YAMPA: Yet Another Matching Pursuit Algorithm for Compressive Sensing

Muhammad A. Lodhi[a], Sergey Voronin[b], and Waheed U. Bajwa[a]

[a]Department of Electrical and Computer Engineering, Rutgers University
[b]Department of Applied Mathematics, University of Colorado Boulder

ABSTRACT

State-of-the-art sparse recovery methods often rely on the restricted isometry property for their theoretical guarantees. However, they cannot explicitly incorporate metrics such as restricted isometry constants within their recovery procedures due to the computational intractability of calculating such metrics. This paper formulates an iterative algorithm, termed yet another matching pursuit algorithm (YAMPA), for recovery of sparse signals from compressive measurements. YAMPA differs from other pursuit algorithms in that: (*i*) it adapts to the measurement matrix using a threshold that is explicitly dependent on two computable coherence metrics of the matrix, and (*ii*) it does not require knowledge of the signal sparsity. Performance comparisons of YAMPA against other matching pursuit and approximate message passing algorithms are made for several types of measurement matrices. These results show that while state-of-the-art approximate message passing algorithms outperform other algorithms (including YAMPA) in the case of well-conditioned random matrices, they completely break down in the case of ill-conditioned measurement matrices. On the other hand, YAMPA and comparable pursuit algorithms not only result in reasonable performance for well-conditioned matrices, but their performance also degrades gracefully for ill-conditioned matrices. The paper also shows that YAMPA uniformly outperforms other pursuit algorithms for the case of thresholding parameters chosen in a clairvoyant fashion. Further, when combined with a simple and fast technique for selecting thresholding parameters in the case of ill-conditioned matrices, YAMPA outperforms other pursuit algorithms in the regime of low undersampling, although some of these algorithms can outperform YAMPA in the regime of high undersampling in this setting.

1. INTRODUCTION

Compressive sensing (CS) has gained a lot of interest in the signal processing community in the past decade. Compressive measurements allow one to sample sparse signals at a much lower rate than that dictated by the Whittaker–Nyquist–Kotelnikov–Shannon sampling theorem and reconstruct them with high fidelity.[1–5] Given a k-sparse signal $\mathbf{u} \in \mathbb{C}^n$ and an $m \times n$ measurement matrix $\mathbf{A}$, compressive measurements can be expressed as $\mathbf{y} = \mathbf{A}\mathbf{u}$, where $\mathbf{y} \in \mathbb{C}^m$ is the vector corresponding to $m \ll n$ measurements. Therefore, the problem of recovering a sparse signal from compressive measurements is essentially equivalent to solving an underdetermined system of linear equations with sparsity constraints. Computationally, this is a very challenging task and a significant portion of research in the CS literature is devoted to addressing this challenge. To this end, a number of methods have been devised that rely on various linear algebra techniques as well as optimization and probability theory for finding efficient solutions to CS problems. Our goal in this paper is to formulate another such efficient method and discuss its relationship with various state-of-the-art sparse recovery methods.

Relation to prior work: Methods for recovery of sparse signals from compressive measurements can be broadly divided into four categories: optimization-based methods, iterative thresholding algortihms, greedy algorithms (also referred to as matching pursuit algorithms) and approximate message passing algorithms. A well-known example of optimization-based methods is basis pursuit (BP), which aims to solve the sparse recovery problem by formulating it as a linear program.[6,7] Although this method is quite robust and has rigorous guarantees for reconstruction,[6–8] it suffers from high computational costs. Iterative hard thresholding (IHT), an example of iterative thresholding algortihms, thresholds a signal estimate plus a signal *proxy* using a nonlinear

Emails—MAL: masad.lodhi@rutgers.edu, SV: sergey.voronin@colorado.edu, WUB: waheed.bajwa@rutgers.edu

thresholding function that returns the k largest (in magnitude) entries of its argument to obtain a new estimate in each iteration.[9,10] IHT is considerably faster than BP and it too has theoretical guarantees for recovery and reconstruction of sparse signals. One of the most familiar examples of greedy algorithms is orthogonal matching pursuit (OMP) (and its many variants).[11–13] OMP estimates a sparse signal from compressive measurements by iteratively estimating the *support* of the signal, one index at a time, and then solving a restricted least-squares problem. While there exist theoretical guarantees for OMP,[14,15] it has been observed that optimization-based methods such as BP tend to perform better than greedy algorithms such as OMP. Nonetheless, OMP is often preferred in many situations due to its low computational cost. Finally, approximate message passing (AMP) is used to describe a fairly recent class of Bayesian sparse recovery methods with remarkable empirical performance, asymptotic theoretical guarantees, and low computational costs.[16–21] In particular, empirical results reported in recent literature suggest AMP algorithms either match or surpass the performance of other sparse recovery methods for random measurement matrices.[18,19]

Despite the availability of such a plethora of efficient methods for sparse signal recovery, many of them have a major limitation: *they cannot explicitly adapt to the underlying measurement matrix*. Specifically, while theoretical analyses of IHT, OMP, etc., rely on the measurement matrix satisfying the *restricted isometry property* (RIP),[22] explicit calculation of RIP constants is a computationally intractable problem.[23] Because of this reason, none of the aforementioned methods can explicitly incorporate RIP constants within their frameworks. Similarly, while AMP algorithms are formulated under the assumption of independent and identically distributed (iid) (sub)Gaussian entries within the measurement matrix, their formulations remain unchanged in the case of non-iid matrices. This inability of existing efficient algorithms to adapt to the specific measurement matrix has serious implications for recovery of sparse signals in CS applications with non-standard measurement matrices.

Our contributions: In this paper, we formulate a new algorithm for recovery of sparse signals that is not only computationally efficient, but is also able to explicitly adapt to the underlying measurement matrix through the use of two coherence metrics of matrices within its formulation. Our proposed algorithm belongs to the class of matching pursuit algorithms; it iteratively estimates the signal support—possibly multiple indices at a time—by thresholding the match-filtered *residual* vector in each iteration and then updates its signal estimate by solving a restricted least-squares problem. The fundamental difference between this algorithm, which we term *yet another matching pursuit algorithm* (YAMPA), and popular matching pursuit algorithms such as OMP, stagewise orthogonal matching pursuit (StOMP),[24] and compressive sampling matching pursuit (CoSaMP)[13] is that the threshold in YAMPA is an explicit function of two coherence metrics of the underlying measurement matrix. These two metrics quantify similarities between the columns of a measurement matrix and, under the assumption of the measurement matrix $\mathbf{A}$ having unit ℓ_2-norm columns, are formally defined as:

$$\begin{aligned} \text{Worst-Case Coherence:} \quad & \mu(\mathbf{A}) = \max_{i,j;i\neq j} \big|\langle \mathbf{a}_i, \mathbf{a}_j \rangle\big|, \text{ and} \\ \text{Average Coherence:} \quad & \nu(\mathbf{A}) = \frac{1}{n-1} \max_{i} \Big| \sum_{j:j\neq i} \langle \mathbf{a}_i, \mathbf{a}_j \rangle \Big|, \end{aligned} \tag{1}$$

where $\mathbf{a}_i$ denotes the i-th column of $\mathbf{A}$. Notice that both these coherence metrics are explicitly computable in polynomial time and, as such, can be incorporated into the frameworks of sparse recovery algorithms. The RIP constants, on the other hand, cannot be explicitly computed for arbitrary matrices and, therefore, cannot be incorporated into sparse recovery frameworks. We refer the reader to [25, Chapter 9] and [26] for further heuristical interpretation of the metrics of worst-case and average coherences. It is worth noting here that the worst-case coherence and the RIP of a matrix are related to each other through the Gersgorin disc theorem [25, Chapter 9]. Unfortunately, RIP constants derived using the worst-case coherence end up being strictly suboptimal [25, Chapter 9].

In addition to the fact that YAMPA adapts to the measurement matrix through the use of the coherence metrics, another notable aspect of the proposed framework is that it does not require an explicit knowledge of the signal sparsity. We conclude by pointing out that while YAMPA lacks theoretical guarantees for its performance, we do establish its usefulness through extensive numerical simulations. In particular, we demonstrate that YAMPA easily outperforms state-of-the-art approximate message passing algorithms for ill-conditioned

measurement matrices. Further, we show that—when combined with a simple and fast technique for selecting the threshold parameters—YAMPA outperforms related matching pursuit algorithms for ill-conditioned matrices with exponentially decaying singular values whenever the number of measurements is much smaller than the length of the signal. In the process, we also pinpoint a key property of the measurement matrices that affects the performance of sparse recovery algorithms, namely, decay of singular values of the measurement matrices. This suggests that CS applications involving ill-conditioned measurement matrices could potentially benefit from any redesign of measurement matrices that reduces the decay of their singular values.

Notation: We now describe some notation used in this paper. The notation $\|\mathbf{A}\|$ and $\|\mathbf{A}\|_F$ refers to the spectral and Frobenius norms of matrices, respectively. By a_{ij}, we refer to the element in row i and column j of $\mathbf{A}$. The notation $\mathbf{a}_j$ refers to the j-th column of $\mathbf{A}$. By $\mathbf{A}^T$ and $\mathbf{A}^H$, we refer to the transpose and Hermitian of $\mathbf{A}$, respectively, while $\mathbf{A}^\dagger$ denotes its Moore–Penrose pseudoinverse. For vectors, $\|\mathbf{v}\|_p$ corresponds to the ℓ_p-norm, $|\mathbf{v}|$ refers to the absolute values of the entries of $\mathbf{v}$ and v_j denotes the j-th entry of $\mathbf{v}$. The notation $\langle \cdot, \cdot \rangle$ is used to represent the innerproduct of two vectors. For a set of positive integers $\mathbf{\Omega}$, $\mathbf{A}_{\mathbf{\Omega}}$ and $\mathbf{v}_{\mathbf{\Omega}}$ refer to the submatrix of $\mathbf{A}$ and the subvector of $\mathbf{v}$ obtained by keeping the columns (of $\mathbf{A}$) and the entries (of $\mathbf{v}$) indexed by $\mathbf{\Omega}$, respectively. We also use $\mathbf{0}$ to denote a vector of all zeros. Lastly, for a positive integer n, $[\![n]\!]$ denotes the set of integers $\{1, 2, \ldots, n\}$.

