An Experimental Study on Compressive Sensing using Highly Coherent Matrices

Sreenithy Chandran

November 21, 2016

Abstract

Compressive sensing provides an efficient method to recover sparse high dimensional signals from low dimensional observations. One of the most important concern in compressive sensing is the high degree of correlation in sensing matrices. Sparse signal recovery has wide applications in source localization, radar detection etc. and in many of these applications, the dictionary matrix is highly coherent. This work numerically compares the efficiency of various sparse recovery algorithms in reconstructing the signal when the design matrix has highly correlated columns. This work also demonstrates the advantages of applying post processing techniques like SWAP algorithm to improve the performance of traditional algorithms like LASSO.

1 Introduction

Compressed sensing exploits sparsity or compressibility when acquiring signals. It is a non adaptive data acquisition method that uses far fewer measurements than traditionally assumed. Compressive Sensing has major applications in medical imaging, remote sensing, image and error correction [1] [2]. The key to compressive sensing is the possibility that many types of signals can be approximated by a sparse expansion. Compression is done by storing only the largest basis coefficients and when reconstructing the signal all the coefficients that was previously not stored a value, is set to zero. Compressive sensing provides a way to reconstruct a compressed version of the original signal by taking only a small amount of linear and non-adaptive measurements. In terms of compressive sensing, the interest is in the under sampled case, where there are fewer measurements than unknown signal values. But in an under sampled case the linear system describing the measurements is under determined and hence has infinitely many solutions. The main idea here is that the sparsity helps in isolating the original vector.

Compressed sensing suggests obtaining a signal $\mathbf{x} \in R^n$ by collecting m linear measurements of the form $\mathbf{y}_k =< \mathbf{a}_k, \mathbf{x} > +\mathbf{w}_k$, $1 \leq k \leq m$. This can be represented in matrix notation as

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

where $\mathbf{A}$ is an $m \times n$ sensing matrix with m usually smaller than n, $\mathbf{y}$ is the $n \times 1$ observation vector, $\mathbf{x}$ is the sparse vector that is to be estimated and $\mathbf{w}$ is an error term modeling measurement errors[3]. Throughout this report, we assume that the columns in $\mathbf{A}$ have unit l_2-norm. Further, the noise $\mathbf{w}$ is assumed to be Gaussian random vector with mean $\mathbf{0}_m$ and covariance $\sigma^2\mathbf{I}_m$. If the unknown signal $\mathbf{x}$ is sparse, it is possible to recover $\mathbf{x}$, under suitable conditions on the matrix $\mathbf{A}$. The optimal reconstruction algorithm is an exhaustive search that entails searching for the sparsest vector that is consistent with the linear measurements. This exhaustive search is unfortunately NP-hard. A number of computationally feasible techniques broadly belonging to two classes, *viz* convex relaxation techniques and greedy algorithms are discussed in literature.

In many applications such as neuro electromagnetic source localization, direction of arrival estimation, radar detection, under-water sonar processing, power spectrum estimation etc the ability of algorithms to handle the cases of highly coherent design matrices is very important (especially in the presence of noise). This work is an experimental study on compressive sensing with highly coherent matrices. The performance of various compressive algorithms are compared for a number of highly coherent dictionary matrices.

1.1 Notations used

In this paper the following notations are used . Bold lower case letters denote vectors and bold uppercase letters denotes matrices. $\mathbf{0}_n$ is a $n \times 1$ vector of all zeros and $\mathbf{I}_n$ is the $n \times n$ identity matrix. $||\mathbf{x}||_p = \left(\sum_{k=1}^{n} |\mathbf{x}_k|^p \right)^{\frac{1}{p}}$ is the l_p norm of $\mathbf{x}$.

2 Compressive sensing algorithms under consideration

2.1 LASSO

LASSO solves the optimization problem given by

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{z}} ||\mathbf{y} - \mathbf{A}z||_2^2 + \lambda||\mathbf{z}||_1 \tag{2}$$

It is one of the most popularly used sparse recovery algorithm. This is a convex quadratic program and then solved by several standard methods such as interior-point methods. In this study *l1_ls* a Matlab implementation of the interior-point method for *l1*-regularized least squares[4] is made use of. The code used for this study was downloaded at *https://stanford.edu/ boyd/l1_ls/*.

2.2 OMP

OMP is an iterative greedy algorithm that selects at each step the dictionary element best correlated with the residual part of the signal. Then it produces a new approximate by projecting the signal onto those elements which have already been selected [5]. Orthogonal Matching Pursuit (OMP) adds a least-squares minimization at each step to obtain the best approximation of the signal over the atoms which have already been chosen.
The OMP algorithm can be stated as follows:

1. Initialize the residual $\mathbf{r_0}$=$\mathbf{y}$ and initialize the set of selected variables $\mathbf{A}(c_0)$=ϕ. Let the iteration counter i=1.
2. Find the variable $\mathbf{A}_{t_i}$,that solves the following problem

   $$\max_t |\mathbf{A'}_t \mathbf{r}_{i-1}|$$

   and add the variable $\mathbf{A}_{ti}$ to the set of selected variables. Update $c_i = c_{i-1} \cup \{t_i\}$.
3. Let $\mathbf{P}_i = \mathbf{A}(c_i)(\mathbf{A}(c_i)'\mathbf{A}(c_i))^{-1}\mathbf{A}(c_i)'$ denote the projection onto the linear space spanned by the elements of $\mathbf{A}(c_i)$. Update $\mathbf{r}_i = (\mathbf{I} - \mathbf{P}_i)\mathbf{y}$.
4. If the stopping condition is achieved, stop the algorithm. Otherwise set i=i+1 and return to Step 2.

The package used in this study is available at
https://www.mathworks.com/matlabcentral/fileexchange/50584-orthogonal-matching-pursuit-algorithm–omp-

2.3 TMSBL

TMSBL exploits the correlation which exists in each non zero row of $\mathbf{x}$(temporal correlation)[6]. However it can be used for SMV models as well. Due to an effective learning rule to automatically choose the optimal regularization value, it gives appreciative performance than most compressed sensing algorithms for this model. Implementation of this algorithm does not require any priori information. The code used for this study was downloaded from
http://dsp.ucsd.edu/˜zhilin/TMSBL.html .

2.4 FBMP

Fast Bayesian Matching Pursuit (FBMP)[7] is an algorithm that performs Bayesian model averaging. FBMP models each unknown coefficient as either inactive or active (with prior probability p_1), where an i.i.d. Gaussian distribution with zero mean and variance σ^2,which is the variance of the non-zero taps. The observation $\mathbf{y}$ is then modeled as an AWGN-corrupted version of the unknown coefficients. FBMP navigates through the tree of active/inactive configurations with the goal of finding the configurations with dominant posterior probability It can be applied to a wide variety of compressive sensing and sparse reconstruction problems. The code utilized was taken from
http://www2.ece.ohiostate.edu/ schniter/FBMP/download.html.

