Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 498589, 6 pages
http://dx.doi.org/10.1155/2013/498589

*Research Article*

# Sparse Recovery by Semi-Iterative Hard Thresholding Algorithm

**Xueqin Zhou,[1] Xiangchu Feng,[1] and Mingli Jing[2,3]**

[1] *School of Science, Xidian University, Xi'an, Shaanxi 710071, China*
[2] *School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China*
[3] *School of Statistics, Xi'an University of Finance and Economics, Xi'an, Shaanxi 710100, China*

Correspondence should be addressed to Xiangchu Feng; xcfeng@mail.xidian.edu.cn

Received 7 October 2012; Revised 30 November 2012; Accepted 10 December 2012

Academic Editor: Ion Zaballa



We propose a computationally simple and efficient method for sparse recovery termed as the semi-iterative hard thresholding (SIHT). Unlike the existing iterative-shrinkage algorithms, which rely crucially on using negative gradient as the search direction, the proposed algorithm uses the linear combination of the current gradient and directions of few previous steps as the search direction. Compared to other iterative shrinkage algorithms, the performances of the proposed method show a clear improvement in iterations and error in noiseless, whilst the computational complexity does not increase.

## 1. Introduction

Compressed sensing (CS) [1–3] is a new framework for acquiring sparse signals based on the revelation that a small number of linear measurements of the signal contain enough information for its reconstruction. CS relies on the fact that many natural signals are sparse or compressible when expressed in the proper basis and frame. The model of CS can be written as a linear sampling operator by a matrix $\mathbf{\Phi}$ yielding a measurement vector

$$\mathbf{y} = \mathbf{\Phi x}, \tag{1}$$

where $\mathbf{\Phi}$ is an $M \times N$ matrix, $\mathbf{x}$ is $S$-sparse vector, and $M \ll N$. Since the linear sampling operator $\mathbf{\Phi}$ is not bijection and therefore has infinitely many solutions. Efficient algorithms to find sparse solutions are becoming very important. This leads to solving the $l_0$-minimization problem

$$\min \|\mathbf{x}\|_0 \quad \text{s.t. } \mathbf{y} = \mathbf{\Phi x}. \tag{2}$$

Unfortunately, this minimization problem is NP-hard [2]. As alternatives, approximation algorithms are often considered. Approximation algorithms to find sparse solutions may be classified into greedy pursuits algorithms, convex relaxation algorithms, Bayesian framework, and nonconvex optimization. In this paper, we will focus on greedy pursuits algorithms and convex relaxation algorithms; thus more details of Bayesian framework and nonconvex optimization methods can be found in [4, 5]. Greedy pursuits algorithms include orthogonal matching pursuit (OMP) [6], stagewise OMP (StOMP) [7], regularized OMP (ROMP) [8], compressive sampling matching pursuit (CoSaMP) [9], iterative hard thresholding (IHT) [10], and gradient descent with sparsification (GraDeS) [11]. Convex relaxation algorithms include gradient projection for sparse reconstruction (GPSR) [12] and sparse reconstruction by separable approximation (SpaRSA) [13]. For more details about convex relaxation algorithms, see, for example [14]. Convex relaxation algorithms succeed with a very small number of measurements, but they tend to be computationally burdensome [15]. An alternative family of numerical algorithms has gradually built, addressing the optimization problems very effectively [15]. This family is the iterative-shrinkage algorithms. Iterative-shrinkage algorithms include iterative hard thresholding (IHT) [10] and gradient descent with sparsification (GraDeS) [11], parallel coordinate descent (PCD) [16], and fast iterative-shrinkage thresholding algorithm (FISTA) [17]. In these methods, each iteration consists of a multiplication by $\mathbf{\Phi}$ and its transpose, along with a scalar shrinkage step on the obtained $\mathbf{x}$. For iterative-shrinkage algorithms, IHT and GraDeS use a negative gradient as the search direction, that is, Landweber iteration [18], but the main drawback of Landweber iteration

is its slow performance, that is, a large number of iterations need to obtain the optimal convergence rates [18]. Inspired by the semi-iterative method [18] and hard thresholding, we present an algorithm for solving sparse recovery, which requires less time and fewer iterations.