Organization: The rest of this paper is organized as follows. Sec. 2 describes YAMPA and provides a heuristical justification of the threshold used within YAMPA. Sec. 3 details various practical considerations and heuristical approaches for an efficient implementation of YAMPA. The details of the experiments and a discussion of the associated results are provided in Sec. 4. And lastly, the paper is summarized in Sec. 5.

2. YET ANOTHER MATCHING PURSUIT ALGORITHM

In this section, we provide a description of YAMPA and motivate the threshold used within the algorithm. But in order to put things into perspective, we first begin with a brief review of existing matching pursuit algorithms.

2.1 REVIEW OF MATCHING PURSUIT ALGORITHMS

All matching pursuit algorithms work along the same lines. In each iteration s, they first form an estimate of the sparse signal $\mathbf{u}$ and then remove the contribution of the estimates from compressive measurements $\mathbf{y}$ to form a residual $\mathbf{r}_s$. This process is repeated for a number of iterations until some appropriate conditions (depending on the specific algorithm) are met, at which point the algorithm is terminated. To find the estimate in each iteration, first a proxy vector of correlations with the residual is formed as $\mathbf{q} = |\mathbf{A}^H \mathbf{r}_s|$. This proxy vector is then *thresholded* to get a set of indices as $\mathbf{T}_s = \{i | \, q_i > \lambda_s\}$, with λ_s depending on the specific pursuit algorithm. This set of indices corresponds to an estimate of locations of the nonzero entries of the sparse signal in the s-th iteration. This estimate of the signal *support* is then either merged with or appended to the support estimate from previous iterations to obtain a global support set $\mathbf{T}_{gl}$. Finally a restricted least-squares problem is solved to obtain an estimate of the entries of $\mathbf{u}$ corresponding to $\mathbf{T}_{gl}$ in the s-th iteration.

In terms of the specific details, the threshold in OMP is chosen such that only one new coordinate enters the support estimate in each iteration. CoSaMP, on the other hand, sets the threshold so that $2k$ components are added to the local support in each iteration (where k is the sparsity of the signal and is assumed to be known). Afterwards, CoSaMP performs a pruning step at the end of each iteration to get a support estimate of size k. Finally in StOMP, the threshold is more complex and can be expressed as $\lambda_s = \frac{t\|\mathbf{r}_s\|_2}{\sqrt{m}}$. Here, t is a parameter that depends on the dimensions of the measurement matrix $\mathbf{A}$ and the signal sparsity k.

2.2 YAMPA

YAMPA follows the same steps as other matching pursuit algorithms. But the distinguishing feature of YAMPA is the threshold it uses to form the support estimate in each iteration: the threshold depends on the coherence metrics of the measurement matrix, as opposed to it being independent of the matrix (as in OMP and CoSaMP) or dependent on just the dimensions of the matrix (as in StOMP). In particular, similar to CoSaMP and StOMP,

YAMPA adds multiple indices to the support estimate in each iteration and thus requires much less number of iterations than that needed in OMP.

In terms of specific details, consider a k-sparse signal $\mathbf{u} \in \mathbb{C}^n$, an $m \times n$ measurement matrix $\mathbf{A}$ with unit–norm columns and compressive measurements $\mathbf{y} \in \mathbb{C}^m$. YAMPA first initializes the following entities: initial estimate of the k-sparse signal $\hat{\mathbf{u}}_0 = \mathbf{0}$, the initial global support set $\mathbf{T}_{gl} = \phi$ and the iteration number $s = 0$. It then runs until one of the stopping criteria is satisfied (details in Sec. 3.1). The s-th iteration of our recovery procedure (starting with $s = 1$) starts by making the s-th residual signal $\mathbf{r}_s = \mathbf{y} - \mathbf{A}\hat{\mathbf{u}}_{s-1}$. Using $\mathbf{r}_s$, we form the (absolute-valued) proxy vector $\mathbf{q} = |\mathbf{A}^H \mathbf{r}_s|$ and find local support set as $\mathbf{T}_s = \{i |\, q_i > \lambda_s\}$. Finally the new and old supports are merged as $\mathbf{T}_{gl} = \mathbf{T}_{gl} \cup \mathbf{T}_s$ with the s-th estimate acquired through a restricted least-squares problem: $\hat{\mathbf{u}}_{s_{\mathbf{T}_{gl}}} = \mathbf{A}^{\dagger}_{\mathbf{T}_{gl}} \mathbf{y}$, where $\hat{\mathbf{u}}_{s_{\mathbf{T}_{gl}}}$ represents the indices of $\hat{\mathbf{u}}_s$ indexed by $\mathbf{T}_{gl}$ and the rest of the indices are set to zero. The whole procedure is outlined again in Algorithm 1.

Algorithm 1: Yet Another Matching Pursuit Algorithm (YAMPA)

Input: Compressive measurements $\mathbf{y}$ and measurement matrix $\mathbf{A}$.
Initialization: Initial estimate $\hat{\mathbf{u}}_0 \leftarrow \mathbf{0}$, initial global support $\mathbf{T}_{gl} \leftarrow \emptyset$, and iteration number $s \leftarrow 0$.
1: **while** stopping criteria is satisfied **do**
2: $s \leftarrow s + 1$
3: Obtain s-th residual signal $\mathbf{r}_s \leftarrow \mathbf{y} - \mathbf{A}\hat{\mathbf{u}}_{s-1}$
4: Form (absolute-valued) proxy vector $\mathbf{q} \leftarrow |\mathbf{A}^H \mathbf{r}_s|$
5: Find local support set $\mathbf{T}_s \leftarrow \{i |\, q_i > \lambda_s\}$
6: Merge supports: $\mathbf{T}_{gl} \leftarrow \mathbf{T}_{gl} \cup \mathbf{T}_s$
7: Find new estimate $\hat{\mathbf{u}}_{s_{\mathbf{T}_{gl}}} \leftarrow \mathbf{A}^{\dagger}_{\mathbf{T}_{gl}} \mathbf{y}$
8: **end while**
Output: Final estimate $\hat{\mathbf{u}}_s$.

As noted earlier, the main difference between YAMPA and related algorithms is the threshold λ_s. In case of YAMPA we require that

$$\lambda_s = c_1 \mu \|\mathbf{r}_s\|_2 + c_2 \nu \sqrt{\hat{k}} \|\mathbf{r}_s\|_2, \tag{2}$$

where c_1 and c_2 are empirically determined constants, μ and ν are the worst-case and average coherences of $\mathbf{A}$, and $\hat{k} = m/\log(n)$ is a crude upper bound on the signal sparsity.[2] We use $\hat{k}$ because we do not assume knowledge of the signal sparsity, but it can be replaced by a better upper bound (or exact k), if known. A detailed discussion of our rationale for the threshold in (2) is provided in the next section. We conclude by noting that the focus of this paper is on sparse recovery in **noiseless settings**, but the results can be extended to include settings with noise. More precisely, the threshold can be modified to account for noisy measurements (along the lines of Ref. 27) to make the YAMPA work in noisy settings.

2.3 DERIVATION OF THE THRESHOLD FOR YAMPA

In this section we justify the use of the thresholding procedure in (2) for YAMPA by using [27, Lemma 4]. Before proceeding further, however, it is important to point out that our derivation is somewhat heuristic in nature. First, although our derivation does not require a particular form of the measurement matrix, it does assume the locations of the nonzero entries of $\mathbf{u}$ to be drawn uniformly at random from $\{1, \ldots, n\}$. This, however, is a rather mild probabilistic assumption and simply reflects our lack of knowledge about the signal support. Second, our derivation is focused only on the first iteration of YAMPA. Specifically, we establish that the threshold given in (2) ensures $\mathbf{T}_1 \subset \mathit{support}(\mathbf{u})$ with high probability. The idea here is that if YAMPA selects only indices from the actual signal support in each iteration then contribution of the signal corresponding to those indices can be subtracted from the compressive measurements and this procedure can be repeated till the entire signal support is exhausted. In order to formally establish this fact, however, we need to ensure $\mathbf{T}_s \subset \mathit{support}(\mathbf{u})$, $\forall s > 1$. But formally proving this for the case of $s > 1$ becomes nontrivial and, as such, is not attempted in this paper. Despite these shortcomings of our derivation, extensive numerical experiments reported in Sec. 4.3 suggest the usefulness of the derived threshold.

We begin our derivation by considering a deterministic k-sparse vector $\bar{\mathbf{x}} \in \mathbb{C}^n$ with all nonzero entries in the first k locations. Next we also consider an $n \times n$ permutation matrix $\mathbf{P}_\omega = [\mathbf{e}_{\omega_1} \quad \mathbf{e}_{\omega_2} \quad \dots \quad \mathbf{e}_{\omega_n}]^T$, where $\mathbf{e}_j$ is the j-th column of an $n \times n$ identity matrix and $\bar{\mathbf{\Omega}} = (\omega_1, \omega_2, \dots, \omega_n)$ is a random permutation on $[\![n]\!]$. We also define $\mathbf{\Omega}$ to be the first k entries of $\bar{\mathbf{\Omega}}$ and $\mathbf{\Omega}^c$ to be the set of last $n-k$ elements of $\bar{\mathbf{\Omega}}$ that are not present in $\mathbf{\Omega}$. In the following, we assume the compressive measurements $\mathbf{y}$ to be given by:

$$\mathbf{y} = \mathbf{A}\mathbf{u} = \mathbf{A}\mathbf{P}_\omega \bar{\mathbf{x}} = \mathbf{A}_{\mathbf{\Omega}}\mathbf{x}, \tag{3}$$

where $\mathbf{A}$ is the $m \times n$ measurement matrix, $\mathbf{x}$ is the vector that contains only the first k nonzero entries of $\bar{\mathbf{x}}$ and $\mathbf{A}_{\mathbf{\Omega}}$ is the $m \times k$ submatrix containing the columns of $\mathbf{A}$ indexed by the elements of $\mathbf{\Omega}$. Notice that the support of $\mathbf{u}$ in this case is simply given by $support(\mathbf{u}) = \Omega$.