2.5 BCS

The basic BCS algorithm adopts the relevance vector machine (RVM) [8]. In Bayesian modeling, all unknowns are treated as stochastic quantities with assigned probability distributions. The unknown signal $\mathbf{x}$ is assigned a prior distribution $p(\mathbf{x}|\gamma)$, which models our knowledge on the nature of $\mathbf{x}$ [9] . The observation $\mathbf{y}$ is also a random process with conditional distribution $p(\mathbf{y}|\mathbf{x}, \beta)$ where $\beta = 1/\sigma^2$. σ^2 is the noise variance. These distributions depend on the model parameters δ and β, which are called hyperparameters, and additional prior distributions, called hyperpriors, are assigned to them. Bayesian compressive sensing code was obtained from
http://people.ee.duke.edu/ lcarin/BCS.html.

2.6 ExCoV

The basic idea of ExCoV is to interleave the expansion and compression steps that modify the estimate of the index set by one element per step. [10]. ExCoV aims at maximizing the generalized maximum likelihood objective function and provides an approximate generalized maximum likelihood estimate of the significant signal element set and an empirical Bayesian signal estimate. The ExCoV method is automatic and demands no prior knowledge about signal-sparsity or measurement-noise levels. The package used in this study was obtained from *http://home.eng.iastate.edu/ ald/ExCoV.htm*

2.7 SWAP

SWAP is used to post-process the output of any sparse regression algorithm without compromising on the performance. SWAP is used to estimate the true support of $\mathbf{x}$, even when the support obtained from the regression algorithm is widely different from the actual signal. This is a greedy algorithm that iteratively swaps variables starting from an initial estimate of $\mathbf{x}$ until a desired loss function cannot be decreased any further. This algorithm is used only when conventional sparse recovery algorithms fail to recover $\mathcal{I}$. This motivates the use of SWAP as a wrapper around sparse recovery algorithms for improved performance[11] [12]. Once $\mathcal{I}$ has been estimated, it is relatively straightforward to estimate $\mathbf{x}$.

The SWAP algorithm is given below:
Input: Measurements $\mathbf{y}$, design matrix $\mathbf{A}$, and initial support $\mathbf{S}$.

1. Let r=1, $S^{(1)}$ = S and $L^{(1)}$=$\mathcal{L}(S^{(1)}; \mathbf{y},\ \mathbf{A})$.
   For a support S, the least square loss is defined as

$$\mathcal{L}(S; \mathbf{y}, \mathbf{A}) = \min_{\alpha \in R^{|S|}} ||\mathbf{y} - \mathbf{A}_S \alpha||_2^2.$$

   where $\mathbf{A}_S$ matrix that only includes the columns indexed by S.

2. Swap $i \in S^{(r)}$ with $i' \in (S^{(r)})^c$
   Compute the loss $L_{i,i'}^{(r)} = L(\{S^{(r)} \backslash i\} \cup i'; \mathbf{y}, \mathbf{A})$.

3. If $min_{i,i'} \mathcal{L}_{i,i'}^{(r)} < L^{(r)} then$

4. $\{\hat{i}, \hat{i'}\} = \arg\min_{i,i'} \mathcal{L}_{i,i'}^{(r)}$

5. Let $S^{(r+1)} = \{S^{(r)} \backslash \hat{i'}\} \cup \hat{i'}$ *and* $L^{(r+1)}$ be the corresponding loss.

6. Let r=r+1 and repeat steps 2-4

7. Else return $\hat{S} = S^{(r)}$.

The code used in this study was taken from *http://dsp.rice.edu/software/swap*

3 Matrix Coherence

Conventionally it is assumed in the field of compressed sensing and sparse signal recovery, that the measurement matrix have uncorrelated columns. The coherence of a matrix A is defined as

$$\mu(A) = \max_{j \neq k} | < A_j, A_k > |$$

where A_j and A_k denote columns of $\mathbf{A}$. The dictionary is incoherent if μ is small. The Restricted Isometry Property and other results enforce a strict incoherence property [5] for the measurement matrix. If two columns are closely correlated, it will be impossible in general to distinguish where the energy in the signal comes from.

4 Experimental Results

In this section, experimental results for the performance of the five recovery algorithms are compared with the performance results obtained when the SWAP algorithm was used to post process the output of the sparse recovery algorithms. The performance was evaluated for five different dictionary matrices for varying values of sparsity and positions of the non-zero values of $\mathbf{x}$. Performance of the various algorithms is evaluated in terms of the probability of error (Pe). A failed trial was recognized when the indexes of the estimated signal with the D largest amplitude were not the same as the true indexes[13].

4.1 Dictionary Matrix 1

This is a highly coherent dictionary that is generated from random matrices. The dictionary is constructed as follows:

1. $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}_{10}, \mathbf{I_{10}})$

2. $\mathbf{y}_{ij} \sim \mathcal{N}(\mathbf{0}_{10}, \sigma\mathbf{e^2}\mathbf{I_{10}})$ for $1 \leq i \leq 4$ and $1 \leq j \leq 5$, and σe^2 = [0.01, 0.03, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8, 1, 2].

3. $\mathbf{x}_{ij} = \mathbf{z}_i + \mathbf{y}_{ij}$ for all values of *i,j*.

Then the dictionary matrix is given as

$$A = [\mathbf{x}_{11}\ \mathbf{x}_{12}\ \mathbf{x}_{13}\ \mathbf{x}_{14}\ \mathbf{x}_{15}\ \mathbf{x}_{21}\ \mathbf{x}_{22}\ \mathbf{x}_{23}\ \mathbf{x}_{24}\ \mathbf{x}_{25}\ \mathbf{x}_{31}\ \mathbf{x}_{32}\ \mathbf{x}_{33}\ \mathbf{x}_{34}\ \mathbf{x}_{35}\ \mathbf{x}_{41}\ \mathbf{x}_{42}\ \mathbf{x}_{43}\ \mathbf{x}_{44}\ \mathbf{x}_{45}]$$

The dictionary generated is of dimension 10x20. Table 1 contains the values of the σe^2 and the corresponding coherence of the matrix, which was averaged over 1000 iterations.

σe^2	Average coherence of dictionary
0.01	0.9979
0.03	0.9939
0.05	0.9899
0.1	0.9807
0.2	0.9654
0.4	0.9392
0.6	0.9178
0.8	0.8984
1	0.8866
2	0.8467

Table 1: Table showing the variation of average coherence of dictionary matrix **A** as the value of σe^2 varies.