## 2. Background on Compressed Sensing

*2.1. Sensing Matrix.* Without further information, it is impossible to recover $\mathbf{x}$ from $\mathbf{y}$, since $\mathbf{y} = \mathbf{\Phi x}$ is highly underdetermined. In order to recover a good estimate of $\mathbf{x}$ from $M$ measurements, the measurement matrix $\mathbf{\Phi}$ must obey the restricted isometry property (RIP) [2],

$$\left(1-\delta_s\right)\|\mathbf{x}\|_2^2 \le \|\mathbf{\Phi x}\|_2^2 \le \left(1+\delta_s\right)\|\mathbf{x}\|_2^2, \tag{3}$$

for all $\mathbf{x} \in \mathbf{\Sigma}_s$, $\mathbf{\Sigma}_s = \{\mathbf{x} \in R^N : \|\mathbf{x}\|_0 \le S\}$ denotes the set of $S$-sparse vectors, $\delta_s \in (0,1)$ is restricted isometry constant, $S \in \{1,2,\ldots,N\}$, provided that $M \ge C \cdot S \cdot \log(N/S)$, where $C$ is some constant depending on each instance. It is difficult to verify the RIP conditions for a given matrix. A widely used technique for avoiding checking the RIP directly is to generate the matrix randomly, such as Gaussian matrix, symmetric Bernoulli matrix, and partial Fourier matrix [1–3], and to show that the resulting random matrix satisfies the RIP with high probability. In this paper, we will use Gaussian matrix as the measurement matrix.

*2.2. Sparse Recovery.* An alternative approach to sparse signal recovery is based on the idea of iterative greedy pursuit and tries to approximate the solution to (2) directly. In this case, the problem (2) is closely related to the following optimization problem:

$$\min \frac{1}{2}\|\mathbf{y}-\mathbf{\Phi x}\|_2^2 \quad \text{s.t. } \|\mathbf{x}\|_0 \le s, \tag{4}$$

where $S$ denotes the sparse level of the vector $\mathbf{x}$.

The second one is convex relaxation. In this case, the problem (2) is closely related to the following optimization problem:

$$\min \|\mathbf{x}\|_1 \quad \text{s.t. } \mathbf{y} = \mathbf{\Phi x}. \tag{5}$$

However, these methods are often inefficient, requiring many iterations and excessive central processing unit time to reach their solutions [15].

An alternative family of numerical algorithms has gradually built, addressing the above optimization problems very effectively [15]. This family is the iterative-shrinkage algorithms. We will discuss iterative-shrinkage algorithms in the next section.

## 3. Semi-Iterative Hard Thresholding

The main drawback of Landweber iteration is its comparatively slow rate of convergence while for Landweber iteration only information about the last iterate $\mathbf{x}^{[k-1]}$ is used to construct the new approximation $\mathbf{x}^{[k]}$. In order to overcome the drawback, more sophisticated iteration methods have been developed on the basis of the so-called semi-iterative methods. A basic step of a semi-iterative method (polynomial acceleration methods) consists of one step of iteration, followed by an averaging process over all or some of the previously obtained approximations. A basic step of a semi-iterative method has the form

$$\begin{aligned}\mathbf{x}^{[k]} &= \mu_{1,k}\mathbf{x}^{[k-1]} + \mu_{2,k}\mathbf{x}^{[k-2]} + \cdots + \mu_{k,k}\mathbf{x}^{[0]} \\ &\quad + \omega_k \mathbf{\Phi}^{\mathrm{T}}\left(\mathbf{y} - \mathbf{\Phi x}^{[k-1]}\right),\end{aligned} \tag{6}$$

where $\sum_{i=1}^{k}\mu_{i,k} = 1$, $\omega_k \ne 0$, $k \ge 1$. An example for semi-iterative methods with optimal rate of convergence are the $\gamma$-methods (two-step methods) by [18], which are defined by

$$\mathbf{x}^{[k]} = \mathbf{x}^{[k-1]} + \mu_k\left(\mathbf{x}^{[k-1]} - \mathbf{x}^{[k-2]}\right) + \omega_k \mathbf{\Phi}^{\mathrm{T}}\left(\mathbf{y} - \mathbf{\Phi x}^{[k-1]}\right), \tag{7}$$