We are now ready to derive our threshold for the $s = 1$ from the proof of [27, Lemma 4]. The proxy vector for the first iteration is given by $\mathbf{q} = |\mathbf{A}^H\mathbf{y}| = |\mathbf{A}^H\mathbf{A}_{\mathbf{\Omega}}\mathbf{x}|$. Notice that $\mathbf{q}_\Omega = |\mathbf{A}_{\mathbf{\Omega}}^H\mathbf{A}_{\mathbf{\Omega}}\mathbf{x}|$ and $\mathbf{q}_{\Omega^c} = |\mathbf{A}_{\mathbf{\Omega^c}}{}^H\mathbf{A}_{\mathbf{\Omega}}\mathbf{x}|$. In order to establish $\mathbf{T}_1 \subset \Omega$ we need to show that $\|\mathbf{q}_{\Omega^c}\|_\infty < \lambda$ with high probability for some $\lambda > 0$. To this end, notice that $\|\mathbf{q}_{\Omega^c}\|_\infty = \|\mathbf{A}_{\mathbf{\Omega^c}}{}^H\mathbf{A}_{\mathbf{\Omega}}\mathbf{x}\|_\infty = \max\limits_{i \in [\![n-k]\!]} \Big| \sum_j x_j \langle \mathbf{a}_{\omega_i^c}, \mathbf{a}_{\omega_j} \rangle \Big|$ where ω_i^c denotes the i-th element of Ω^c. Now defining the event $\mathcal{A}_{i'} = \{\omega_i^c = i'\}$, our goal is to show that $\Pr\Big(\Big| \sum_{j=1}^k x_j \langle \mathbf{a}_{\omega_i^c}, \mathbf{a}_{\omega_j} \rangle \Big| > \lambda \mid \mathcal{A}_{i'} \Big)$ is small. We resort to the *method of bounded differences*[28] for this purpose. Specifically, we first define the $(k-1)$-vector $\mathbf{\Omega}^{-i} = (\omega_1, \dots, \omega_{i-1}, \omega_{i+1}, \dots, \omega_k)$ and then construct a Doob Martingale as follows:

$$M_0 = \mathbb{E}\Big[\sum_{j=1}^k x_j \langle \mathbf{a}_{\omega_i^c}, \mathbf{a}_{\omega_j} \rangle | \mathcal{A}_{i'} \Big] = \mathbb{E}\Big[\sum_{j=1}^k x_j \langle \mathbf{a}_{i'}, \mathbf{a}_{\omega_j} \rangle | \mathcal{A}_{i'} \Big], \text{and} \tag{4}$$

$$M_l = \mathbb{E}\Big[\sum_{j=1}^k x_j \langle \mathbf{a}_{\omega_i^c}, \mathbf{a}_{\omega_j} \rangle | \omega_{1\to l}^{-i}, \mathcal{A}_{i'} \Big] = \mathbb{E}\Big[\sum_{j=1}^k x_j \langle \mathbf{a}_{i'}, \mathbf{a}_{\omega_j} \rangle | \omega_{1\to l}^{-i}, \mathcal{A}_{i'} \Big],\ l = 1, 2, \dots, k, \tag{5}$$

where $\omega_{1\to l}^{-i}$ denotes the first l elements of Ω^{-i}. Now similar to [27, Lemma 4], we use the linearity of expectation to get:

$$|M_0| = \Big| \sum_{j=1}^k x_j \mathbb{E}\Big[\langle \mathbf{a}_{i'}, \mathbf{a}_{\omega_j} \rangle | \mathcal{A}_{i'} \Big] \Big| \leq \sum_{j=1}^k |x_j| \Big| \mathbb{E}\Big[\langle \mathbf{a}_{i'}, \mathbf{a}_{\omega_j} \rangle | \mathcal{A}_{i'} \Big] \Big| \leq \nu \|\mathbf{x}\|_1. \tag{6}$$

The proof of [27, Lemma 4] also provides a bound on the difference of martingales, given as follows:

$$|M_l(r) - M_l(s)| \leq 2\mu c_l, \tag{7}$$

where $M_l(r) = \mathbb{E}\Big[\sum_{j=1}^k x_j \langle \mathbf{a}_{i'}, \mathbf{a}_{\omega_j} \rangle | \omega_{1\to l-1}^{-i}, \omega_l = r, \mathcal{A}_{i'} \Big]$ and $c_l = |x_l| + \frac{\sum_{j>l} |x_j|}{n-l-1}$. It can be also verified through calculations that $\sum_{l=1}^k c_l^2 \leq (1 + a^{-1})^2 \|x\|_2^2$ under the assumption of $k \leq \frac{n}{1+a}$. Then using complex Azuma Inequality [27, Lemma 5], we obtain:

$$\Pr\Big(\Big| \sum_{j=1}^k x_j \langle \mathbf{a}_{i'}, \mathbf{a}_{\omega_j} \rangle \Big| > \lambda \Big| \mathcal{A}_{i'} \Big) \leq \Pr\Big(\Big| M_k - M_0 \Big| > \lambda - \nu \|\mathbf{x}\|_1 \Big| \mathcal{A}_{i'} \Big) \leq 4 \exp\Big(- \frac{(\lambda - \nu\|\mathbf{x}\|_1)^2}{8(1 + a^{-1})^2 \mu^2 \|\mathbf{x}\|_2^2} \Big). \tag{8}$$

We need to find a λ such that $\Pr\Big(\Big| \sum_{j=1}^k x_j \langle \mathbf{a}_{i'}, \mathbf{a}_{\omega_j} \rangle \Big| > \lambda \Big| \mathcal{A}_{i'} \Big) \leq 4\delta$ for a small enough $\delta \in (0, 1)$. Defining $\alpha = 8(1 + a^{-1})^2$, this simply leads to:

$$\frac{(\lambda - \nu\|\mathbf{x}\|_1)^2}{\alpha \mu^2 \|\mathbf{x}\|_2^2} \geq \log \delta^{-1} \Leftrightarrow \lambda - \nu\|\mathbf{x}\|_1 \geq \mu \sqrt{\alpha \log \delta^{-1}} \|\mathbf{x}\|_2 \Leftrightarrow \lambda \geq \mu \sqrt{\alpha \log \delta^{-1}} \|\mathbf{x}\|_2 + \nu \|\mathbf{x}\|_1, \tag{9}$$

and using $\|\mathbf{x}\|_1 \leq \sqrt{k}\|\mathbf{x}\|_2$ and $\|\mathbf{x}\|_2 \leqslant \|\mathbf{y}\|_2 = \|\mathbf{r}\|_2$ for $s = 1$, we can use:

$$\lambda = c_1 \mu \|\mathbf{r}\|_2 + c_2 \sqrt{k} \nu \|\mathbf{r}\|_2, \tag{10}$$

where k is replaced by the upper bound on signal sparsity $\hat{k}$, while c_1 and c_2 are to be determined empirically (see Sec. 3.2 for details). It should be noted here that we have derived our threshold conditioned on one occurrence of ω_i^c, i.e., $\mathcal{A}_{i'} = \{\omega_i^c = i'\}$. But since ω_i^c's are identically (not independently) distributed and event $\mathcal{A}_{i'}$ has a uniform distribution over $[\![n]\!]$, we can show, using basic facts of probability, that our choice of λ ensures $\Pr\left(\|\mathbf{q}_{\Omega^c}\|_\infty > \lambda\right)$ is also small. This concludes our derivation of the threshold for YAMPA.

3. PRACTICAL CONSIDERATIONS

We now discuss some practical aspects that have to be considered for implementation of YAMPA. These include the stopping criteria and the procedure for selecting constants in the threshold. Our discussion is based on experimental observations and an intuitive understanding of the sparse recovery problem.

3.1 STOPPING CRITERIA FOR YAMPA

Since YAMPA is iterative in nature, we have to specify criteria under which it terminates successfully and efficiently. Otherwise, the algorithm might keep iterating indefinitely or might start adding indices to the support estimate that do not belong to the original support. Some of these stopping conditions are listed below:

1. **Empty local support set**: In any iteration 's' ($s > 1$), an empty local support, i.e., $\mathbf{T}_s = \{i | q_i > \lambda_s\} = \emptyset$, indicates the energy left in the residual $\mathbf{r}_s$ is likely not due to the signal. Therefore, we terminate YAMPA whenever an empty local support set is encountered in an iteration.

2. **Unchanged global support set for two iterations**: An unchanged global support set for two consecutive iterations is an indication that YAMPA cannot extract any new information from the residual signal. Thus, we terminate YAMPA whenever this condition is met. While this stopping condition is rarely met in practice, we did observe in our experiments that sometimes we get a nonempty local support set in higher iterations that, when merged with the global support set, does not add any new elements. This condition is helpful in terminating the algorithm in such situations.

3. **Norm of residual below a tolerance level**: This is a standard stopping criterion in compressive sensing literature and is needed to account for finite-precision calculations. Specifically, once the norm of the residual falls below a certain level, it is better to terminate the algorithm rather than add negligible components to the signal estimate.

4. **Maximum number of iterations reached**: This user-specified parameter, as the name implies, terminates YAMPA after a maximum number of allowable iterations is reached.

5. **New threshold larger than previous threshold (backtracking)**: The YAMPA threshold in Sec. 2.3 is derived under the assumption that only indices belonging to the signal support get added to the support estimate in each iteration. This of course dictates that the residual vector $\mathbf{r}_s$ should have lower and lower energy with each successive iteration. While this does happen for a normal run of YAMPA, there are instances where it ends up picking incorrect support indices. In this case, the residual vector in the next iteration, $\mathbf{r}_{s+1}$, ends up having a larger norm than the previous residual, $\mathbf{r}_s$. Fortunately, this also results in a larger threshold, i.e., $\lambda_{s+1} > \lambda_s$, which is an indication of unsuccessful previous iteration. When this occurs, it is better to *backtrack* one iteration, i.e., go back to the last correct estimate, and then stop. Because of this, we always keep track of our current and previous global support sets in each iteration, and we make use of the previous global support set to get back to the point where no incorrect indices were added to the support estimate and then terminate the algorithm.

