# Accelerated Hard Thresholding Algorithms for Sparse Recovery

Xueci Zhao[1, a], Peibing Du[1,b], Tao Sun [1,c] and Lizhi Cheng[1,d]

[1] College of Science, National University of Defense Technology, Changsha, China

[a]zhaoxueci11@nudt.edu.cn, [b]dupeibing10@nudt.edu.cn,

[c]nudtsuntao@163.com, [d]clzcheng@nudt.edu.cn.

**Keywords:** Hard thresholding pursuit, restricted isometry property, compressive sensing, optimization methods.

**Abstract.** In linear observations, i.e., a system of linear equations $Ax = b\,(A \in \mathcal{R}^{M\times N})$, the hard thresholding pursuit (HTP) is used to find a sparse signal. HTP is an iterative algorithm that has been found many applications in compressive sensing, due to its good recovery performance, which includes linear convergence speed, high recovery rate, and stability. In this paper, we further develop accelerated algorithms to deal with a inear least square (LLS) problem in each iteration. Theoretically, we prove that all these algorithms are convergent, provided that the sensing matrix has suitable restricted isometry property. Numerical experiments on sparse signal recovery demonstrate the efficiency of the proposed methods.

## Introduction

The compressive sensing [1,2] aims to reconstruct a sparse signal $x \in \mathcal{R}^N$ from a few linear measurements $b = Ax$ ($A \in \mathcal{R}^{M\times N}$). Such problem frequently appeared in signal and image processing, face recognition, and more [3,4]. Many literatures have made great contributions to this research [1,2,3,4]. As is well known, the basis pursuit (BP) [1], which is one of the most famous reconstruction methods in , is to solve the following $l_1$-minimization problem,

$$\min_{x\in\mathcal{R}^N} \parallel x \parallel_1 \; st. Ax = y, \tag{1}$$

where $A \in \mathcal{R}^{M\times N}$ is the sensing matrix and $y \in \mathcal{R}^M$. Problem (1) is a convex problem, and then many convex methods are available. Although many methods are developed, solving BP still requires a lot of time. Besides the basic pursuit, paper [5] proposed an alternative strategy based on learning the sparsity of the signal, called hard thresholding pursuit (HTP), which is quite fast (converges linearly) and stable. The hard thresholding pursuit, which combines the hard thresholding strategy of iterative hard thresholding algorithm (IHT) [6] and the projection method of compressive sampling matching pursuit algorithm (CoSaMP) [7], can be described as follows:

$$S_{n+1} = C_s(x_n + A^*(y\text{-}Ax_n)), HTP_1$$

$$x_{n+1} = arg\min_z \parallel y\text{-}Az \parallel_2, supp(z) \subseteq S_{n+1}, HTP_2$$

where $x_n$ is the $n^{th}$ iterate and $C_s(x)$ denotes the indices of $s$ largest absolute entries of $x$.

Here, we introduce the definition of Restricted Isometry Property (RIP) which plays an important role in analyzing the convergence of HTP algorithm. The $s$ order restricted isometry constant $\delta_s = \delta_s(A)$ of $A \in \mathcal{R}^{M\times N}$ has been defined in [8] as the smallest $\delta$ such that

$$(1-\delta) \parallel x \parallel_2^2 \leq \parallel Ax \parallel_2^2 \leq (1+\delta) \parallel x \parallel_2^2, \tag{2}$$

for any $s$-sparse vector $x \in \mathcal{R}^N$. The author in [5] has proved that the HTP algorithm has a linear convergent rate provided $\delta_{3s} < \frac{\sqrt{3}}{3}$.

In the second step of each iteration of HTP algorithm (HTP 2), a linear least square (LLS) problem needs to be solved. However, as the scale of the problem increases, solving LLS directly will be very expensive sometimes even intractable. Due to this, HTP algorithm will encounter challenge if the scale of the problem and the sparsity level are all very large. A scheme called fast hard thresholding pursuit (FHTP) proposed in [5] provides a method in large scale case. The main idea of FHTP is generating an approximate solution by applying the gradient method to the LLS problem in HTP 2 rather than solving it exactly. Here, we present the detailed procedure of this scheme:

**Algorithm 1** Fast Hard Thresholding Pursuit[5]

```
Require: parameters $h^i_{n+1}, k$
  Initialization: $y^0 = 0$
  for $n = 0, 1, 2, \ldots$
    $S_{n+1} = C_s(x_n + A^*(y - Ax_n))$
    $x^0_{n+1} = (x_n + A^*(y - Ax_n))_{S_{n+1}}$
    $y^1_{n+1} \in \mathcal{R}^s, y^1_{n+1} = (x^0_{n+1})_{S_{n+1}}$
    for $i = 1, 2, \ldots, k-1$
      $A_{n+1} \in \mathcal{R}^{n \times s}, A_{n+1} = A(:, S_{n+1})$
      $y^{i+1}_{n+1} = y^i_{n+1} + h^i_{n+1} A^*_{n+1}(y - A_{n+1} y^i_{n+1})$
    end for
    $x_{n+1} \in \mathcal{R}^n, (x_{n+1})_{S_{n+1}} = y^k_{n+1}, (x_{n+1})_{\overline{S_{n+1}}} = 0$
  end for
```

where $h^i_{n+1}$ is the step parameter. The FHTP also has a linear convergence rate if the iterative number $k$ in the loop of Algorithm 1 is big enough and $\delta_{3s} < \frac{\sqrt{3}}{3}$. Paper [5] proves that when $h^i_{n+1} \equiv 1$ the following inequality holds,

$$\| x_{n+1} - x \|_2 \le \sqrt{\frac{2\delta_{3s}^2 + \delta_{3s}^{2k+2}(1 - 3\delta_{3s}^2)}{1 - \delta_{3s}^2}} \| x_n - x \|_2, \tag{3}$$

where $x$ is any $s$-sparse and obeys $Ax = b$, $\{x_n\}_{n=0,1,2,\ldots}$ are obtained by FHTP.