where

$$\begin{aligned}\mu_1 &= 0, \qquad \omega_1 = \frac{(4\gamma+2)}{(4\gamma+1)}, \\ \omega_k &= \frac{4\left(2k+2\gamma-1\right)\left(k+\gamma-1\right)}{\left(k+2\gamma-1\right)\left(2k+2\gamma-1\right)}, \\ \mu_k &= \frac{\left(k-1\right)\left(2k-3\right)\left(2k+2\gamma-1\right)}{\left(k+2\gamma-1\right)\left(2k+4\gamma-1\right)\left(2k+2\gamma-3\right)}, \\ \mathbf{x}^{[-1]} &= \mathbf{x}^{[0]} = \mathbf{x}_0.\end{aligned} \tag{8}$$

From (4), the gradient of the cost function $f(\mathbf{x}) = (1/2)\|\mathbf{y}-\mathbf{\Phi x}\|_2^2$ is given by $\nabla f(\mathbf{x}) = \mathbf{\Phi}^{\mathrm{T}}(\mathbf{\Phi x}-\mathbf{y})$ and easy to compute the step length $\alpha$ that minimizes $f(\mathbf{x}^{[k]} - \alpha\nabla f_k)$. By differentiating the function $\varphi(\alpha) = f(\mathbf{x}^{[k]} - \alpha\nabla f_k)$ with respect to $\alpha$, we obtain

$$\varphi'(\alpha) = f\left(\mathbf{x}^{[k]} - \alpha\nabla f_k\right)^{\mathrm{T}}\nabla f_k. \tag{9}$$

By setting the derivative to zero, we obtain

$$\alpha_k = \frac{\nabla f_k^{\mathrm{T}}\nabla f_k}{\nabla f_k^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\nabla f_k}. \tag{10}$$

If we choose the step lengths by (10), thus $f(\mathbf{x}^{[k]} - \alpha\nabla f_k)^{\mathrm{T}}\nabla f_k = 0$, that is, the search direction $\nabla f_{k+1} = \nabla f(\mathbf{x}^{[k]} - \alpha\nabla f_k)$ is orthogonal to the gradient $\nabla f_k$ (previous search direction). In this case, the sequence of iterations is subject to zigzags. Since IHT and GraDeS use the negative gradient of the cost function $f(\mathbf{x}) = (1/2)\|\mathbf{y}-\mathbf{\Phi x}\|_2^2$ as the search direction, and sampling matrix $\mathbf{\Phi}$ must obey the RIP, that is, $\|\mathbf{\Phi}\|_2^2 \approx 1$, which means $\alpha_k \approx 1$, thus the iteration zigzag toward the solution. As a result, a large number of iterations need to obtain the optimal solution.

In order to avoid zigzagging toward solution and find the sparse solution for (4), inspired by the $\gamma$-methods [18]

as mentioned above, we present the semi-iterative hard thresholding method, which has the form

$$\mathbf{x}^{[k]} = P_s\left(\mathbf{x}^{[k-1]} + \mu_k\left(\mathbf{x}^{[k-1]} - \mathbf{x}^{[k-2]}\right) + \omega_k \mathbf{\Phi}^{\mathrm{T}}\left(\mathbf{y} - \mathbf{\Phi}\mathbf{x}^{[k-1]}\right)\right), \tag{11}$$

where $P_s(\cdot)$ is the nonlinear operator that sets all but the largest (in magnitude) $S$ elements of a vector to zero. From (11), we use the linear combination of the current negative gradient $\mathbf{\Phi}^{\mathrm{T}}(\mathbf{y} - \mathbf{\Phi}\mathbf{x}^{[k-1]})$ and the search direction of the previous step $(\mathbf{x}^{[k-1]} - \mathbf{x}^{[k-2]})$ as the new search direction. In this case, the search direction $\mu_k(\mathbf{x}^{[k-1]} - \mathbf{x}^{[k-2]}) + \omega_k\mathbf{\Phi}^{\mathrm{T}}(\mathbf{y} - \mathbf{\Phi}\mathbf{x}^{[k-1]})$ dose not tend to become orthogonal to the gradient $\nabla f_k$; thus SIHT avoids zigzagging toward solution. The algorithm is summarized as in Algorithm 1.