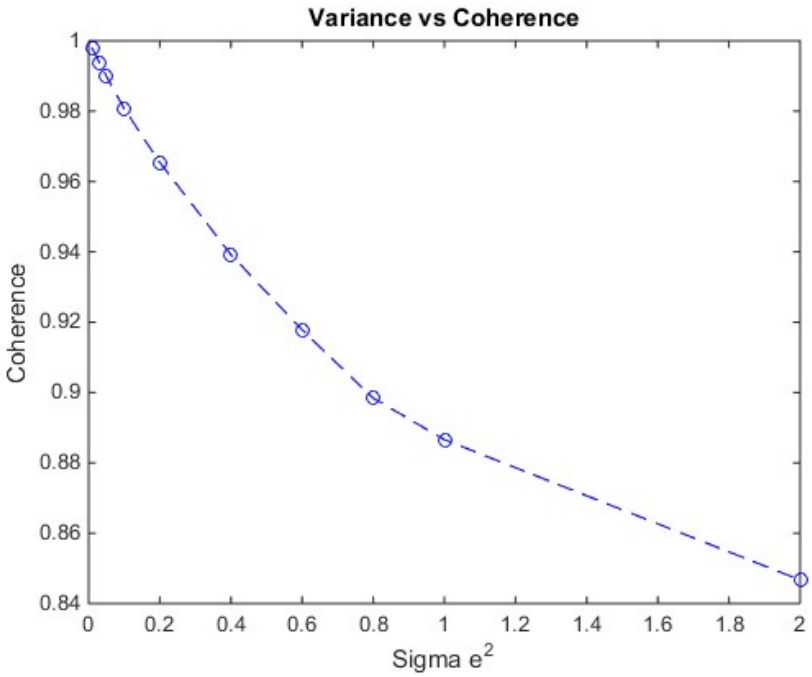


Figure 1: Plot showing variance versus coherence.

A plot of the variation between σe^2 and coherence of the matrix is shown in Figure 1. σe^2 is referred to as Sigma e^2 in the graphs. As seen from the plot above when the value of the σe^2 decreases and approaches 0, the coherence between the columns of the matrix increases. When σe^2 =0.01, value of average coherence approaches 0.9979. And when the σe^2 is equal to 2 the coherence is 0.8467. The experiment is carried out for 10 different values of σe^2 and averaged over for 1000 iteration for each value of the σe^2. In each trial the number of non zero elements in the signal were equal to 4. The location of the indexes were either:

1. Randomly chosen
2. Fixed at the 1st, 2nd, 3rd and 4th position of the signal.
3. Fixed at the 1st, 7th, 13th and 19th position of the signal.

The noise vector $\mathbf{w}$ was generated as a Gaussian vector such that the SNR was 20dB. When $\mathcal{I} = [1, 2, 3, 4]$ then the non-zero values in $\mathbf{x}$correspond to highly correlated columns of the dictionary matrix. And when the $\mathcal{I} = [1, 7, 13, 19]$ then the non-zero values correspond to least correlated columns of the $\mathbf{A}$.

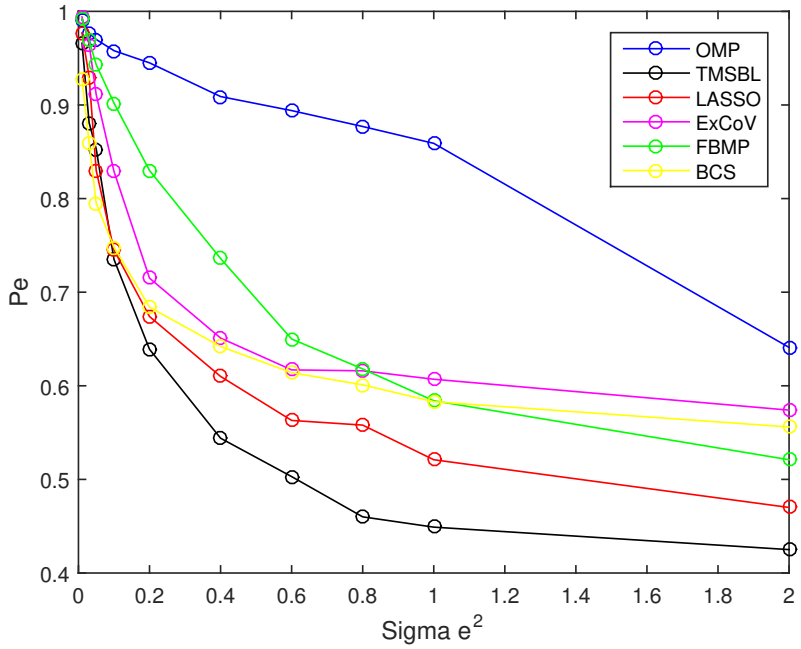


Figure 2: The plot of probability of error and Sigma e^2 when the locations of the nonzero values is randomly generated. The magnitude of the nonzero values varies between -0.5 and 0.5, sparsity of the signal is D=4. The dimension of the sensing matrix is 10x20.

Thus when the four non zero locations in the vector $\mathbf{x}$ is randomly chosen, TMSBL gives a better performance compared to the other algorithms. As the value of σe^2 increases i.e., when the coherence decreases, TMSBL gives the lowest probability of error, which approaches to 0.4 when the coherence is 0.85.

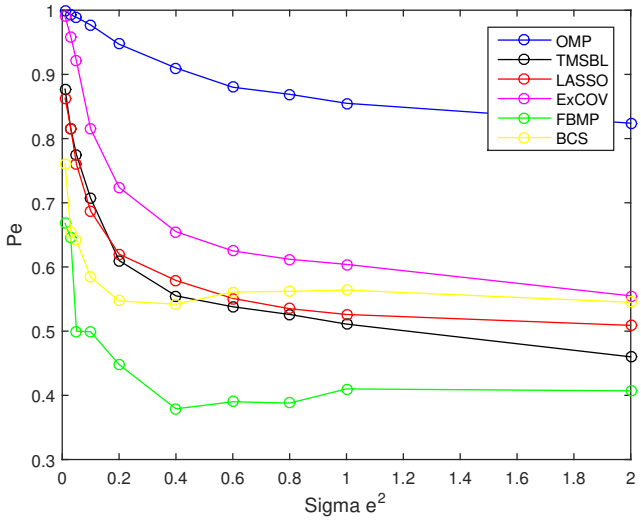


Figure 3: Plot showing performance of various algorithm as the coherence of the dictionary changes when the non-zero values are clustered together, i.e., they correspond to highly correlated columns of the dictionary.

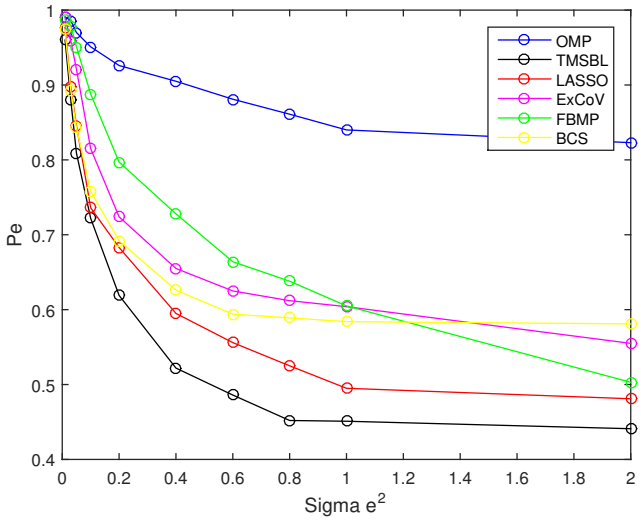


Figure 4: Plot showing performance of various algorithm as the coherence of the dictionary changes when the non-zero values are distributed, i.e., they correspond to least correlated columns of the dictionary.

