channel simulations shown above, we are also interested in correlated MIMO channels, specifically,

$$\mathbf{H}_C = \mathbf{R}_r^{1/2} \tilde{\mathbf{H}} \mathbf{R}_t^{1/2},$$

where $\tilde{\mathbf{H}}$ is element-wise i.i.d. circular Gaussian with mean zero and unit variance; \mathbf{R}_r and \mathbf{R}_t are the spatial correlation matrices at the receiver and transmitter, respectively, and they are modeled as

$$[\mathbf{R}_r]_{i,j} = \begin{cases} r^{i-j}, & \text{if } i \leq j \\ [\mathbf{R}_r]_{i,j}^*, & \text{otherwise} \end{cases}$$

for $|r| \leq 1$ [62]. In the simulation below, we set $r = 0.2$. Also, DeepHOTML is trained under the SNR range of 18dB to 34dB. The other simulation settings are same as the i.i.d. Gaussian case above. It is worth noting that LAMA works poorly for correlated channels because it exploits the assumption of i.i.d. Gaussian channels [12]. To mend this issue we apply the recently proposed damping technique in approximate message passing [59] to LAMA; the damping rate is **damp** = 0.7. LAMA with damping leads to a slower convergence by our empirical experience (the per-iteration complexity is almost twice of that of the original LAMA), but the performance is improved significantly. Fig. 9 shows the BER results for different problem sizes. It is seen that both HOTML and DeepHOTML achieve near-optimal detection performance; SDR is only slightly worse; LAMA with damping works well in the large-scale case in Fig. 9(b), but suffers from error floor effects for the moderate size case in Fig. 9(a).

In Fig. 10 we compare the complexities of the tested algorithms by examining their FLOPs under various problem sizes N ; the SNR is fixed at 10dB. We see that DeepHOTML stands as the fastest algorithm in general.

We finish by showing the BER performance of DeepHOTML for different numbers of layers in Fig. 11. The results indicate that DeepHOTML can achieve promising performance with the number of layers as small as 10.