As mentioned above, the semi-iterative hard thresholding algorithm is easy to implement. It involves the application of the matrix $\mathbf{\Phi}$ and $\mathbf{\Phi}^{\mathrm{T}}$ at each iteration as well as two vector additions. The storage requirements are small. Apart from storage of $\mathbf{y}$, we only require the storage of the vector $\mathbf{x}^{[k-1]}$ and $\mathbf{x}^{[k-2]}$, which require two $S$ elements to be stored. The choice of the parameter $\gamma$ will be discussed in the next section.

## 4. Experimental Results

This section describes some experiments testifying to the performances of the proposed algorithm. All the experiments were carried out on HP z600 workstation with eight Intel Xeon 2.13 GHz processors and 16 GB of memory, using a MATLAB implementation under Windows XP.

*4.1. Choice of the Parameter $\gamma$.* In our experiment, we consider a typical CS scenario, where the goal is to reconstruct a length-$N$ sparse vector from $M$ measurements. In this case, first, the $M \times N$ random matrix $\mathbf{\Phi}$ is created by filling it with entries generated independently and identically distribution and then orthogonalizing the rows. Second, original vector $\mathbf{x}$ contains $S$ randomly placed $\pm 1$ spikes, and the measurement $\mathbf{y}$ is generated according to (1). Unless otherwise stated, we terminate the iteration after $\|\mathbf{y} - \mathbf{\Phi}\mathbf{x}^{[k]}\|_2^2 \leq \varepsilon\|\mathbf{y}\|_2^2$, with $\varepsilon = 10^{-10}$.

The experiment assesses how the running time of the proposed algorithm grows with the parameter $\gamma$. In order to find a better optimization parameter $\gamma$ in the experiment, we set the parameter $\gamma$, respectively, to $\gamma \in \{2, 3, \ldots, 19, 20\}$, whilst the running time of our method is computed. Figure 1 shows the running time of our algorithm as the parameter $\gamma$ is varied. The label $M/N/S$ stands for $M$ measurements and $S$ sparse length-$N$ vector in our experiments. A careful examination reveals that as parameter $\gamma$ is increased, the running time of our method is minimized with respect to $\gamma = 7$. For $4 \leq \gamma \leq 20$, the running time increases only marginally as $\gamma$ is increased, that is, the choice of parameter $\gamma$ appears to give good performance for a wide range of problems.

*4.2. Comparison in Recovery Rate.* In this experiment, we compared the empirical performance of GraDes, IHT, SpaRSA, FISTA, and SIHT solutions to the sparse recovery. We generated a Gaussian $\mathbf{N}(0, 1)$ random matrix $\mathbf{\Phi} \in R^{256\times512}$ and generated $S$ sparse $\pm 1$ spikes vector. The reconstruction is considered to be exact when the 2 norm of the difference between the reconstruction and original vector is below $10^{-2}$. We repeated the experiment 100 times for each value of $S$ from 2 to 128 (in steps of 2). Figure 2 shows that SIHT algorithm provides higher probability of perfect recovery than GraDes, IHT, SpaRSA, and FISTA, when the sparse vectors are drawn $\pm 1$ spikes. Furthermore, in the perfect recovery case, we observe that the GraDes and SpaRSA algorithms perform similarly. While it reveals that measurements of SIHT require less than those of IHT, GraDes, SpaRSA, and FISTA to recover the sparse vector for a given $N$ and $S$.

*4.3. Comparison in Running Time.* In order to evaluate running time of the proposed algorithm, these experiments include comparisons with OMP, StOMP, ROMP, IHT, and GraDeS. Now, the sampling matrix $\mathbf{\Phi}$, the measurement vector $\mathbf{y}$, and the sparsity level $S$ are given to each of the algorithms as inputs. For the proposed algorithm, we set $\gamma = 7$; the performance is insensitive to the choices. Table 1 compares the running time of the MATLAB implementation of SIHT and the five existing methods. The symbol "$\infty$" indicates the algorithm fails.

Table 1 shows that the iterative-shrinkage algorithms are significantly faster than match pursuit algorithms. For simplicity, we will compare the performance of SIHT with IHT and GraDes in the next experiments.