Note that the first three conditions are important for YAMPA to terminate efficiently, the fourth condition is a user-defined constraint and the fifth one is meant to prevent the addition of wrong indices to the support set. In the end, all of these criteria signify settings in which we need to terminate the algorithm because keeping the algorithm running would either be wasteful of resources or would introduce false entries in the estimated signal.

3.2 SELECTION OF CONSTANTS IN YAMPA THRESHOLD

The exact threshold expression derived in Sec. 2.3 is just a lower bound on the threshold needed for YAMPA to succeed. In practice, the constants c_1 and c_2 in (2) should be empirically determined. To find these constants, we devise a method called *YAMPA constants finder* (YCF). The basic motivation behind YCF is as follows. We ran YAMPA for a number of different problem setups and exhaustively searched for the best collection of constants using the ground truth. The resulting set of constants suggest that YAMPA's performance stays more or less consistent for a range of values of constants c_1 and c_2. It therefore suffices to select a reasonable set of constants within this range for YAMPA to perform well, which is precisely what YCF does.

YAMPA Constants Finder

It follows from the preceding discussion that any set of constants c_1 and c_2 within a predefined range that does not result in *abnormal behavior* is a *good candidate*. To select the best candidate among good candidates in an efficient manner, YCF makes use of the residual energy and the cardinality of the global support set. Specifically, YCF first involves running YAMPA for just **two iterations** over the predefined (experimentally determined) range of constants and discarding any combination of constants that invokes any of the stopping criteria (i.e., abnormal behavior). The idea is that any combination of constants that makes YAMPA quit in two iterations results in a threshold that is either too high or too low. We declare the remaining sets of constants as *good* and, for each of these sets, we next focus on two parameters: *cardinality of global support set* and *residual energy*.

Heuristically, the best set of constants among good constants should result in an estimate after two iterations that is the sparsest and that has the least residual energy. However, as evident from an example of YCF in Fig. 2, the set of constants with the sparsest solution can result in the highest residual energy and the set of constants with the least residual energy can have the least sparse solution. Therefore, we need to look for a *balanced* set of constants that results in a sufficiently small cardinality of the global support set and that also produces low residual energy. To choose such a set of constants, we compare the cardinality of the global support set with a *suitable sparsity level* defined as $k_{suit} = (\frac{m}{n} + 1)\frac{m}{\log(n)}$. Notice that this parameter varies between $\frac{m}{\log(n)}$ and $\frac{2m}{\log(n)}$, depending on the ratio $\frac{m}{n}$. Specifically, we sort the sets of constants according to how close the cardinality of their global support set is to k_{suit} and then pick the set with the *first lowest* residual energy. In particular, this pertains to picking the set of constants that corresponds to the first local minima in the sorted residual energies. The example in Fig. 2 illustrates the whole procedure in terms of plots.

4. NUMERICAL EXPERIMENTS AND DISCUSSION

In order to compare and characterize the performance of YAMPA, we run simulations for numerous measurement matrices and test signals. We evaluate the performance for different undersampling factors ($\rho = m/n$) and sparsity factors ($\delta = k/m$). For each combination of undersampling and sparsity factors, we keep the measurement matrix fixed and run **50** trials with random test signals. The number of columns of measurement matrices used throughout these experiments is $n = 1000$, and m and k are picked as $m = \lfloor \rho n \rfloor$ and $k = \delta m$. The undersampling factor (ρ) and sparsity factor (δ) are varied between 0 and 1 to get different values of m and k.

The performance metrics that we focus on for comparison purposes are the *mean relative error*, the *mean success value*, and the *mean support intersect ratio* across different trials for fixed ρ and δ. The *relative error* between a test signal $\mathbf{u}$ and its recovered version $\hat{\mathbf{u}}$ is defined as $\epsilon = \frac{\|\mathbf{u}-\hat{\mathbf{u}}\|_2}{\|\mathbf{u}\|_2}$. The success value for each random test signal is set to "1" if the relative error is less than 0.05 and to "0" if the relative error is greater than 0.05. The support intersect ratio between the support of a test signal, $\mathcal{S}_{\mathbf{u}}$, and the support of its recovered version, $\mathcal{S}_{\hat{\mathbf{u}}}$, is defined as $\zeta = \frac{|\mathcal{S}_{\hat{\mathbf{u}}} \cap \mathcal{S}_{\mathbf{u}}|}{|\mathcal{S}_{\mathbf{u}}|}$, which is the ratio of 'the cardinality of intersection of $\mathcal{S}_{\hat{\mathbf{u}}}$ and $\mathcal{S}_{\mathbf{u}}$' to the 'cardinality of $\mathcal{S}_{\mathbf{u}}$'. In addition to these parameters, the cardinality of the support of the recovered signal, i.e., the number of nonzero entries, is also reported. Finally, for all experiments, the tolerance level set for residual energy is 10^{-6}.

4.1 Measurement Matrices

In this section, we describe the construction of different measurement matrices used to test the performance of YAMPA. It should be noted here the columns of all measurement matrices are normalized to have **unit ℓ_2-norm** after their initial construction.

Normal random matrices: Normal random matrices are matrices with entries drawn from the standard normal distribution in an independent and identically distributed fashion.

Binary random matrices: Binary random matrices have equiprobable binary entries that belong to either $\{0, 1\}$ or $\{+1, -1\}$ and that are drawn in an iid fashon.

Cauchy random matrices: Each entry in a Cauchy random matrix is obtained as tangent of the angle θ that is uniformly distributed on $[-\pi, \pi]$ in an i.i.d. fashion.

Matrices with correlated columns: For this type of matrices, we randomly take p columns from an $m \times (n-p)$ matrix and then add perturbations to these p columns in the form of Gaussian white noise with variance 0.25. These perturbed columns are then appended to the original matrix to make a new $m \times n$ matrix with correlated columns. This procedure can be carried out for any of the matrices described in this section.

Random ill-conditioned matrices (with exponentially decaying singular values): These matrices are formed from the product of three matrices $\mathbf{U}$, $\mathbf{S}$ and $\mathbf{V}$ as $\mathbf{A} = \mathbf{USV}$, where $\mathbf{U}$ and $\mathbf{V}$ are random orthogonal matrices and $\mathbf{S}$ is a diagonal matrix containing *singular values* that have an exponential decay. The dimensions of $\mathbf{U}$, $\mathbf{S}$ and $\mathbf{V}$ are $m \times d$, $d \times d$ and $d \times n$, respectively, where $d = \min(m, n)$. Let us assume that the diagonal entries of $\mathbf{S}$ are between s_{max} and s_{min}, then the i-th diagonal entry s_i is obtained as $s_{max}(\frac{s_{min}}{s_{max}})^{\frac{i-1}{d-1}}$ for $i = 1, \ldots, d$. For neighboring singular values, this construction results in the ratio $\frac{s_i}{s_{i+1}} = (\frac{s_{max}}{s_{min}})^{\frac{1}{d-1}}$. To construct matrices with same extreme singular values but different decays, i.e., to control the decay rate, we introduce another parameter called *cutoff index* (d_{cut}). If $d_{cut} < d$, then for $i = 1, \ldots, d_{cut}$ the diagonal entries are formed as $s_i = s_{max}(\frac{s_{min}}{s_{max}})^{\frac{i-1}{d_{cut}-1}}$ and for $i = d_{cut} + 1, \ldots, d$ we assign $s_i = s_{min}$.

In order to assess the performance of recovery algorithms, we mainly focus on four parameters of these matrices: *maximum singular value* (s_{max}), *minimum singular value* (s_{min}), *condition number* ($\frac{s_{max}}{s_{min}}$), and *the rate of decay of singular values*. Control of these parameters gives us the ability to experiment with different matrix properties and exactly pinpoint the factors that affect the performance of different sparse recovery methods. Finally, note that we allow for $m > n$ for the case of ill-conditioned matrices with correlated columns.

Modified normal random matrices: We define modified normal random matrices to be matrices that are initially constructed as normal random matrices but then their singular values are redistributed in various ways. This redistribution is done so as not to affect the energy of the matrix, defined as $E_A = \sum_{i=1}^{m} s_i^2$, with $s_1, \ldots, s_m$ denoting singular values of the matrix $\mathbf{A}$.

4.2 Test Signals

Besides experimenting with various measurement matrices, we also experiment with different types of test signals such as uniform random signals, binary random signals and normal random signals. In order to make an n-length k-sparse random signal of a particular type, first k samples are drawn from the respective distribution (e.g., normal random signals are obtained from normal distribution) and then k locations are drawn in a uniform fashion from $[\![n]\!]$ to place these samples.

4.3 RESULTS

In this section, we describe the results of our experiments using three algorithms: YAMPA, StOMP and expectation-maximization gaussian-mixture approximate message passing[18] (EMGMAMP). We chose EMGMAMP (among different variants of AMP) because it is a state-of-the-art algorithm with the best performance among many variants of AMP.[18] Similarly, we picked StOMP because it has the best performance among matching pursuit algorithms.[24] To assess the performance of YAMPA, we focus on two setups: one in which we pick constants for the threshold using YCF and the other in which we pick constants in an oracle fashion (i.e., with the knowledge of the ground truth). We use the second setup to get an idea of the best-case performance of YAMPA.

4.3.1 Normal random matrices

We start our discussion with normal random matrices and sparse uniformly distributed test signals. For normal random matrices (and any type of test signal), the algorithm with best performance is EMGMAMP. The rest of the algorithms, i.e., StOMP, YAMPA with oracle, and YAMPA with YCF have similar performances as shown in Fig. 3. The graphs in Fig. 3 also show that YCF picks good constants and gives performance close to thet of the oracle setup. The results for other test signals (binary random signals and normal random signals) look quite similar to Fig. 3 and are hence omitted. Plots for Binary random matrices look quite similar to the plots for normal random matrices and are therefore also omitted.