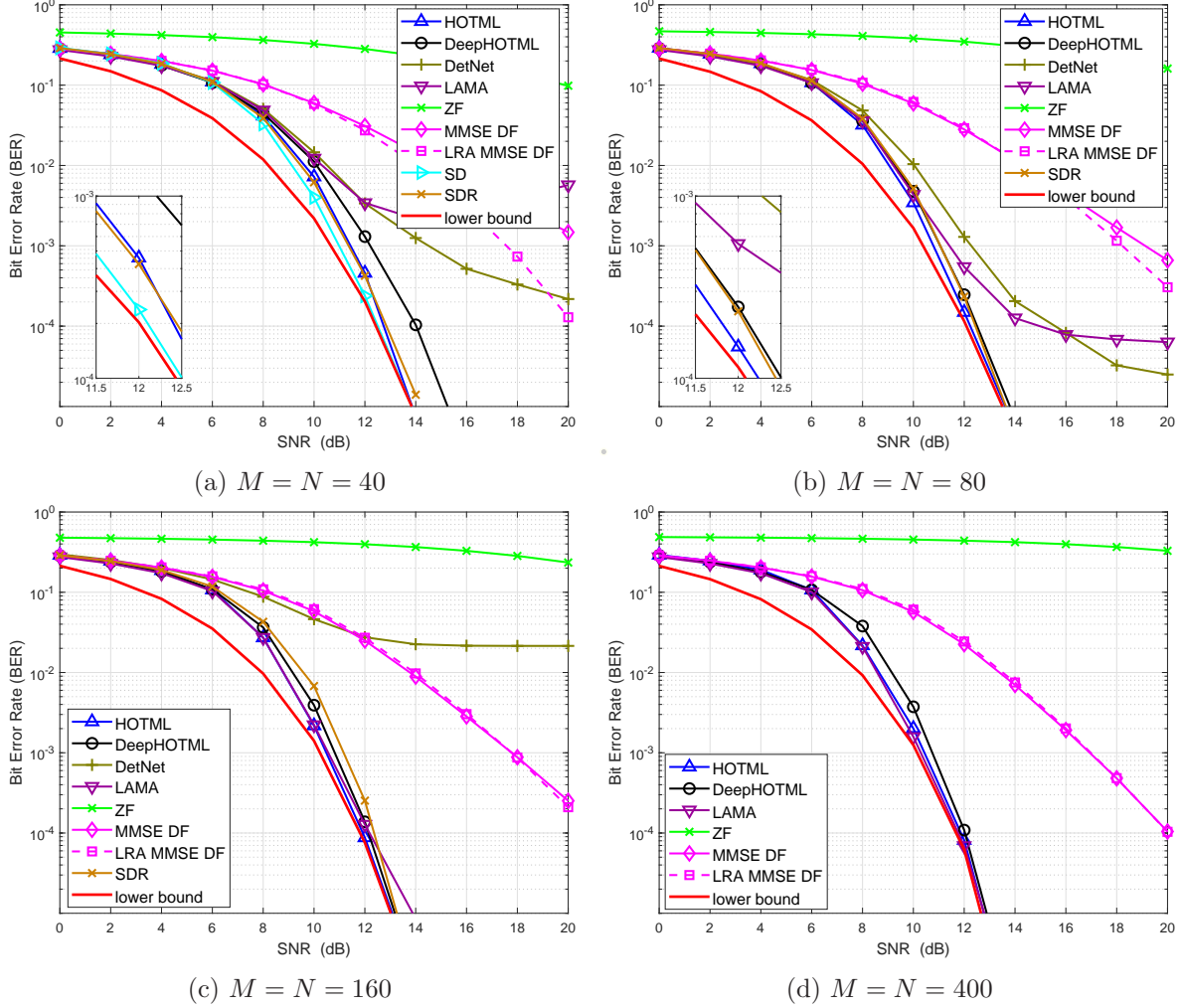


Figure 8: Classical MIMO detection performance for critically determined channel cases.

7 Conclusion

To conclude, we developed a homotopy optimization algorithm for efficient high-performance MIMO detection under one-bit quantized or unquantized observations. We also studied the possibility of using deep unfolding to enhance the performance of our homotopy algorithm. Our numerical results illustrated that the homotopy algorithms, non-deep and deep, achieve promising detection performance. Also, the deep homotopy algorithm was found to be easy and stable to train for a variety of MIMO settings, and its operational cost very competitive compared to those of many existing MIMO detection methods. Our present study focused on binary symbol constellations, and we hope that our endeavor would provide further insights into attacking other types of symbol constellations. In addition it will be interesting to study the extension to the frequency-selective fading scenario, which is much more challenging as revealed in studies such as [15].

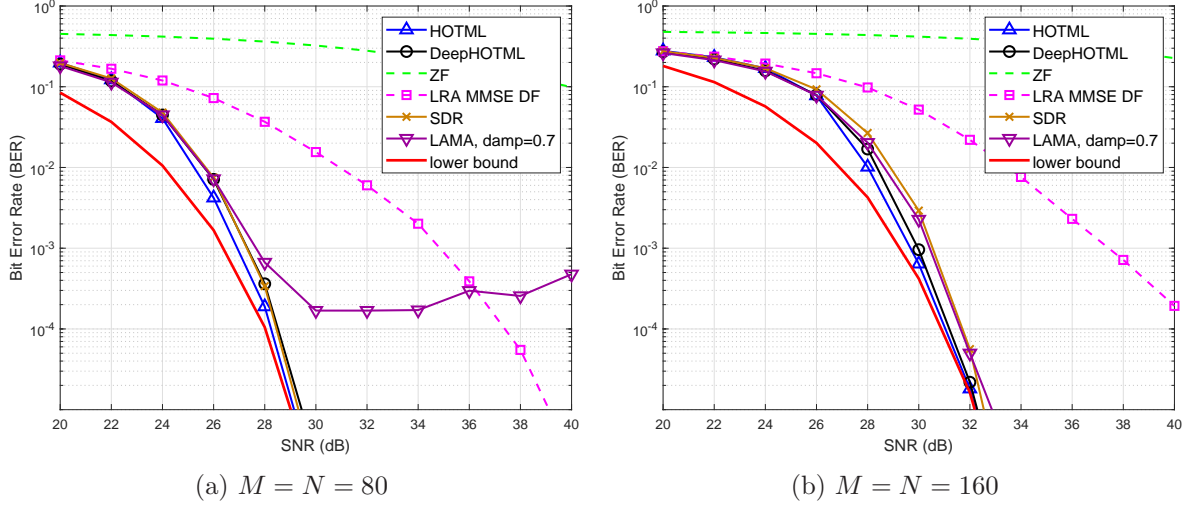


Figure 9: Classical MIMO detection performance for correlated channels.