*4.4. Comparison in Sparsity.* In this experiment, we show the dependence of the 2-norm errors of different algorithms in different sparsity level $S$. In Figure 3, we show the 2-norm errors of SIHT comparison with IHT, GraDes, SpaRSA, and FISTA in different sparsity levels. We generated a Gaussian $\mathbf{N}(0, 1)$ random matrix $\mathbf{\Phi} \in R^{512\times1024}$ and generated $S$ sparse $\pm 1$ spikes vector or Gaussian vector. We repeated the experiment 100 times for each value of $S$ from 2 to 120 (in steps of 10). Both GraDes and SpaRSA begin to fail when sparsity level is above 120; thus the failed results are omitted from the figure.

Figure 3 shows that GraDes, IHT, and SIHT algorithms perform similarly for Gaussian sparse vectors, and GraDes algorithm is to fail in recovery for sparse $\pm 1$ spikes vectors when sparsity level $S$ is above 70, that is, GraDes algorithm requires more measurements to recover the sparse vectors. It reveals that FISTA, IHT, and SIHT algorithms are insensitive to the sparsity level $S$, whilst GraDes and SpaRSA algorithms are sensitive to the sparsity level $S$. In addition, SIHT algorithm outperforms other algorithms in 2-norm errors for sparse $\pm 1$ spikes or Gaussian vector.

*4.5. Comparison in Number of Iterations.* In the experiment, we show the number of iterations required by SIHT algorithm in comparison with four algorithms, namely, IHT, GraDes, SpaRSA, and FISTA for sparse $\pm 1$ spikes vector or Gaussian vector. We generated a Gaussian $\mathbf{N}(0, 1)$ random matrix $\mathbf{\Phi} \in R^{512\times1024}$ and generated $S$ sparse $\pm 1$ spikes or Gaussian vector.

Table 1: The running time of different algorithm.

| Rows | Colum | Sparsity | Running time (seconds) | | | | | |
|---|---|---|---|---|---|---|---|---|
| $M$ | $N$ | $S$ | OMP | StOMP | ROMP | IHT | GraDes | SIHT (7) |
| 3000 | 8000 | 500 | 17.14 | 34.47 | 7.93 | 2.81 | 5.22 | 2.06 |
| 3000 | 10000 | 300 | 8.83 | $\infty$ | 2.77 | 3.52 | 5.69 | 2.72 |
| 3000 | 10000 | 600 | 19.39 | $\infty$ | $\infty$ | 9.33 | $\infty$ | 2.86 |
| 3000 | 10000 | 1050 | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| 4000 | 10000 | 500 | 25.33 | $\infty$ | 9.35 | 3.11 | 4.94 | 2.93 |

Step 1. Initialize $\mathbf{x}^{[-1]} = \mathbf{x}^{[0]} = \mathbf{x}_0$

Step 2. Compute for $k \geq 1$, for a given $\gamma$.

$$\mu_1 = 0, \quad \omega_1 = (4\gamma + 2)/(4\gamma + 1)$$

$$\omega_k = \left(4\left(2k + 2\gamma - 1\right)\left(k + \gamma - 1\right)\right) / \left(\left(k + 2\gamma - 1\right)\left(2k + 2\gamma - 1\right)\right), k \geq 2$$

$$\mu_k = \left(\left(k - 1\right)\left(2k - 3\right)\left(2k + 2\gamma - 1\right)\right) / \left(\left(k + 2\gamma - 1\right)\left(2k + 4\gamma - 1\right)\left(2k + 2\gamma - 3\right)\right), k \geq 2$$

$\mathbf{a}^{[k]} = \mathbf{x}^{[k-1]} + \mu_k(\mathbf{x}^{[k-1]} - \mathbf{x}^{[k-2]}) + \omega_k \mathbf{\Phi}^{\mathrm{T}}(y - \mathbf{\Phi}\mathbf{x}^{[k-1]})$

Step 3. Compute $\mathbf{x}^{[k]} = P_s(\mathbf{a}^{[k]})$

Step 4. Repeat step2-step3, until stopping criterion

$\left\|\mathbf{y} - \mathbf{\Phi}\mathbf{x}^{[k]}\right\|_2 \leq \varepsilon \|\mathbf{y}\|_2$ is satisfied.