Since EMGMAMP performs quite well for normal random matrices (better than YAMPA and StOMP), one can question the need for another sparse recovery method. In practical situations, however we don't always have well-conditioned random matrices of this sort as measurement matrices. Instead, we often come across measurement matrices that are ill-conditioned and, as it will be demonstrated in later sections, EMGMAMP breaks down for certain ill-conditioned matrices. While YAMPA, on the other hand, suffers a loss in performance, it does not break down completely (see Sec. 4.3.3 for further details).

4.3.2 Cauchy random matrices

For Cauchy random matrices, EMGMAMP does not perform well in high undersampling conditions but as the number of samples increases, the performance of EMGMAMP gets better (Fig. 4). This initial degradation for EMGMAMP is expected, given that it was primarily designed with normal random matrices in mind and for Cauchy random matrices the assumption of Gaussianity is broken.[18] As for YAMPA and StOMP, we can see from Fig. 4 that the performance of StOMP worsens more rapidly compared to YAMPA, whose performance degrades gradually. However, the regions where StOMP does recover the test signals, it recovers with lesser errors than YAMPA. The results for binary random signals and normal random signals using Cauchy measurement matrices are similar to the results for uniform random signals which are reported in Fig. 4.

4.3.3 Random ill-conditioned matrices

For the case of random ill-conditioned matrices, we have four parameters that can affect the wellness of the matrix in terms of recovery and we experiment with all of these to find out which parameters, individually or in combination with others, actually effect the recovery performance. For all the experiments done with random ill-conditioned matrices, EMGMAMP breaks down completely (as shown in Fig. 5), since the assumption of normal measurement matrices is completely violated. So in all the experiments with these matrices we only compare YAMPA with StOMP. It should be noted here that both setups of YAMPA outperform StOMP in the high undersampling regime (i.e., for low values of ρ), while StOMP outperforms in low undersampling regime (i.e., for high values of ρ). Further discussion of these experiments with ill-conditioned matrices is provided at the end of this section. The performance plots for uniform random signals using random ill-conditioned matrices are shown in Fig. 5.

4.3.4 Matrices with correlated columns

To study the effect of adding correlated columns to a measurement matrix on recovery performance of sparse recovery algortihms, we run experiments using normal random matrices and ill-conditioned matrices with increasing number of correlated columns. The specific effect of this procedure depends on how the decay of singular values of a particular measurement matrix changes in response to the addition of correlated columns.

For ill-conditioned matrices, the performance of YAMPA and StOMP actually improves as we introduce more correlated columns (since EMGMAMP breaks down for ill-conditioned matrices, we do not investigate it further). This phenomena occurs due to the fact that inclusion of correlated columns introduces more energy in the lower singular values of the matrix. This essentially has two effects: the condition number is lowered and the decay of singular values becomes less rapid. Both of these events contribute towards better performance with correlated columns in the matrix. This is also evident from the plots in Fig. 6.

On the other hand, the opposite is true for normal random matrices with correlated columns. Adding correlated columns degrades the performance for all algorithms for exactly the opposite reason. Introducing correlated columns introduces more energy into the higher singular values of the matrix and thus results in poor

performance. In particular, EMGMAMP starts to break down completely as the number of correlated columns is increased. The final results for this set of experiments are shown in Fig. 7.

4.3.5 Discussion

In this section, we expand on our previous experimental observations involving ill-conditioned measurement matrices. As noted earlier, we have control over four parameters related to random ill-conditioned matrices. We want to study the effect of each one of these parameters on the recovery performance. We start by exploring the effects of the condition number. In this regard, the first parameter that we manipulate is *maximum singular value*. Specifically, to increase the condition number ($\frac{s_{max}}{s_{min}}$), we increase the maximum singular value by scaling it with a factor α greater than 1, i.e., $\bar{s}_{max} = \alpha s_{max}$ with $\alpha > 1$. While doing this, we keep other matrix parameters (the minimum singular value and the rate of decay of singular values) and the total energy, E_A, in the matrix fixed. In order to keep E_A constant, we reduce all the other singular values by an appropriate amount. Doing this increases the condition number by the same factor α and makes the matrix more ill-conditioned. Our experiments suggest that just increasing the maximum singular value has little effect on the performance of YAMPA and StOMP as shown in Fig. 8. Similar experiments in which we scale the minimum singular value by a factor less than 1 to increase the condition number lead us to believe that decreasing the minimum singular value by itself doesn't have much effect on the performance either.

To further investigate the effect of condition number, we also run experiments with *modified normal random matrices* that have same condition number and energy but different extreme singular values. The extreme singular values of these matrices differ by a scale of $\bar{\alpha} > 1$. Specifically, let $s_{1,max}$ and $s_{1,min}$ denote the extreme singular values of the first matrix. Similarly, let $s_{2,max}$ and $s_{2,min}$ denote the extreme singular values of the second matrix. Then we have the following relation between their singular values: $s_{2,max} = \bar{\alpha} s_{1,max}$ and $s_{2,min} = \bar{\alpha} s_{1,min}$. In order to keep the energy same in both matrices, we redistribute the energy in the non-extreme singular values of the second matrix. The results suggest that matrices with same condition number but with different distribution of singular values tend to have different performance; see Fig. 9 for further details.

Next, we focus on the decay of singular values of the matrices and investigate its affect on performance. For this purpose we run simulations using matrices with different rates of exponential decay and different minimum singular values, but with same maximum singular values and energy. The rate of decay in these experiments is controlled by the cutoff index d_{cut} (Sec. 4.1) and minimum singular value is controlled by a scale factor $\gamma > 0$. These experiments (see Fig. 10) show that when the minimum singular value is small, faster decay leads to worse recovery performance. This suggests matrices with a fast decaying tail of singular values tend to result in worse performance.

To further substantiate this observation, we devise a procedure for *modified normal random matrices* to redistribute energy from a subset of adjacent singular values to another subset of singular values (all the while keeping the total energy constant). We run experiments in which we redistribute energy from lower (singular values in the tail), middle and upper singular values to other regions of singular values. These experiments verify our observation that sparse recovery performance depends on the distribution of energy in the tail of singular values. These results are supported by plots in Fig. 11 and Fig. 12.

5. CONCLUSION

In conclusion, we devised a new matching pursuit algorithm, termed *yet another matching pursuit algorithm* (YAMPA) that incorporates two coherence metrics of the measurement matrix within its thresholding procedure. Although state-of-the-art algorithms (e.g., EMGMAMP) work better than YAMPA for random well-conditioned matrices, they break down for ill-conditioned matrices and suffer performance loss for measurement matrices with correlated columns. Such matrices tend to arise in many practical settings and that is where YAMPA makes a contribution. We established the advantages of using YAMPA through extensive simulations and in the process we also identified key parameters that tend to affect the recovery performance of sparse recovery algorithms. Our work demonstrated that rapidly decaying singular values are likely to be the true culprit for bad recovery performance. In terms of future work, more work is required to refine the procedure for selection of constants. There are other questions that also need to be addressed, including the fact that the threshold is derived for the first iteration of YAMPA only and as of yet we have no analysis of how the threshold should change from iteration to iteration.

ACKNOWLEDGMENTS

This paper was made possible by grant number NPRP 06-070-2-024 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

REFERENCES

[1] Candès, E. J. et al., "Compressive sampling," *Proc. Int. Congr. Math.* **3**, 1433–1452 (2006).

[2] Candès, E. J., Romberg, J., and Tao, T., "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory* **52**, 489–509 (2006).

[3] Candes, E. J., Romberg, J. K., and Tao, T., "Stable signal recovery from incomplete and inaccurate measurements," *Commun. Pure Appl. Math.* **59**, 1207–1223 (2006).

[4] Candes, E. J. and Tao, T., "Near-optimal signal recovery from random projections: Universal encoding strategies?," *IEEE Trans. Inf. Theory* **52**, 5406–5425 (2006).

[5] Donoho, D. L., "Compressed sensing," *IEEE Trans. Inf. Theory* **52**, 1289–1306 (2006).

[6] Chen, S. S., Donoho, D. L., and Saunders, M. A., "Atomic decomposition by basis pursuit," *SIAM Rev.* **43**, 129–159 (2001).

[7] Donoho, D. L. and Elad, M., "On the stability of the basis pursuit in the presence of noise," *Signal Processing* **86**, 511–532 (2006).

[8] Donoho, D. L., "For most large underdetermined systems of linear equations the minimal $\ell 1$-norm solution is also the sparsest solution," *Commun. Pure Appl. Math.* **59**, 797–829 (2006).

[9] Blumensath, T. and Davies, M. E., "Iterative hard thresholding for compressed sensing," *Appl. Comp. Harmonic Anal.* **27**, 265–274 (2009).

[10] Blumensath, T. and Davies, M. E., "Iterative thresholding for sparse approximations," *Journal of Fourier Analysis and Applications* **14**, 629–654 (2008).

[11] Mallat, S. G. and Zhang, Z., "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Process.* **41**, 3397–3415 (1993).

[12] Needell, D. and Vershynin, R., "Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit," *Foundations of Computational Mathematics* **9**, 317–334 (2009).

[13] Needell, D. and Tropp, J. A., "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comp. Harmonic Anal.* **26**, 301–321 (2009).

[14] Tropp, J. A. and Gilbert, A. C., "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory* **53**, 4655–4666 (2007).

[15] Davenport, M. A. and Wakin, M. B., "Analysis of orthogonal matching pursuit using the restricted isometry property," *IEEE Trans. Inf. Theory* **56**, 4395–4401 (2010).

[16] Donoho, D. L., Maleki, A., and Montanari, A., "Message passing algorithms for compressed sensing: I. motivation and construction," *Proc. IEEE Inf. Theory Workshop (ITW)* , 1–5 (2010).