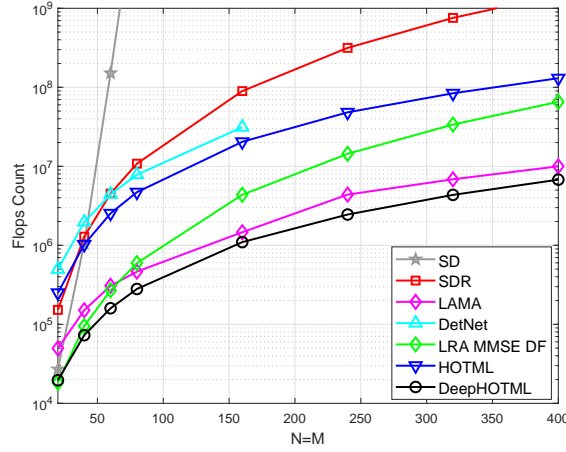


Figure 10: Complexity comparison with the classical MIMO detection algorithms.

References

- [1] M. O. Damen, H. El Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2389–2402, 2003.
- [2] J. Jaldén and B. Ottersten, "On the complexity of sphere decoding in digital communications," *IEEE Trans. Signal Process.*, vol. 53, no. 4, pp. 1474–1484, 2005.
- [3] P. H. Tan, L. K. Rasmussen, and T. J. Lim, "Constrained maximum-likelihood detection in CDMA," *IEEE Trans. Commun.*, vol. 49, no. 1, pp. 142–153, 2001.
- [4] A. Yener, R. D. Yates, and S. Ulukus, "CDMA multiuser detection: A nonlinear programming approach," *IEEE Trans. Commun.*, vol. 50, no. 6, pp. 1016–1024, 2002.