Algorithm 1: Semi-iterative hard thresholding algorithm.

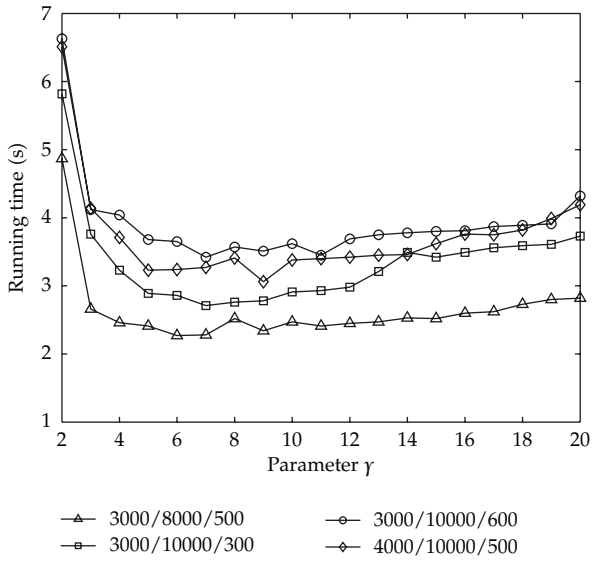


Figure 1: Influence of different parameter gamma on running time.

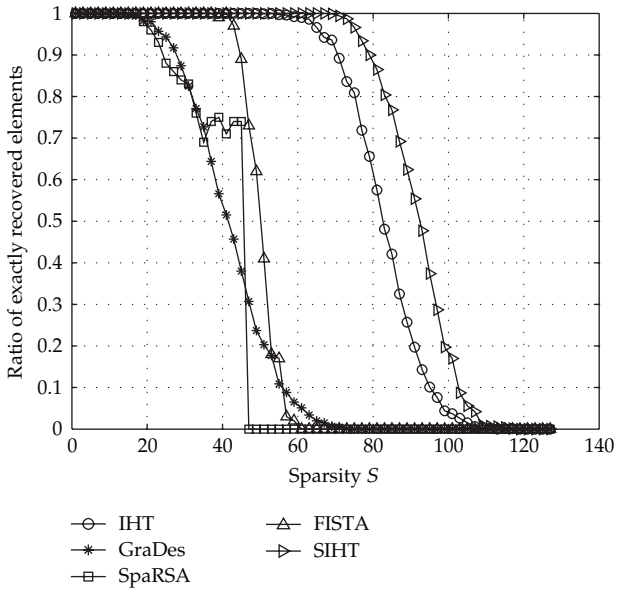


Figure 2: Simulation of the exact recovery rate.

Figures 4 and 5 show the number of iterations needed by the algorithms as mentioned above for $M = 512$, $N = 1024$, and $S = 20$.

Figures 4 and 5 depict that IHT and GraDes algorithms show a faster rate of convergence, when the number of iteration is less than 4. However, when the number of iteration is above 6, owing to polynomial acceleration, FISTA and SIHT algorithms show a faster rate of convergence than the others. In addition, from Figures 4 and 5, for each $S \geq 7$, FISTA and IHT algorithms are roughly similar in terms of number of iterations. In that SIHT algorithm uses the linear combination of the current gradient and directions of a few previous steps as the new search direction, SIHT algorithm shows a faster rate of convergence than the others. While GraDes algorithm exhibits poorer performance than the others in rate of convergence.

From Figures 4 and 5, the 2-norm errors of those algorithms except SpaRSA algorithm are insensitive to the number of iterations, that is, 2-norm errors are strictly monotone reduced as iteration is increased.

As expected, these results suggest that SIHT outperforms other iterative-shrinkage algorithms in iterations and 2-norm errors.