[17] Rangan, S., "Generalized approximate message passing for estimation with random linear mixing," *Proc. IEEE Intl. Symp. Information Theory (ISIT)* , 2168–2172 (2011).

[18] Vila, J. P. and Schniter, P., "Expectation-maximization gaussian-mixture approximate message passing," *IEEE Trans. Signal Process.* **61**, 4658–4672 (2013).

[19] Vila, J., Schniter, P., Rangan, S., Krzakala, F., and Zdeborová, L., "Adaptive damping and mean removal for the generalized approximate message passing algorithm," *Proc. IEEE Int. Conf. Acoust. Speech Signal. Process. (ICASSP)* , 2021–2025 (2015).

[20] Donoho, D. L., Maleki, A., and Montanari, A., "Message-passing algorithms for compressed sensing," *Proc. Nat. Acad. Sci.* **106**, 18914–18919 (2009).

[21] Baron, D., Sarvotham, S., and Baraniuk, R. G., "Bayesian compressive sensing via belief propagation," *IEEE Trans. Signal Process.* **58**, 269–280 (2010).

[22] Candès, E. J., "The restricted isometry property and its implications for compressed sensing," *Comptes Rendus Mathematique* **346**, 589–592 (2008).

[23] Bandeira, A. S., Dobriban, E., Mixon, D. G., and Sawin, W. F., "Certifying the restricted isometry property is hard," *IEEE Trans. Inf. Theory* **59**, 3448–3450 (2013).

[24] Donoho, D. L., Tsaig, Y., Drori, I., and Starck, J.-L., "Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit," *IEEE Trans. Inf. Theory* **58**, 1094–1121 (2012).
[25] Bajwa, W. U. and Pezeshki, A., "Finite frames for sparse signal processing," in [*Finite Frames*], 303–335, Springer (2013).
[26] Bajwa, W. U., Calderbank, R., and Mixon, D. G., "Two are better than one: Fundamental parameters of frame coherence," *Appl. Comput. Harmon. Anal.* **33**(1), 58–78 (2012).
[27] Bajwa, W. U., Calderbank, R., and Jafarpour, S., "Why gabor frames? two fundamental measures of coherence and their role in model selection," *J. Commun. Netw.* **12**, 289–307 (2010).
[28] McDiarmid, C., "On the method of bounded differences," *Surveys in Combinatorics* **141**, 148–188 (1989).

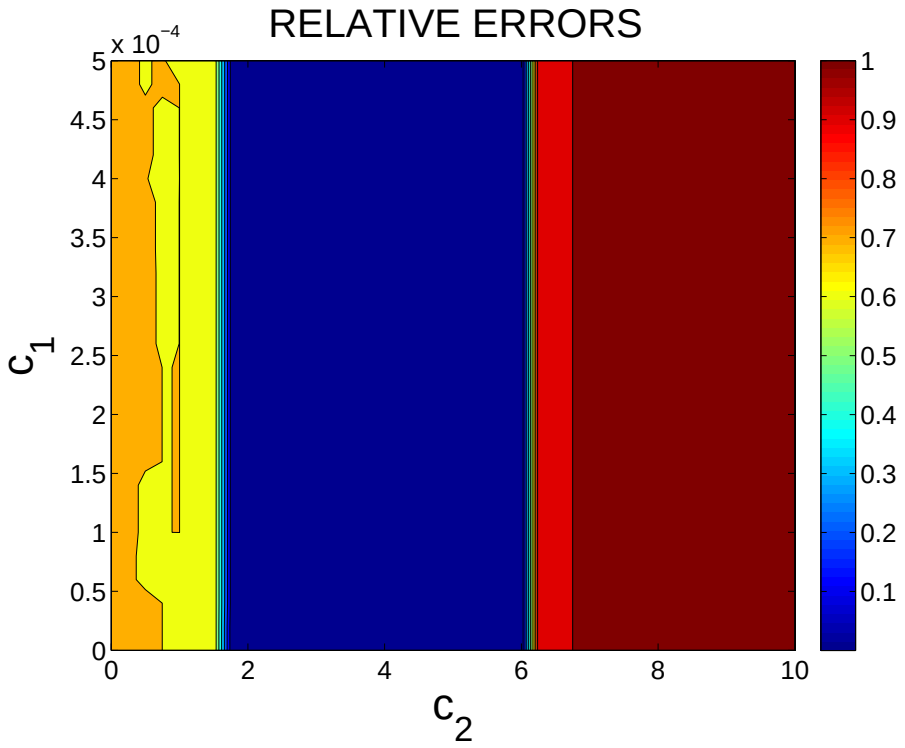

Figure 1. YAMPA results in small error over a range of values for constants c_1 and c_2, rather than for just one particular value for each. This observation motivates our method for selecting constants, YAMPA Constants Finder, in which we search for suitable sets of constants over this experimentally determined range of good constants.

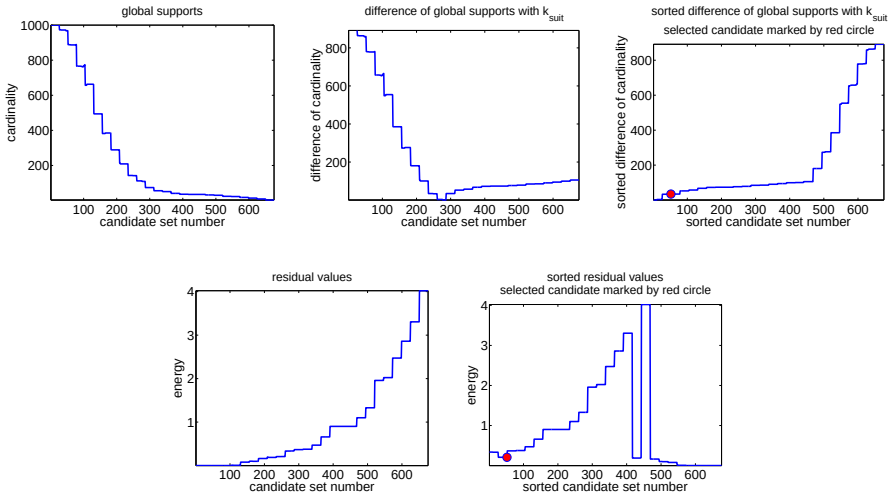

Figure 2. This is an example of how YCF works. The first plot (from left) in the first row is for the cardinalities of global supports for the good candidates. The second plot is the difference these cardinalities with k_{suit} and the third plot is the sorted (ascending) version of the second plot. The first plot in the second row is the unsorted vector of residual values for the good candidates and the second plot is the sorted vector of residual values. The sorting of residual values is done in accordance with the sorted difference of cardinalities of global supports with k_{suit}. The highlighted point in the final plot is the selected candidate with first lowest residual.

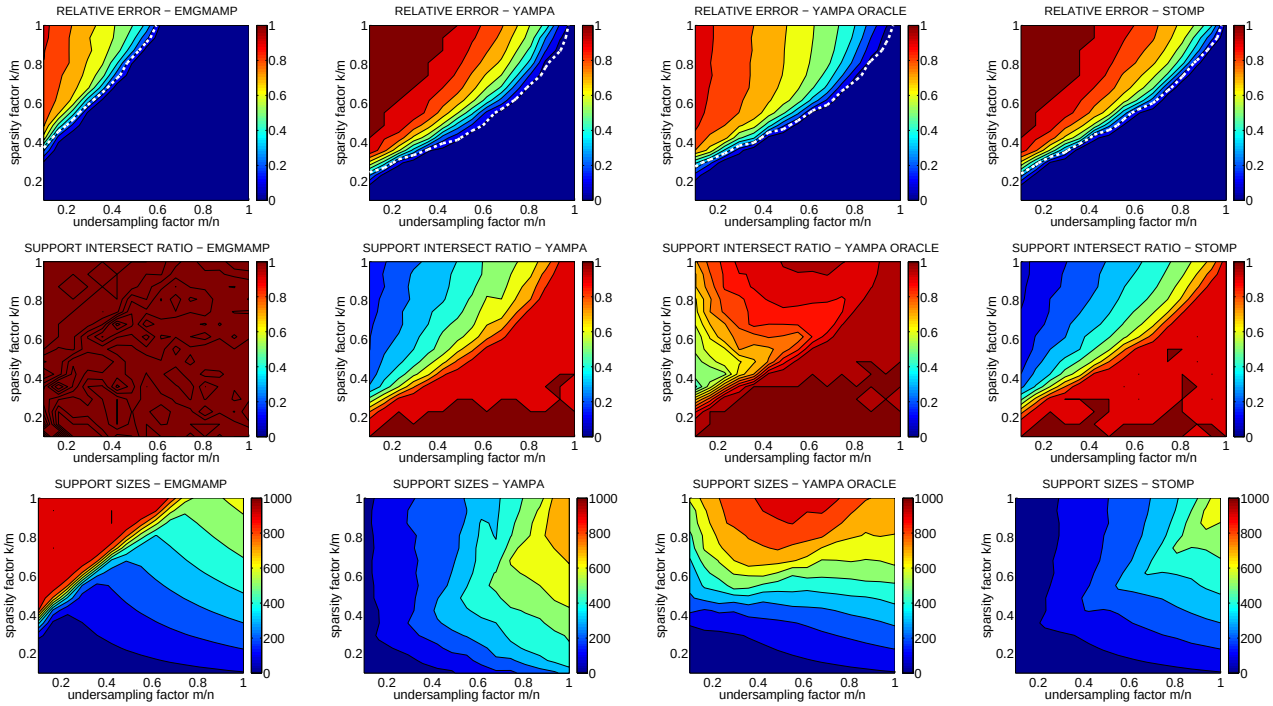

Figure 3. Figures for normal random matrices with $n = 1000$. The results are reported for EMGMAMP, StOMP, YAMPA with oracle and YAMPA with YCF. Each point in these plots is an average over 50 trials. The dotted white line in the relative error plots indicates the 50% success value boundary. The plots for other test signals look similar to these plots and are thus omitted. Moreover, binary random matrices show similar performance.