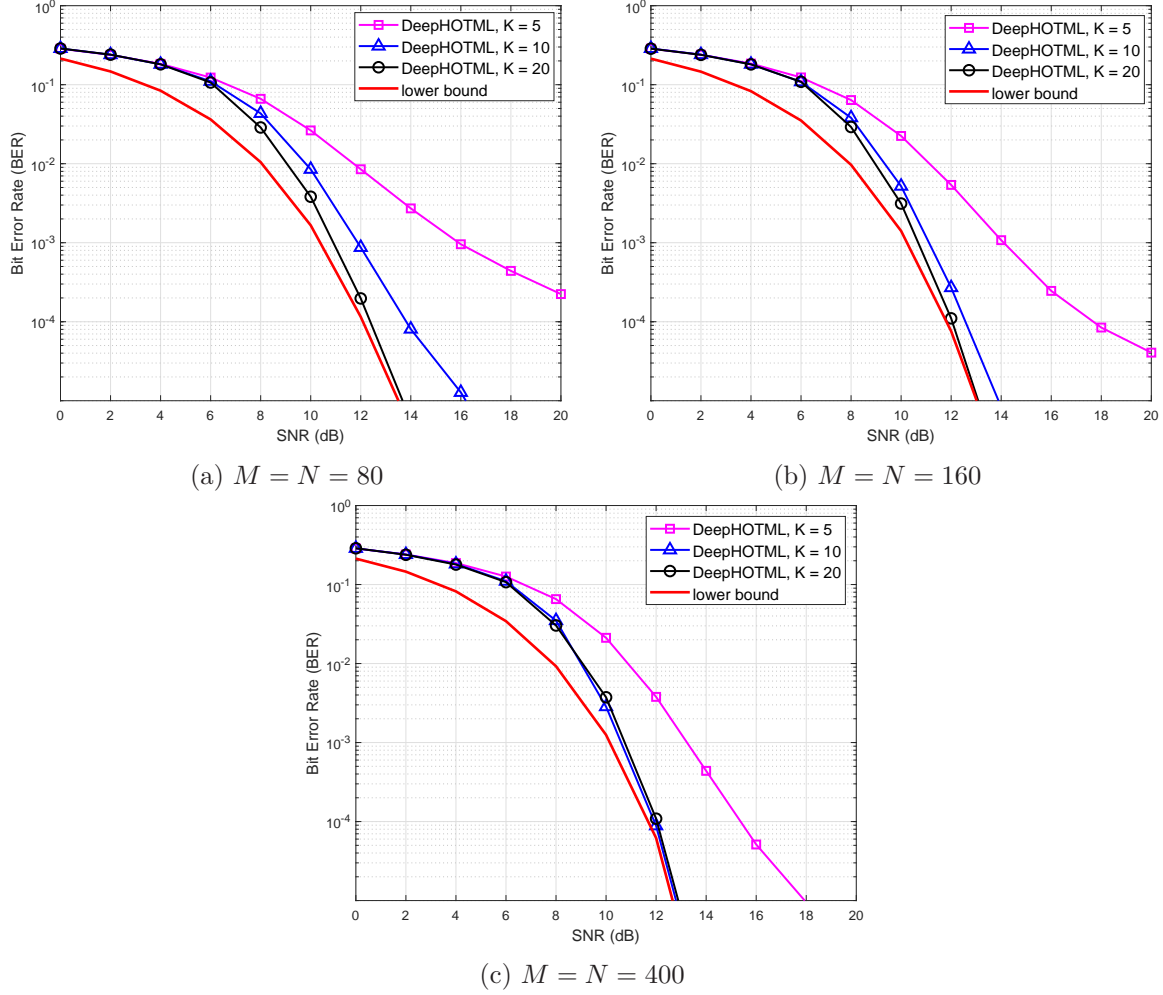


Figure 11: Classical MIMO detection performance of DeepHOTML under different numbers of layers.

- [5] P. H. Tan and L. K. Rasmussen, "The application of semidefinite programming for detection in CDMA," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 8, pp. 1442–1449, 2001.
- [6] W.-K. Ma, T. N. Davidson, K. M. Wong, Z.-Q. Luo, and P.-C. Ching, "Quasi-maximum-likelihood multiuser detection using semi-definite relaxation with application to synchronous CDMA," *IEEE Trans. Signal Process.*, vol. 50, no. 4, pp. 912–922, 2002.
- [7] C. Thrampoulidis, W. Xu, and B. Hassibi, "Symbol error rate performance of box-relaxation decoders in massive MIMO," *IEEE Trans. Signal Process.*, vol. 66, no. 13, pp. 3377–3392, 2018.
- [8] D. Wubben, D. Seethaler, J. Jalden, and G. Matz, "Lattice reduction," *IEEE Signal Process. Mag.*, vol. 28, no. 3, pp. 70–91, 2011.

- [9] J. Luo, K. R. Pattipati, P. K. Willett, and F. Hasegawa, "Near-optimal multiuser detection in synchronous CDMA using probabilistic data association," *IEEE Commun. Lett.*, vol. 5, no. 9, pp. 361–363, 2001.
- [10] P. Fertl, J. Jaldén, and G. Matz, "Performance assessment of MIMO-BICM demodulators based on mutual information," *IEEE Trans. Signal Process.*, vol. 60, no. 3, pp. 1366–1382, 2011.
- [11] A. Maleki, L. Anitori, Z. Yang, and R. G. Baraniuk, "Asymptotic analysis of complex LASSO via complex approximate message passing (CAMP)," *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4290–4308, 2013.
- [12] C. Jeon, R. Ghods, A. Maleki, and C. Studer, "Optimal data detection in large MIMO," *arXiv:1811.01917*, 2018.
- [13] S. Yang and L. Hanzo, "Fifty years of MIMO detection: The road to large-scale MIMOs," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 1941–1988, 2015.
- [14] C. Risi, D. Persson, and E. G. Larsson, "Massive MIMO with 1-bit ADC," *arXiv:1404.7736*, 2014.
- [15] C. Studer and G. Durisi, "Quantized massive MU-MIMO-OFDM uplink," *IEEE Trans. Commun.*, vol. 64, no. 6, pp. 2387–2399, 2016.
- [16] J. Choi, J. Mo, and R. W. Heath, "Near maximum-likelihood detector and channel estimator for uplink multiuser massive MIMO systems with one-bit ADCs," *IEEE Trans. Commun.*, vol. 64, no. 5, pp. 2005–2018, 2016.
- [17] Y.-S. Jeon, N. Lee, S.-N. Hong, and R. W. Heath, "One-bit sphere decoding for uplink massive MIMO systems with one-bit ADCs," *IEEE Trans. Wireless Commun.*, vol. 17, no. 7, pp. 4509–4521, 2018.
- [18] S. Wang, Y. Li, and J. Wang, "Multiuser detection for uplink large-scale MIMO under one-bit quantization," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2014, pp. 4460–4465.
- [19] C.-K. Wen, C.-J. Wang, S. Jin, K.-K. Wong, and P. Ting, "Bayes-optimal joint channel-and-data estimation for massive MIMO with low-precision ADCs," *IEEE Trans. Signal Process.*, vol. 64, no. 10, pp. 2541–2556, 2015.
- [20] S.-N. Hong, S. Kim, and N. Lee, "A weighted minimum distance decoding for uplink multiuser MIMO systems with low-resolution ADCs," *IEEE Trans. Commun.*, vol. 66, no. 5, pp. 1912–1924, 2017.
- [21] Y. Cho and S.-N. Hong, "One-bit successive-cancellation soft-output (oss) detector for uplink MU-MIMO systems with one-bit ADCs," *IEEE Access*, vol. 7, pp. 27 172–27 182, 2019.
- [22] D. Kim, S.-N. Hong, and N. Lee, "Supervised-learning for multi-hop MU-MIMO communications with one-bit transceivers," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 11, pp. 2559–2572, 2019.