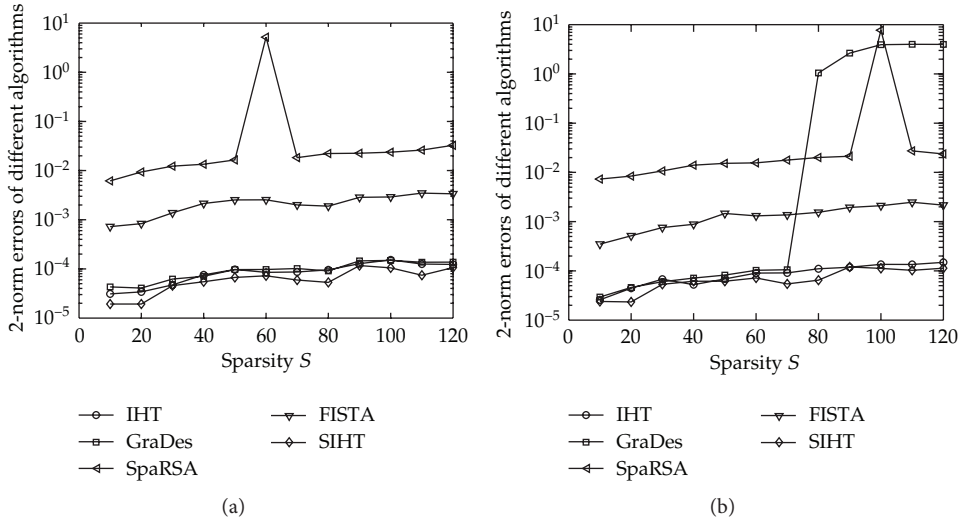


Figure 3: Recover error versus sparsity with fixed $M = 512$ and $N = 1024$: (a) Gaussian sparse vectors and (b) sparse ±1 spikes vectors.

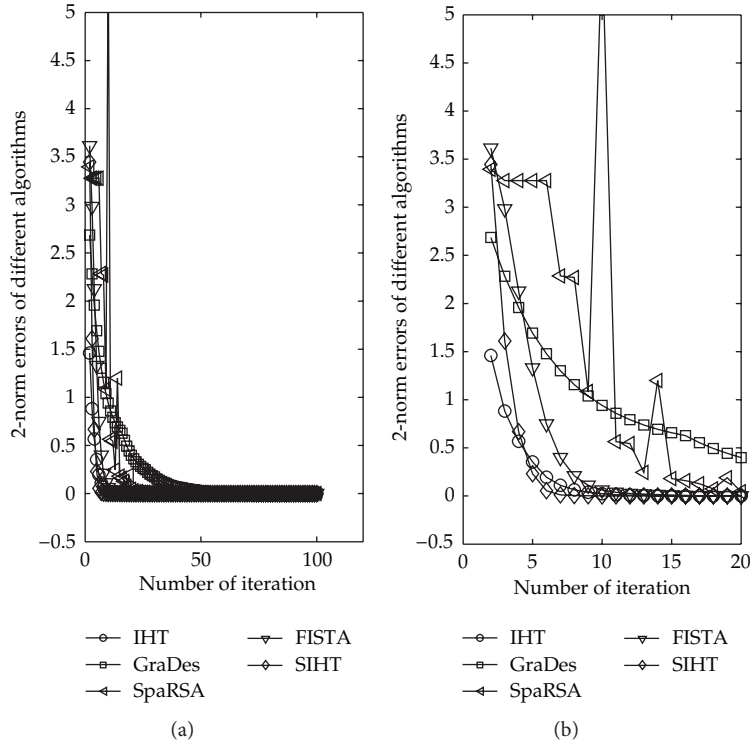


Figure 4: Recover error versus number of iteration with fixed $M = 512$, $N = 1024$, and $S = 20$ for Gaussian sparse vectors: (a) iteration between 2 and 120 and (b) iteration between 2 and 20.

## 5. Conclusions

In this paper, semi-iterative hard thresholding recovery algorithm for sparse recovery was proposed in this work. The proposed algorithm uses the linear combination of the current gradient and directions of a few previous steps as the new search direction and avoids zigzagging toward solution. Owing to using the new search direction, the performance of SIHT is improved compared with iterative-shrinkage algorithms.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant no. 61271294). The authors