As evident from the plot the TMSBL algorithm resulted in the least probability of error for the highly coherent matrix. Also when the coherence of the matrix decreases TMSBL gives better performance than the other algorithms.

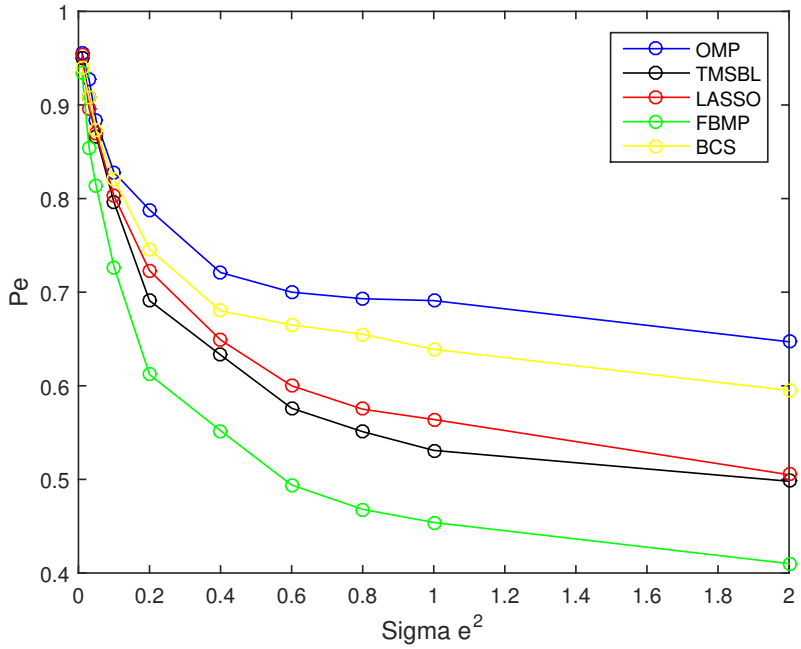
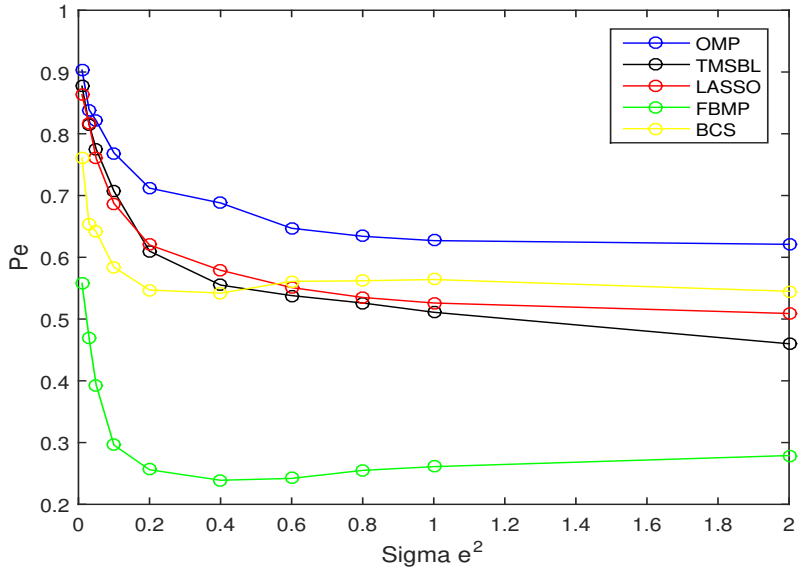
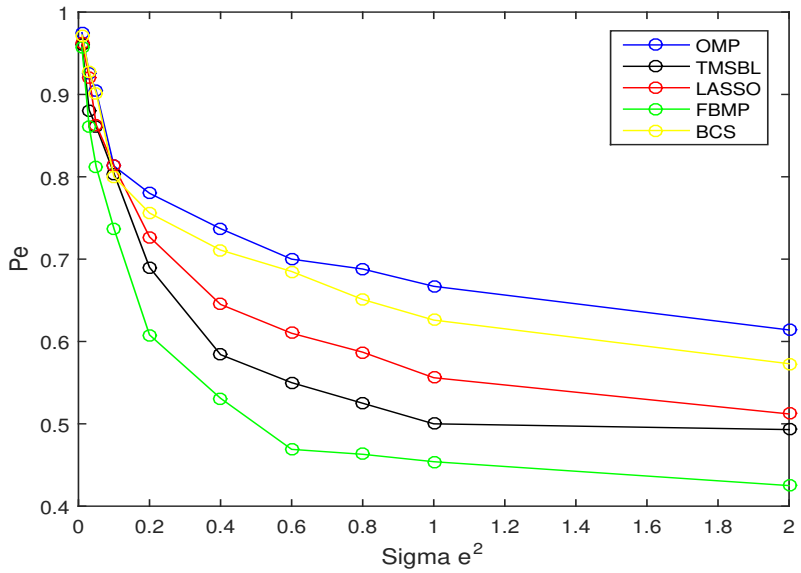


Figure 5: The plot of probability of error and Sigma e^2 when the locations of the nonzero values is randomly generated and SWAP is used to post-process the result of the sparse recovery algorithms. The magnitude of the nonzero values varies between -0.5 and 0.5, sparsity of the signal is D=4. The dimension of the sensing matrix is 10x20

In the Figure 5, the SWAP algorithm is used to post process the output of the sparse recovery algorithm. It is observed that the FBMP gives the best performance compared with the other algorithms. And the maximum improvement in performance by using SWAP is also for the FBMP algorithm.
Next we consider the cases when the positions of the non-zero is clustered and when it is distributed across the matrix. As observed from Figure 6, the FBMP algorithm performed the best among the other algorithms considered. When the active elements are present in locations corresponding to the least correlated columns, the lesser correlation helps to better identify the location of the elements. This is in contrast to the case when the active elements are present in locations corresponding to highly correlated columns of the dictionary. This makes it difficult to distinguish the position of the non-zero values. Figure 6.a is the performance plot when the non-zero elements are separated apart and Figure 6.b is the performance plot when the non-zero elements are clustered together. The probability of error Pe is lower for all values of σe^2 when the non-zero positions are present in least correlated columns. Thus clearly the coherence of the matrix and the location of the non-zero element affect the recovery performance of an algorithm.



(a) Performance when the $\mathcal{I} = [1, 7, 13, 19]$



(b) Performance when the $\mathcal{I} = [1, 2, 3, 4]$

Figure 6: Plot showing performance of various sparse recovery algorithms when the results of the algorithms are post-processed using SWAP for two different positions of the non-zero values

When the recovery algorithms were implemented along with SWAP it is observed that the FBMP algorithm gives a better performance than the other algorithms compared, for the previously mentioned three different positions of the non-zero indexes.

4.2 Dictionary Matrix 2

The second dictionary matrix used for the experiment is generated similar to the first one. The dictionary matrix is constructed as follows:

1. Vector $\mathbf{b} \sim \mathcal{N}(\mathbf{0}_{10}, \mathbf{I_{10}})$

2. $\mathbf{a}_{ij} \sim \mathbf{b} + \mathcal{N}(\mathbf{0}_{10}, \sigma e^2 \mathbf{I_{10}})$ for $1 \leq i \leq 4$ and $1 \leq j \leq 5$, and σe^2 = [0.01 0.03 0.05 0.1 0.2 0.4 0.6 0.8 1 2].