- [23] D. Plabst, J. Munir, A. Mezghani, and J. A. Nossek, "Efficient non-linear equalization for 1-bit quantized cyclic prefix-free massive MIMO systems," in *Proc. IEEE 15th Int. Symposium Wireless Commun. Syst. (ISWCS)*, 2018.
- [24] S. H. Mirfarshbafan, M. Shabany, S. A. Nezamalhoseini, and C. Studer, "Algorithm and VLSI design for 1-bit data detection in massive MIMO-OFDM," *IEEE Open J. Circuits Syst.*, vol. 1, pp. 170–184, 2020.
- [25] M. Shao and W.-K. Ma, "Divide and conquer: One-bit MIMO-OFDM detection by inexact expectation maximization," to appear in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Process. (ICASSP)*, 2021.
- [26] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cognitive Commun. Network.*, vol. 3, no. 4, pp. 563–575, 2017.
- [27] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for interference management," *IEEE Trans. Signal Process.*, vol. 66, no. 20, pp. 5438–5453, 2018.
- [28] N. Samuel, T. Diskin, and A. Wiesel, "Deep MIMO detection," in *Proc. 2017 IEEE Int. Workshop Signal Process. Advances Wireless Commun. (SPAWC)*, 2017.
- [29] N. Samuel, T. Diskin, and A. Wiesel, "Learning to detect," *IEEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2554–2564, May 2019.
- [30] S. Takabe, M. Imanishi, T. Wadayama, R. Hayakawa, and K. Hayashi, "Trainable projected gradient detector for massive overloaded MIMO channels: Data-driven tuning approach," *IEEE Access*, vol. 7, pp. 93 326–93 338, 2019.
- [31] V. Corlay, J. J. Boutros, P. Ciblat, and L. Brunel, "Multilevel MIMO detection with deep learning," in *Proc. 52nd Asilomar Conf. Signals, Systems, Comput.*, 2018, pp. 1805–1809.
- [32] X. Tan, W. Xu, K. Sun, Y. Xu, Y. Be'ery, X. You, and C. Zhang, "Improving massive MIMO message passing detectors with deep neural network," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1267–1280, 2020.
- [33] H. He, C. Wen, S. Jin, and G. Y. Li, "Model-driven deep learning for MIMO detection," *IEEE Trans. Signal Process.*, vol. 68, pp. 1702–1715, 2020.
- [34] M. Mohammadkarimi, M. Mehrabi, M. Ardakani, and Y. Jing, "Deep learning-based sphere decoding," *IEEE Trans. Wireless Commun.*, vol. 18, no. 9, pp. 4368–4378, 2019.
- [35] G. J. Gibson, S. Siu, and C. Cowen, "Multilayer perceptron structures applied to adaptive equalisers for data communications," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process. (ICASSP)*, 1989, pp. 1183–1186.
- [36] S. Chen, B. Mulgrew, and P. M. Grant, "A clustering technique for digital communications channel equalization using radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 4, no. 4, pp. 570–590, 1993.