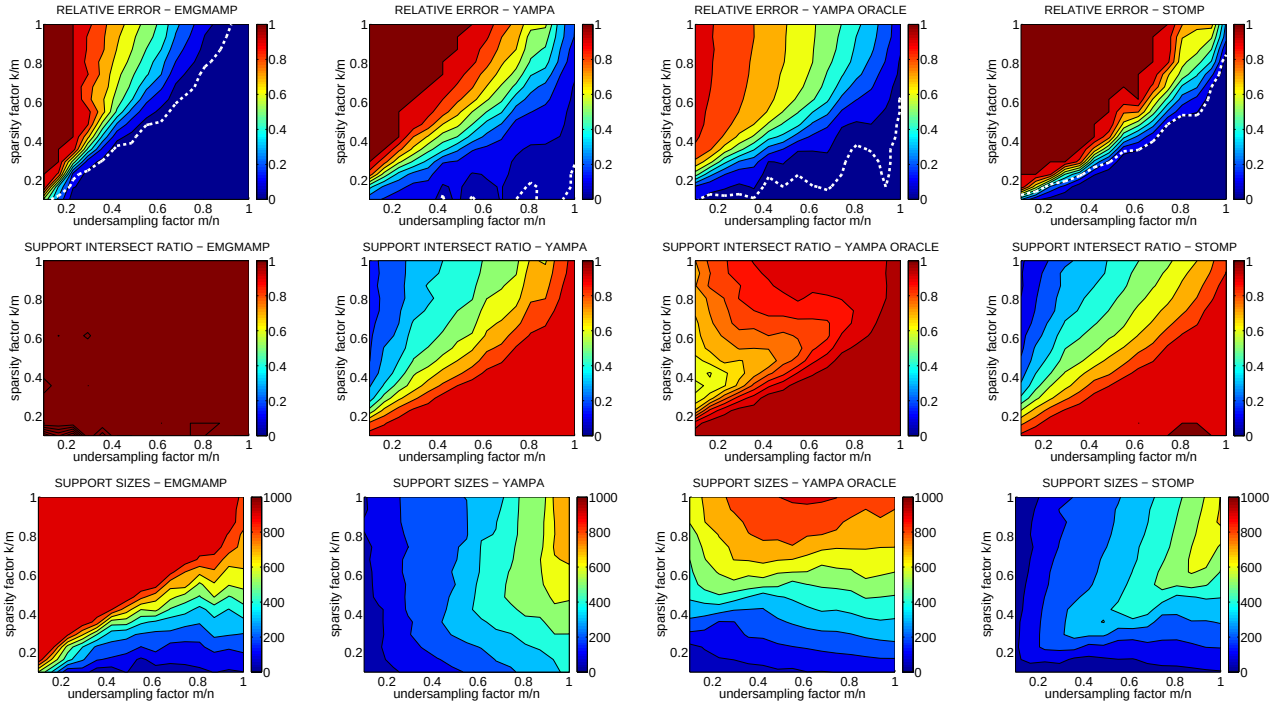

Figure 4. Figures for Cauchy matrices with $n = 1000$. The results are reported for EMGMAMP, StOMP, YAMPA with oracle and YAMPA with YCF. Each point in these plots is an average over 50 trials. The dotted white line in the relative error plots indicates the 50% success value boundary. The plots for other test signals look similar to these plots and are thus omitted.

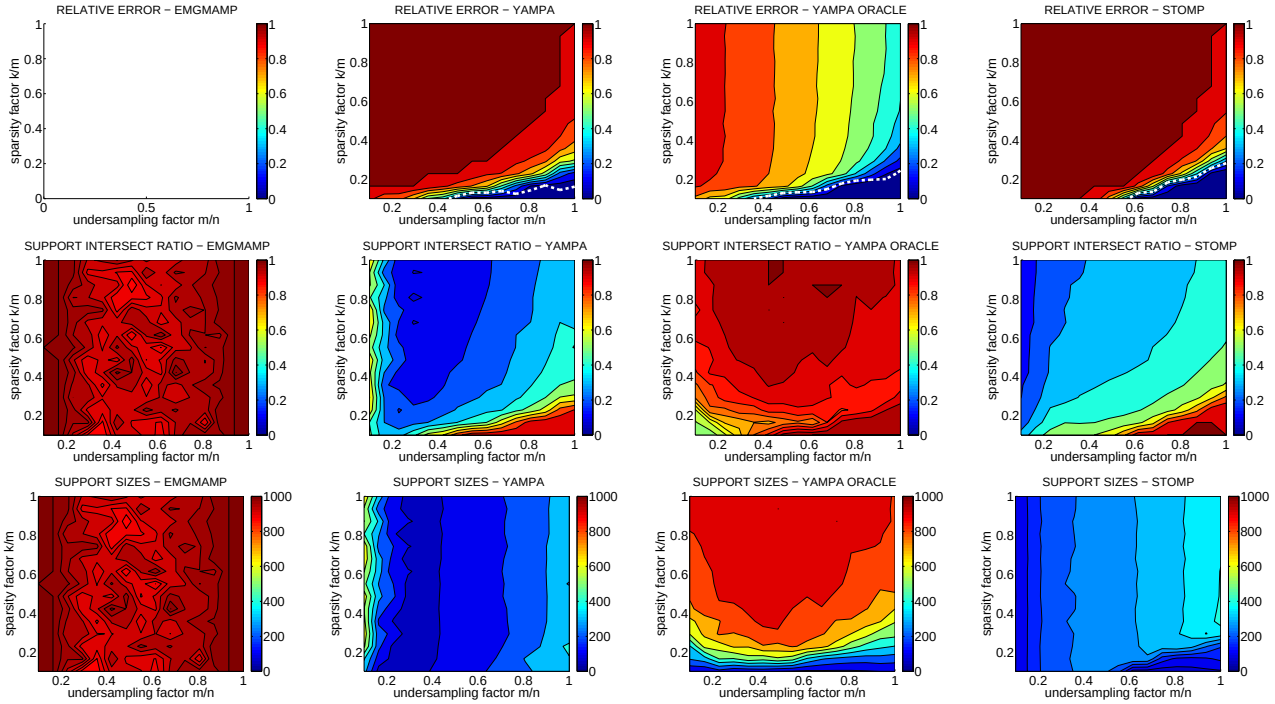

Figure 5. Figures for random ill-conditioned matrices with $n = 1000$, $s_{max} = 1$ and $s_{max} = 0.01$. The results are reported for EMGMAMP, StOMP, YAMPA with oracle and YAMPA with YCF. Each point in these plots is an average over 50 trials. The dotted white line in the relative error plots indicates the 50% success value boundary. The plot for relative error of EMGMAMP is all white because EMGMAMP fails completely and results in a mean relative error value 1 for all points. The plots for other test signals look similar to these plots and are thus omitted.

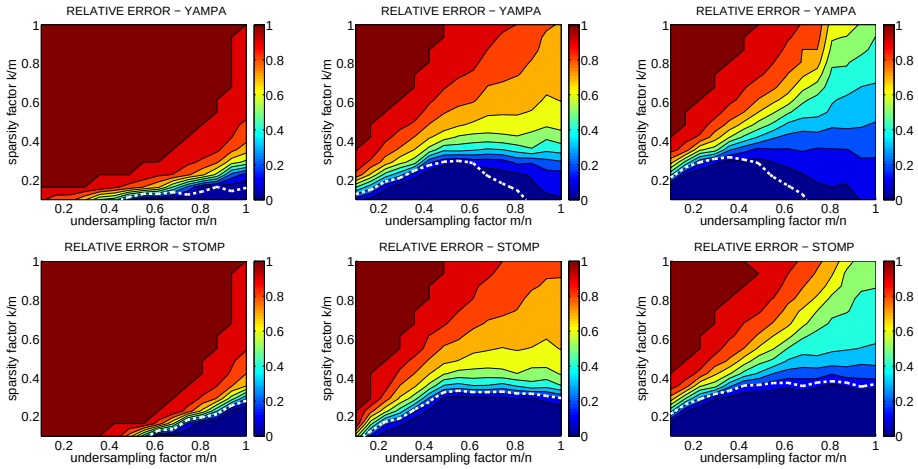

Figure 6. Figures for random ill-conditioned matrices for recovering uniform random signals with increasing number of correlated columns. The number of correlated columns are 0, 500 and 800 in the left, middle and right figures. The results are reported for StOMP and YAMPA with YCF. Each point in these plots is an average over 50 trials. The dotted white line in the relative error plots indicates the 50% success value boundary.

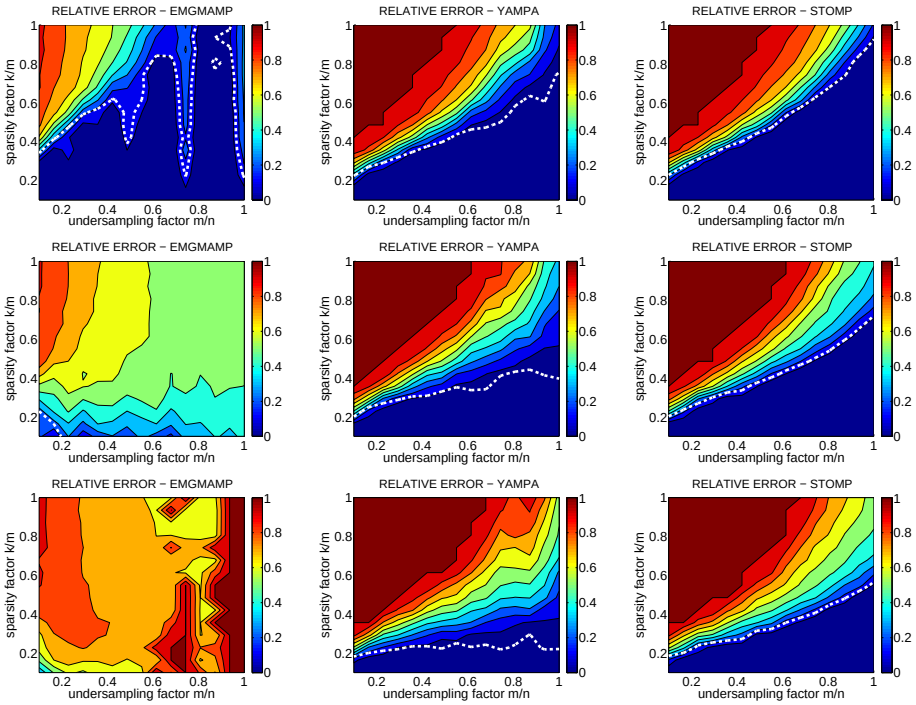

Figure 7. Figures for normal random matrices for recovering uniform random signals with increasing number of correlated columns. The number of correlated columns are 100, 300 and 500 in the first second and third figures. The results are reported for EMGMAMP, StOMP and YAMPA with YCF. Each point in these plots is an average over 50 trials. The dotted white line in the relative error plots indicates the 50% success value boundary.