3. Matrix $\mathbf{A}_i$=[$\mathbf{a}_{i1}$ $\mathbf{a}_{i2}$ $\mathbf{a}_{i3}$ $\mathbf{a}_{i4}$ $\mathbf{a}_{i5}$] for $1 \leq i \leq 4$.

The dictionary matrix is then given as $\mathbf{A} = [\mathbf{A}_1\ \mathbf{A}_2\ \mathbf{A}_3\ \mathbf{A}_4]$. The matrix obtained has very high correlation between two columns of the matrix. As the value of σe^2 becomes close to zero the average coherence increases. The tabulation of variation between σe^2 and coherence is shown in the Table 2. And the plot of σe^2 and the average coherence is shown in the Figure 7.

Sigma e^2	Average coherence of dictionary
0.001	0.9998
0.005	0.9990
0.01	0.9980
0.05	0.9901
0.1	0.9817

Table 2: Table showing the variation of average coherence of dictionary matrix A as the value of σe^2 varies.

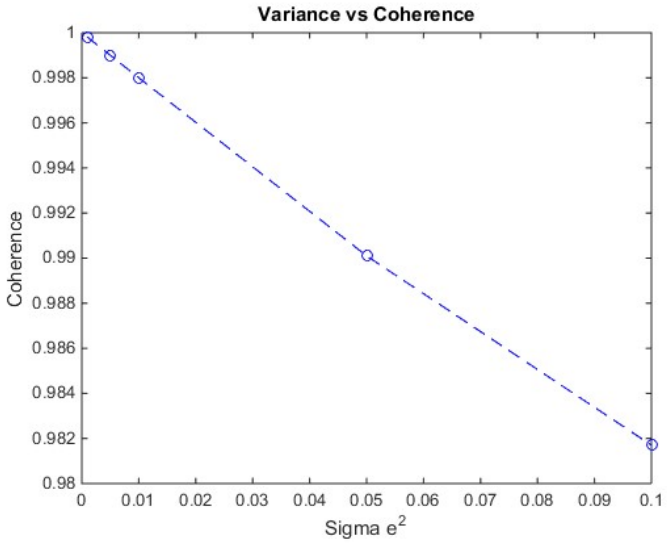


Figure 7: Plot of variance versus coherence.

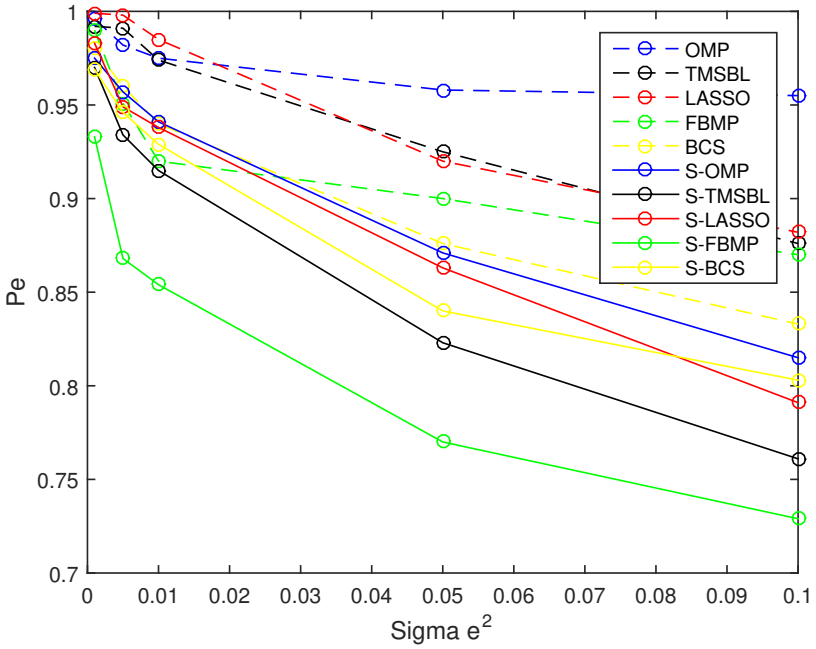
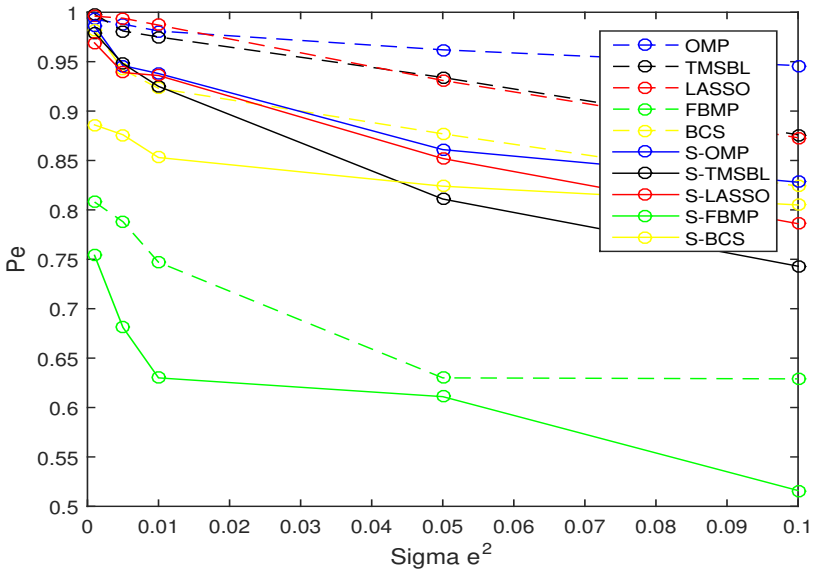
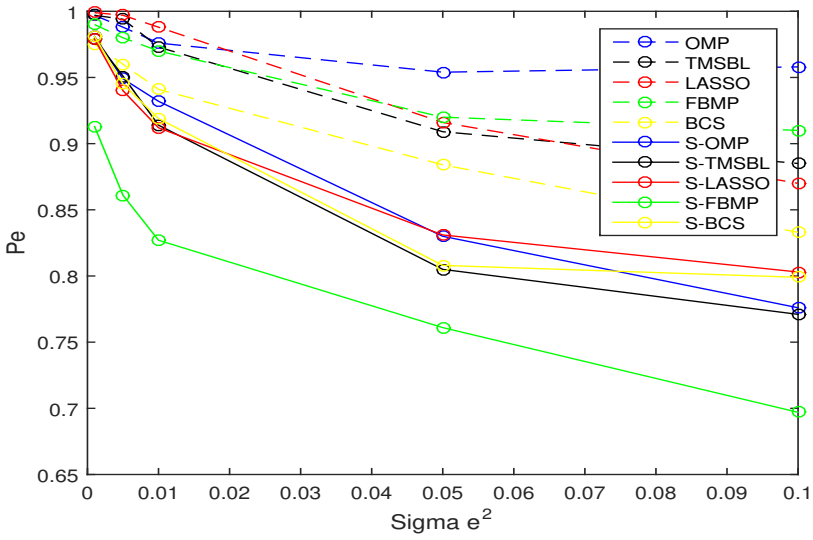


Figure 8: Plot of Sigma e^2 and probability of error for very highly coherent dictionary when the non-zero indexes are randomly generated.