- [37] K. Gregor and Y. LeCun, “Learning fast approximations of sparse coding,” in *Proc. Int. Conf. Machine Learning*, 2010, pp. 399–406.
- [38] P. Sprechmann, A. M. Bronstein, and G. Sapiro, “Learning efficient sparse and low rank models,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1821–1833, 2015.
- [39] X. Chen, J. Liu, Z. Wang, and W. Yin, “Theoretical linear convergence of unfolded ista and its practical weights and thresholds,” in *Proc. Advances Neural Inf. Process. Systems*, 2018, pp. 9061–9071.
- [40] Z. Wu, “The effective energy transformation scheme as a special continuation approach to global optimization with application to molecular conformation,” *SIAM J. Opt.*, vol. 6, no. 3, pp. 748–768, 1996.
- [41] D. M. Dunlavy and D. P. O’Leary, “Homotopy optimization methods for global optimization,” *Report SAND2005-7495*, Sandia National Laboratories, 2005.
- [42] L. Xiao and T. Zhang, “A proximal-gradient homotopy method for the sparse least-squares problem,” *SIAM J. Optim.*, vol. 23, no. 2, pp. 1062–1091, 2013.
- [43] H. Mobahi and J. W. Fisher, “On the link between gaussian homotopy continuation and convex envelopes,” in *Proc. Int. Workshop Energy Minimization Methods in Computer Vision and Pattern Recognit.* Springer, 2015, pp. 43–56.
- [44] E. Hazan, K. Y. Levy, and S. Shalev-Shwartz, “On graduated optimization for stochastic non-convex problems,” in *Proc. Int. Conf. Machine Learning*, vol. 48, 2016, pp. 1833–1841.
- [45] A. Anandkumar, Y. Deng, R. Ge, and H. Mobahi, “Homotopy analysis for tensor PCA,” in *Proc. Machine Learning Research*, vol. 65, 2017, pp. 79–104.
- [46] M. Shao, Q. Li, W.-K. Ma, and A. M.-C. So, “A framework for one-bit and constant-envelope precoding over multiuser massive MISO channels,” *IEEE Trans. Signal Process.*, vol. 67, no. 20, pp. 5309–5324, 2019.
- [47] S. Poljak, F. Rendl, and H. Wolkowicz, “A recipe for semidefinite relaxation for $(0, 1)$ -quadratic programming,” *J. Global Optim.*, vol. 7, no. 1, pp. 51–73, 1995.
- [48] J. Pan, W.-K. Ma, and J. Jalden, “MIMO detection by Lagrangian dual maximum-likelihood relaxation: Reinterpreting regularized lattice decoding,” *IEEE Trans. Signal Process.*, vol. 62, no. 2, pp. 511–524, 2013.
- [49] S. D. Blunt and K. Ho, “An iterative approximate MAP symbol estimator for uncoded synchronous CDMA,” *IEEE Trans. Wireless Commun.*, vol. 4, no. 4, pp. 1663–1673, 2005.
- [50] H. Liu, M.-C. Yue, A. M.-C. So, and W.-K. Ma, “A discrete first-order method for large-scale MIMO detection with provable guarantees,” in *Proc. 2017 IEEE Int. Workshop Signal Process. Advances Wireless Commun. (SPAWC)*, 2017.
- [51] J. Choi, D. J. Love, D. R. Brown, and M. Boutin, “Quantized distributed reception for MIMO wireless systems using spatial multiplexing,” *IEEE Trans. Signal Process.*, vol. 63, no. 13, pp. 3537–3548, 2015.

- [52] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
- [53] X. Liu and S. C. Draper, “The ADMM penalized decoder for LDPC codes,” *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 2966–2984, June 2016.
- [54] O. Castañeda, S. Jacobsson, G. Durisi, M. Coldrey, T. Goldstein, and C. Studer, “1-bit massive MU-MIMO precoding in VLSI,” *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 7, no. 4, pp. 508–522, 2017.
- [55] S. Boyd, L. Xiao, and A. Mutapcic, *Subgradient Methods*. Lecture Notes of EE392o, Stanford University, Autumn Quarter 2003–2004.
- [56] A. Beck, *First-Order Methods in Optimization*. Philadelphia, PA, USA: SIAM, 2017, vol. 25.
- [57] M. Shaked and J. G. Shanthikumar, *Stochastic Orders*. Springer Science & Business Media, 2007.
- [58] H.-T. Wai, W.-K. Ma, and A. M.-C. So, “Cheap semidefinite relaxation MIMO detection using row-by-row block coordinate descent,” in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Process. (ICASSP)*, 2011, pp. 3256–3259.
- [59] S. Rangan, P. Schniter, A. K. Fletcher, and S. Sarkar, “On the convergence of approximate message passing with arbitrary matrices,” *IEEE Trans. Inf. Theory*, vol. 65, no. 9, pp. 5339–5351, 2019.
- [60] M. Abadi *et al.*, “Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” *arXiv:1603.04467*, 2016.
- [61] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. 3rd Int. Conf. Learn. Representations*, 2014.
- [62] S. L. Loyka, “Channel capacity of MIMO architecture using the exponential correlation matrix,” *IEEE Commun. Lett.*, vol. 5, no. 9, pp. 369–371, 2001.