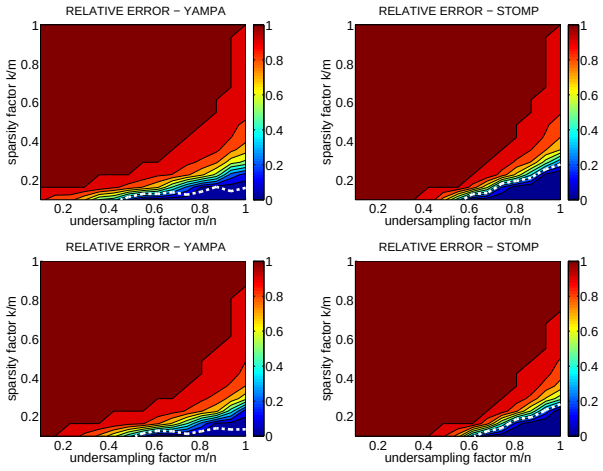

Figure 8. Figures for random ill-conditioned matrices for recovering uniform random signals with scaled maximum singular value. The lower figures are plots for matrices with maximum singular values that are 1.35 times the maximum singular value of the matrices in the upper figures. Each point in these plots is an average over 50 trials. The dotted white line in the relative error plots indicate the 50% success value boundary.

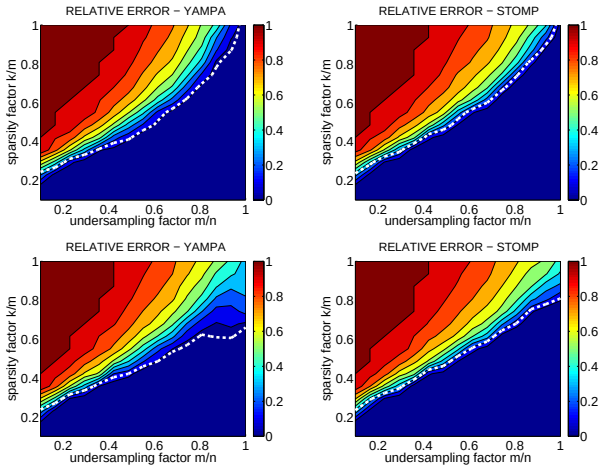

Figure 9. Figures for two different matrices for recovering uniform random signals with same condition number but scaled extreme singular values. The lower figures correspond to extreme singular values scaled by 1.3.

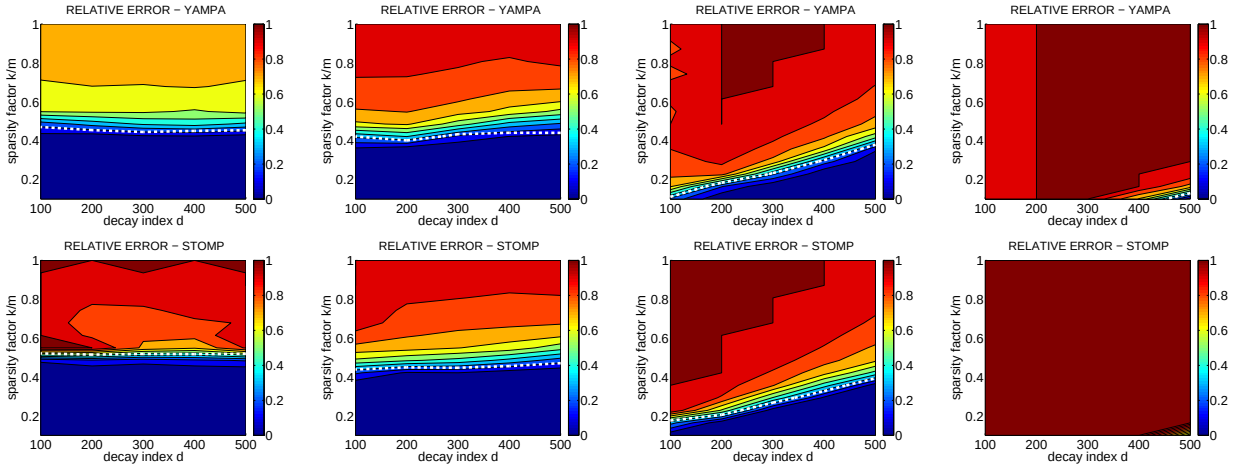

Figure 10. Plots for observing the effects of min singular value and the decay rate. For all figures $n = 1000$ and $m = 500$. The minimum singular values, from left to right, are 2, 1, 0.5, 0.05. The performance starts to degrade when the minimum singular value becomes small with a fast enough decay rate.

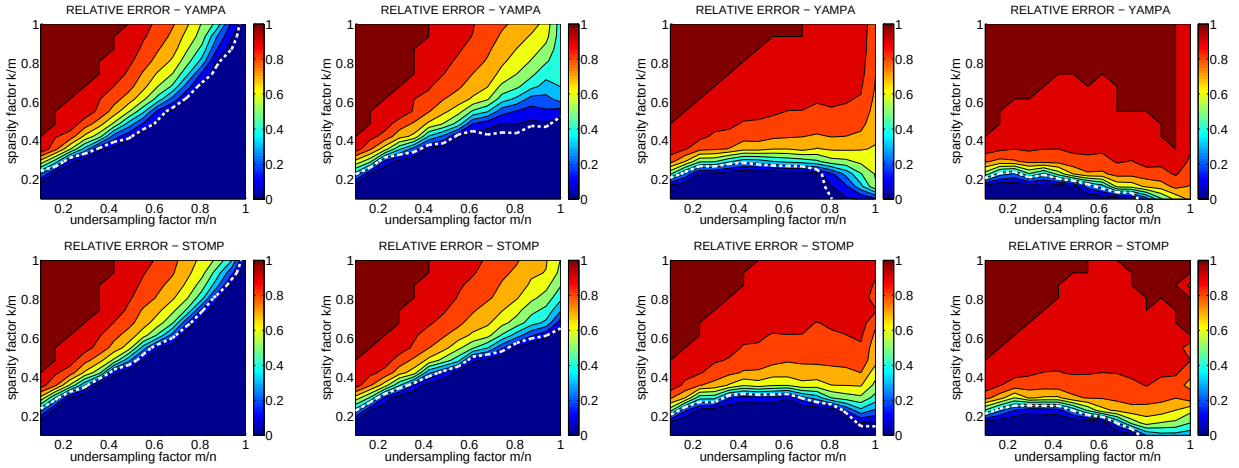

Figure 11. Figures for modified decays of singular values to demonstrate that the distribution of energy in the tail of singular values has considerable effect on the performance. The first figure (top left) is the reference figure. In the 2nd, 3rd and 4th figures (from left to right), the last 20%, 50% and 80% of singular values are set equal to the lowest singular value and the difference in energy is added to the rest of the singular values, thus keeping the extreme singular values and total energy in the matrices constant.

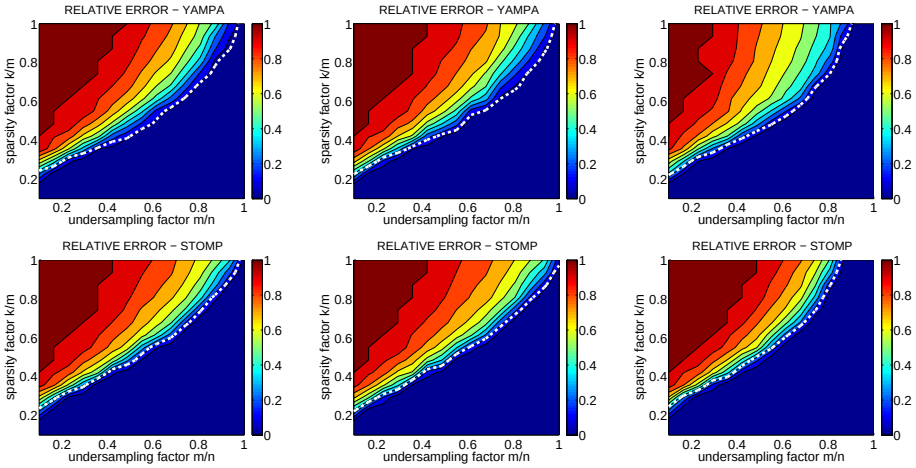

Figure 12. Figures for modified decays of singular values to demonstrate that the distribution of energy in singular values other than the tail has minor effect on sparse recovery performance. The first figure (top left) is the reference figure. In the 2nd figure (top middle) energy from the middle singular values (for d singular values, energy from indices $0.4d$ to $0.8d$) is reduced by a factor of $\frac{1}{5}$ and is redistributed to the remaining singular values. This has minor effect on the performance. In the 3rd figure (top right), the energy from upper singular values (for d singular values, energy from indices 3 to $0.2d$) is reduced by a factor of $\frac{1}{5}$ and is redistributed to the lower singular values. This slightly improves the performance for high values of undersampling factor $\frac{m}{n}$.