From this dictionary onward the results of performance for the ExCOV algorithm is not reported since the Pe values were very high to be of any practical significance. The plot of Sigma e^2 and probability of error is shown in Figure 8. The performance of the algorithm when followed by SWAP is plotted by solid lines and when the recovery is performed without the SWAP it is plotted with dotted lines. As evident FBMP followed by SWAP implementation gives the best performance. When the coherence=0.9998(maximum) FBMP gives a Pe=0.9330 followed by TMSBL which results in Pe=0.9690 and BCS=0.9700 . And when the coherence decreases considerably FBMP followed by SWAP gives the best performance. Also, as seen from comparing the plots of the dotted and solid line, the Pe value has decreased considerably for all values of Sigma e^2 and for all algorithms when SWAP algorithm is used as a post processing techniques.



(a) The non zero locations of the signal are distributed at positions $[1, 2, 3, 4]$



(b) the indexes of nonzero values are at $[1, 7, 13, 19]$

Figure 9: Plot of Sigma e^2 and Probability of error for very highly coherent dictionary. Performance plots when recovery is performed with and without using SWAP algorithm

As evident FBMP followed by SWAP implementation gives the best performance in both the situations when the non-zero locations are distributed throughout the matrix (i.e) at position [1, 7, 13, 19] and when they are in the same block (i.e) at positions [1, 2, 3, 4]. Also an overall reduction in Pe is obtained when the SWAP algorithm is employed. At positions [1, 2, 3, 4]

the FBMP algorithm resulted in consistent better performance the the other algorithms considered. BCS performed better recovery than TMSBL at high coherence of the dictionary for support vector [1, 2, 3, 4].
There was an observed increase in performance for all the algorithms when implemented along with SWAP. But the combination of FBMP followed by SWAP proved to give the least probability of error, i.e., the best performance among all the algorithms for this study for the above considered dictionary matrix.

4.3 Dictionary Matrix 3

Consider an oversampled DCT matrix $A = [a_1, ..., a_N] \in R^{M \times N}$ with

$$a_j = \frac{1}{\sqrt{N}} \cos \frac{2\pi w j}{F}, j = 1, ..., N;$$

where w is a random vector of length M. The matrix is highly coherent. For a $100 \times 1,000$ matrix with F =10, the coherence is 0.9981, while the coherence of a same size matrix with F=20 is 0.9999. The sparse recovery under such matrices is possible only if the non-zero elements of $\mathbf{x}$ are sufficiently separated. This phenomenon is characterized as minimum separation [14], and this minimum length is referred to as the Rayleigh length (RL). The value of RL is equal to F. The larger F corresponds to larger coherence of a matrix. It can be empirically found that at least 2RL is necessary to ensure optimal sparse recovery. Intuitively, sparse spikes need to be further apart for more coherent matrices. Table 3 shows the variation of the coherence of the matrix as the value of F changes. Under the assumption of sparse signal with 2RL separated spikes, we compare algorithms in terms of the probability of error as described previously. The experiment was carried out for F=5 and dimension of A is 51×501 and the results were averaged over 1000 iteration. Additionally as in the previous cases, the evaluation is carried out for the following positions of non-zero values of $\mathbf{x}$:

1. When the locations of the nonzero values of $\mathbf{x}$ correspond to least correlated columns of the dictionary.
2. When the locations of the nonzero values of $\mathbf{x}$ correspond to highly correlated columns of the dictionary.
3. When the locations of the nonzero values of $\mathbf{x}$ are randomly chosen.

F	Coherence
1	0.43579
5	0.97378
10	0.99842
15	0.99962
20	0.99989

Table 3: Table showing the variation of coherence of the dictionary with changes in value of F.

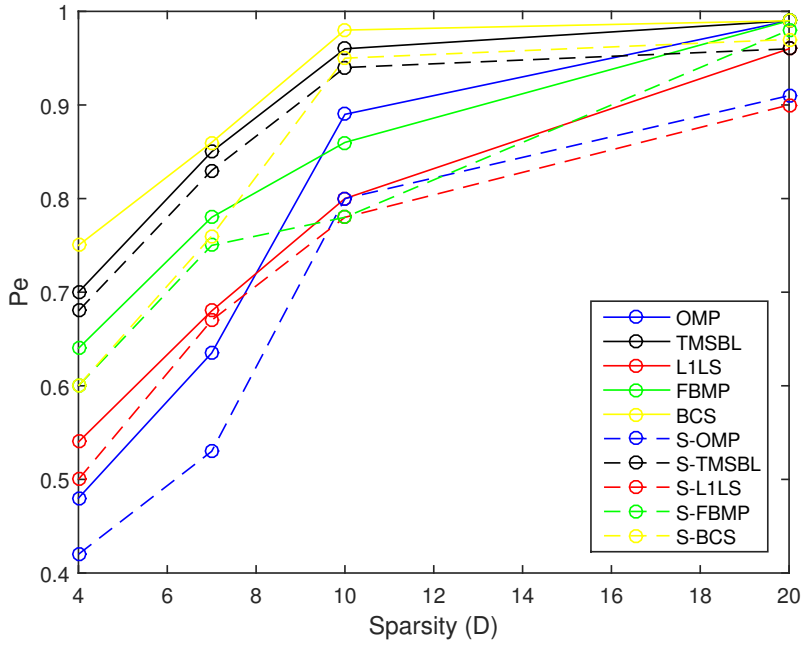


Figure 10: Plot showing the variation of the Pe as the sparsity value changes when F=5. The performance using sparse recovery algorithms is shown by solid lines. The performance using the sparse recovery algorithms and the results of which are post-processed using SWAP is shown by dotted lines. The non-zero elements in $\mathbf{x}$ corresponded to the least correlated columns of $\mathbf{A}$

.

In this evaluation the positions of non-zero values of $\mathbf{x}$ was chosen such that they corresponded to the least correlated columns of $\mathbf{A}$. The positions of the non-zero values are tabulated as below in Table 4. The locations were chosen by observing the correlation between the columns of the dictionary matrix $\mathbf{A}$. As observed from Figure 10, Orthogonal Matching Pursuit and LASSO performed very efficient recovery of the signal $\mathbf{x}$ in this case.

Sparsity (D)	$\mathcal{I}$
4	[1, 15, 203, 488]
7	[1, 11, 99, 165, 298, 466, 488]
10	[1, 4, 15, 89, 156, 325, 365, 425, 468, 498]
20	[1, 5, 25, 36, 41, 78, 100, 135, 165, 199, 225, 256, 298, 321, 365, 389, 421, 455, 489, 499]

Table 4: Table showing the $\mathcal{I}$ for different sparsity levels.