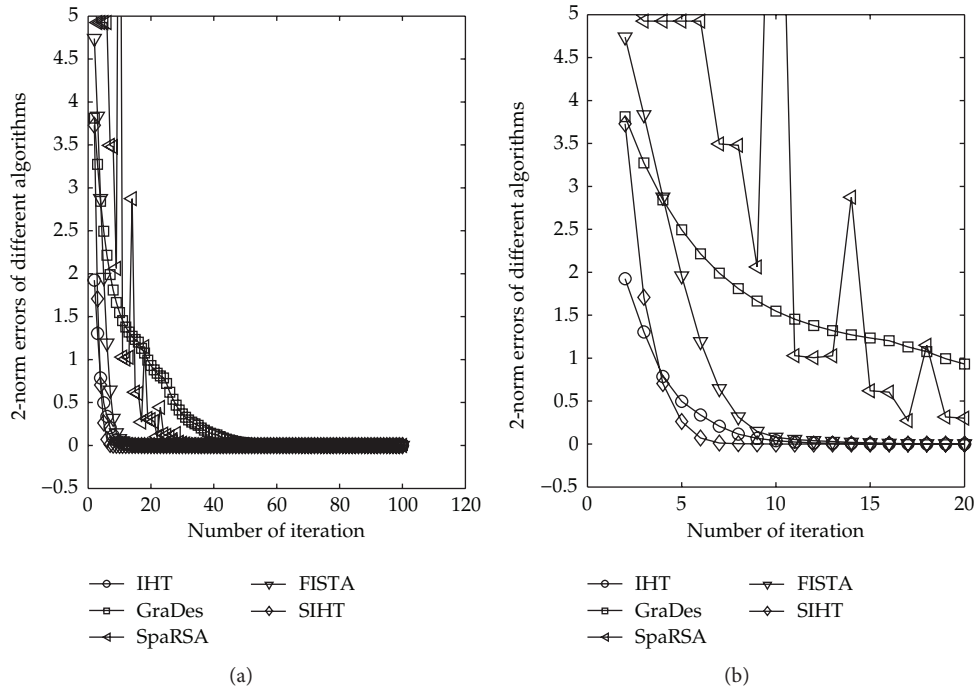


Figure 5: Recover error versus number of iteration with fixed $M = 512$, $N = 1024$, and $S = 20$ for sparse $\pm 1$ spikes vectors: (a) iteration between 1 and 120 and (b) iteration between 1 and 20.

would like to thank Arvind Ganesh, Allen Y. Yang, and Zihan Zhou for sharing their software packages (L1benchmark) with us.

## References

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

[2] E. J. Candes 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 and T. Tao, "Near-optimal signal recovery from random projections: universal encoding strategies?" *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5406–5425, 2006.

[4] 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.

[5] R. Chartrand, "Exact reconstruction of sparse signals via nonconvex minimization," *IEEE Signal Processing Letters*, vol. 14, no. 10, pp. 707–710, 2007.

[6] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

[7] D. L. Donoho, Y. Tsaig, I. Drori, and J. L. Starck, "Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit," Stanford Statistics Technical Report 2006-2, 2006.

[8] D. Needell and R. Vershynin, "Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit," *Foundations of Computational Mathematics*, vol. 9, no. 3, pp. 317–334, 2009.

[9] D. Needell and J. A. Tropp, "CoSaMP: iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 301–321, 2009.

[10] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, no. 3, pp. 265–274, 2009.

[11] R. Garg and R. Khandekar, "Gradient descent with sparsification: an iterative algorithm for sparse recovery with restricted isometry property," in *Proceedings of the 26th International Conference on Machine Learning*, pp. 337–344, 2009.

[12] M. Figueiredo, R. Nowak, and S. Wright, "Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 586–597, 2007.

[13] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo, "Sparse reconstruction by separable approximation," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2479–2493, 2009.

[14] A. Y. Yang, A. G. Zihan Zhou, S. S. Sastry, and Yi Ma, "Fast L1-minimization algorithms for Robust face recognition," . In press, http://arxiv.org/abs/1007.3753.

[15] M. Zibulevsky and M. Elad, "L1-L2 optimization in signal and image processing," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 76–88, 2010.

[16] M. Elad, B. Matalon, and M. Zibulevsky, "Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization," *Applied and Computational Harmonic Analysis*, vol. 23, no. 3, pp. 346–367, 2007.

[17] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[18] M. Hanke, "Accelerated Landweber iterations for the solution of ill-posed equations," *Numerische Mathematik*, vol. 60, no. 3, pp. 341–373, 1991.