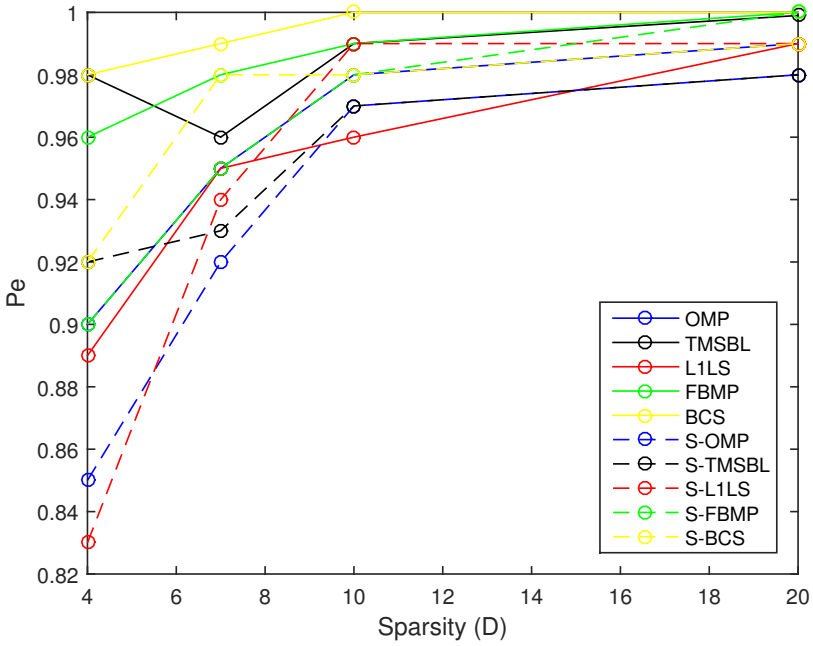


Figure 11: Plot showing the variation of the Pe as the sparsity value changes when F=5. The performance using sparse recovery algorithms is shown by solid lines. The performance using the sparse recovery algorithms and the results of which are post-processed using SWAP is shown by dotted lines.The non-zero elements in $\mathbf{x}$ corresponded to the highly correlated columns of $\mathbf{A}$

.

The Figure 11 shows the results when the active elements in $\mathbf{x}$ correspond to highly correlated columns in the dictionary matrix $\mathbf{A}$. As seen from the plot the probability of error has increased for all the algorithms. Moreover when the sparsity value increases, the Pe also increases. But however,a better performance was obtained when SWAP was used as a post-processing step.

Sparsity (D)	$\mathcal{I}$
4	[1, 2, 3, 4]
7	[240, 241, 242, 243, 244, 245, 246]
10	[180, 181, 182, 183, 184, 200, 201, 202, 203, 204]
20	[120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139]

Table 5: Table showing the $\mathcal{I}$ for different values of D.

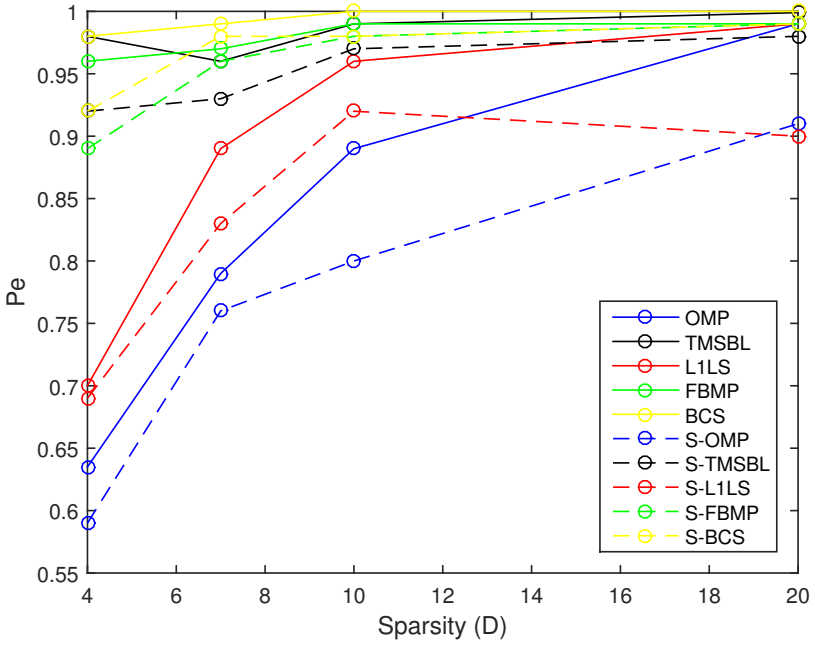


Figure 12: Plot showing the variation of the Pe as the sparsity value changes when F=5. The performance using sparse recovery algorithms is shown by solid lines. The performance using the sparse recovery algorithms and the results of which are post-processed using SWAP is shown by dotted lines.The non-zero elements in $\mathbf{x}$ are randomly chosen

The Figure 12 shows the results when the active elements in $\mathbf{x}$ are randomly chosen. Besides OMP and LASSO all the algorithms considered in the evaluation yielded very poor performance. And when the sparsity value increases, the probability of error also increases. When D=4 the best performance was given by OMP followed by SWAP where Pe=0.58, and when the D=20, the best performance was again given by LASSO followed by SWAP where Pe=0.90

4.4 Dictionary Matrix 4

The next matrix considered is an over sampled Discrete Fourier Transform (DFT) matrix. The DFT matrix is an matrix whose kth column is given by

$$d_k(t) = \frac{1}{\sqrt{n}} \exp^{-2\pi ikt/n}$$

where $0 \leq t \leq n-1$ and $0 \leq k \leq n-1$. In an over sampled DFT the sampled frequencies are taken over smaller equally spaced intervals, or at small intervals of varying lengths. This leads to an over complete frame whose columns may be highly correlated.

ALGORITHM	Pe (Probability of Error)	Pe_s (Probability of Error with SWAP)	Location of non-zero values in x
TMSBL	0.9910	0.9650	Randomly Generated
	0.9970	0.9840	Correlated columns
	0.5500	0.5000	Uncorrelated Columns
BCS	0.9780	0.9650	Randomly Generated
	0.9920	0.9890	Correlated columns
	0.8600	0.8430	Uncorrelated Columns
LASSO	0.9890	0.9580	Randomly Generated
	0.9930	0.9840	Correlated columns
	0.4680	0.3810	Uncorrelated Columns
OMP	0.9800	0.9860	Randomly Generated
	0.9980	0.9870	Correlated columns
	0.4560	0.4300	Uncorrelated Columns
FBMP	0.9950	0.9730	Randomly Generated
	0.9999	0.9810	Correlated columns
	0.3510	0.3370	Uncorrelated Columns

Table 6: Table showing the performance of various algorithms in terms of the probability of error when number of non-zero values in $\mathbf{x}$ is equal to 4.

The dictionary matrix considered in this study is of dimension 20×100 and the average coherence of the matrix is 0.9987. $\mathcal{I}$=[1, 2, 3, 4] for the non-zero values in $\mathbf{x}$ to correspond to the highly correlated columns of $\mathbf{A}$. And $\mathcal{I}$=[1, 9, 15, 33] for the non-zero values in $\mathbf{x}$ to correspond to the least correlated columns of $\mathbf{A}$. $\mathcal{I}$ was chosen upon inspecting the correlation among the columns of $\mathbf{A}$. From Table 6 it is clear that the when the non-zero values in $\mathbf{x}$ are present in positions corresponding to uncorrelated columns of the dictionary matrix, the Pe value was the least. And when the non-zero values in $\mathbf{x}$ are present in positions corresponding to highly correlated columns of $\mathbf{A}$, the Pe value was higher. Thus when the correlation between the columns is low, the sparse recovery algorithms resulted in a better performance than when the columns were highly correlated. OMP resulted in the best performance among the algorithms considered. The probability of error also decreases when the SWAP was used to post-process the results obtained from the sparse recovery algorithm.

ALGORITHM	Pe (Probability of Error)	Pe_s (Probability of Error with SWAP)	Location of non-zero values in x
	0.9750	0.9600	Randomly Generated
TMSBL	0.9970	0.9840	Correlated columns
	0.6500	0.6200	Uncorrelated Columns
	0.9800	0.9790	Randomly Generated
BCS	0.9990	0.9800	Correlated columns
	0.8900	0.8500	Uncorrelated Columns
	0.9890	0.9580	Randomly Generated
LASSO	0.9930	0.9840	Correlated columns
	0.7900	0.7100	Uncorrelated Columns
	0.9800	0.9860	Randomly Generated
OMP	0.9980	0.9870	Correlated columns
	0.6900	0.6800	Uncorrelated Columns
	0.9980	0.9830	Randomly Generated
FBMP	0.9999	0.9810	Correlated columns
	0.7890	0.7800	Uncorrelated Columns

Table 7: Table containing the performance of various algorithms in terms of the probability of error when D=12.

Table 7 contains the probability of error for the various algorithms when the sparsity value of the $\mathbf{x}$ is kept at 12. The value of Pe is higher for all the recovery algorithms in comparison to the results obtained when the sparsity level was kept at 4. Thus, when the sparsity of the vector increases the probability of error increases correspondingly.
$\mathcal{I}$=[84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95] for high correlation between the columns and $\mathcal{I}$=[1, 2, 7, 15, 22, 27, 39, 44, 51, 63, 79, 93] for low correlation among the columns are chosen on inspecting the correlation among the columns of $\mathbf{A}$. As observed when the correlation between the columns is low, the sparse recovery algorithms resulted in a better performance than when the columns were highly correlated. OMP, TMSBL, LASSO resulted in appreciable performance. Also using SWAP to post-process the results of the sparse algorithm does not yield appreciative improvement in the performance.

4.5 Dictionary Matrix 5

Now the algorithms were evaluated on the gene expression data set taken from *http://www.biolab.si/supp/bi-cancer/projections/info/leukemia.htm.* The data

set contains gene expression values from patients with two different types of cancers related to leukemia. The average coherence of the matrix evaluated to 0.9999 and the value of p = 5147 and n = 72. This is a high dimensional and highly coherent matrix. The performance of the sparse recovery algorithms when D=4 is given in Table 8 below. The locations of the non-zero values is randomly generated.

ALGORITHM	Pe (Probability of Error)	Pe_s (Probability of Error with SWAP)
TMSBL	0.9800	0.9950
BCS	1	0.9998
LASSO	0.9990	0.9980
OMP	0.9999	0.9998
FBMP	0.9999	0.9990

Table 8: Table showing the performance of various algorithms.

As observed from the Table 8, none of the sparse recovery algorithms considered resulted in an appreciative performance.

5 Conclusion

Compressive sensing is a developing field which asserts that one can recover certain signals and images from far fewer samples or measurements than traditional least squares based methods. In many important applications of CS algorithms like source localization in EEG, radar detection etc. the dictionary matrix is highly coherent. The efficiency of the various sparse recovery algorithms in reconstructing the signal when the design matrix is highly correlated is numerically studied in this report. The algorithms considered in this report include TMSBL, FBMP, OMP, BCS and LASSO. This study indicates that the performance of the above mentioned compressive sensing algorithms are dependent on the correlation structure of the matrix, position of the non-zero values in the signal $\mathbf{x}$ and the sparsity of $\mathbf{x}$. Better performance was obtained when the non-zero elements were chosen such that they correspond to least coherent columns of the design matrix $\mathbf{A}$, than when they correspond to the highly coherent columns of $\mathbf{A}$. In very high dimensional and highly coherent matrix like that obtained from the Leukemia data all the algorithms fail to report appreciable results. Also when the sparsity level is high all algorithms fail to perform efficient signal recovery. Moreover when the output of the recovery algorithm was post-processed using SWAP algorithm, improvement in the Pe obtained was not significant. Thus this report points to the need of developing efficient sparse recovery algorithms and post processing techniques that can work well in highly correlated dictionaries.

References

[1] A. M. Zoubir and D. R. Iskandler, "Bootstrap methods and applications," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 10–19, 2007.

[2] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[3] E. J. Candes, Y. C. Eldar, D. Needell, and P. Randall, "Compressed sensing with coherent and redundant dictionaries," *Applied and Computational Harmonic Analysis*, vol. 31, no. 1, pp. 59–73, 2011.

[4] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale-regularized least squares," *IEEE journal of selected topics in signal processing*, vol. 1, no. 4, pp. 606–617, 2007.

[5] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[6] Z. Zhang and B. D. Rao, "Sparse signal recovery with temporally correlated source vectors using sparse bayesian learning," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 5, pp. 912–926, 2011.

[7] P. Schniter, L. C. Potter, and J. Ziniel, "Fast bayesian matching pursuit," in *Information Theory and Applications Workshop, 2008.* IEEE, 2008, pp. 326–333.

[8] M. E. Tipping, A. C. Faul *et al.*, "Fast marginal likelihood maximisation for sparse bayesian models." in *AISTATS*, 2003.

[9] S. Ji, Y. Xue, and L. Carin, "Bayesian compressive sensing," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2346–2356, 2008.

[10] K. Qiu and A. Dogandzic, "Variance-component based sparse signal reconstruction and model selection," *IEEE Transactions on Signal Processing*, vol. 58, no. 6, pp. 2935–2952, 2010.

[11] D. Vats and R. G. Baraniuk, "Swapping variables for high-dimensional sparse regression with correlated measurements," *arXiv preprint arXiv:1312.1706*, 2013.

[12] D. Vats and R. Baraniuk, "When in doubt, swap: High-dimensional sparse recovery from correlated measurements," in *Advances in Neural Information Processing Systems*, 2013, pp. 989–997.

[13] D. P. Wipf and B. D. Rao, "An empirical bayesian strategy for solving the simultaneous sparse approximation problem," *IEEE Transactions on Signal Processing*, vol. 55, no. 7, pp. 3704–3716, 2007.

[14] E. J. Candès and C. Fernandez-Granda, "Super-resolution from noisy data," *Journal of Fourier Analysis and Applications*, vol. 19, no. 6, pp. 1229–1254, 2013.