Note that HTP 2 is actually a optimization problem, which can be solved by faster optimization method such as Nesterov method [9, 10], from the perspective of optimization. While, HTP 2 is also to solve a positive definition linear system, which can be solved by fast linear algebra method such as conjugate gradient method [11], from the perspective of linear algebra. In this paper, we show that if the gradient method applied in FHTP 2 is replaced by other methods, the scheme is also convergent. Therefore, we can employ some other fast methods to get faster HTP schemes. The rest of the paper is organized as follows. Section 2 introduces the preliminary knowledge. Section 3 presents the algorithms and their convergence. Section 4 contains numerical results. Finally, Section 5 concludes the paper.

## Notations

In this paper, $A = \{A_1, A_2, \ldots, A_N\} \in \mathcal{R}^{M \times N}$, $x = \{x(1), x(2), \ldots, x(n)\}^* \in \mathcal{R}^N$ and $S = \{i_1, i_2, \ldots, i_s\} \subseteq \{1,2,\ldots,N\}$. $\overline{S}$ denotes as the set $\{1,2,\ldots,N\} \backslash S$, $x_S \in \mathcal{R}^N$ denotes as $x_S(i) = h(i), i \in S$, $x_S(i) = 0, i \in \overline{S}$. $A(:,S) \in \mathcal{R}^{M \times k}$ denotes as $A(:,S) = \{A_{i_1}, A_{i_2}, \ldots, A_{i_s}\}$.

## Preliminaries

In this part, we introduce two powerful algorithms which have been widely used in optimization and numerical linear algebra. These two algorithms will be employed in the proposed algorithm to accelerate the speed of HTP.

**2.1 The Nesterov method.** The Nesterov method can be used to accelerate the speed of gradient method in minimizing the smooth convex function problem

$$\min f(x), \tag{4}$$

where $f$ has a Lipschitz continuous gradient with constant $L = L(f)$, i.e.,

$$\| \nabla f(x_1) - \nabla f(x_2) \|_2 \le L \| x_1 - x_2 \|_2, x_1, x_2 \in dom(f). \tag{5}$$

There are many variants of the Nesterov method [12, 13, 14]. Here, we present a simple one appeared in [14] as follows :

**Algorithm 2** The Nesterov method[14]

```
Require: parameters $h > 0$
  Initialization: $y^0 = x^0, \theta_0 = 1$
  for $k = 0, 1, 2, \ldots$
    $x^{k+1} = y^k - h_k \nabla f(y^k)$
    $\beta_{k+1} = \frac{1}{2}(1-\theta_k)(\sqrt{\theta_k^2+4}-\theta_k)$
    $y^{k+1} = x^k + \beta_k(x^{k+1} - x^k)$
    $\theta_{k+1} = \frac{1}{2}\theta_k(\sqrt{\theta_k^2+4}-\theta_k)$
  end for
```

Actually, this algorithm is equivalent to Constant Step Scheme II on Page 80 of [9] and FISTA on Page 193 of [12] without the nonsmooth regularization function. The $h_k$ is the step parameter, a usually choice is $h_k = \frac{1}{L}$. The author in [9] shows that

$$f(y^k)\text{-}f^* \leq \frac{2L\|y^0\text{-}y^*\|_2^2}{(k+1)^2}, \tag{6}$$

where $y^*$ is a solution to (4) and $f^* = f(y^*)$. When apply Algorithm 2 to $HTP_2$, the objective function is $f(x) = \| y\text{-}A_n x \|_2^2$, where $A_n = A(:,S_n) \in \mathcal{R}^{M\times s}$. Because $A$ has the restricted isometry property, it is easy to obtain that

$$\| A_n x \|_2^2 \leq (1+\delta_s) \| x \|_2^2 < 2, \forall x \in \mathcal{R}^s. \tag{7}$$

That means $\| A_n^* A_n \|_2 < 2$, which also means that

$$\| \nabla f(x_1) - \nabla f(x_2) \|_2 = \| A_n^* A_n (x_1\text{-}x_2) \|_2 < 2 \| x_1\text{-}x_2 \|_2. \tag{8}$$

Then, we can set $h_k = \frac{1}{2}$ in Algorithm 4 (see below).

**2.2 Conjugate Gradient Method.** The HTP 2 is essentially solving following linear system

$$A_n^* A_n x = A_n^* y, \tag{9}$$

where $A_n = A(:,S_n) \in \mathcal{R}^{M\times s}$ and $x \in \mathcal{R}^s$. Because $A$ has the restricted isometry property, we can immediately get two conclusions: the first is that $A_n^* A_n$ is positive definition; the second is that $cond(A_n^* A_n) \leq \frac{1+\delta_s}{1-\delta_s}$. Based on these two points, we can apply conjugate gradient method (CG) whose convergence rate closely depends on the condition number of the coefficient matrix to (9).

We emphasize that the matrix-matrix-vector multiplication $A_n^* A_n x$ should be computed as following scheme: $\bar{z} \to A_n x, z \to A_n^* \bar{z}$. A linear convergent rate about CG method has been proposed by [15] as

$$\| x_k\text{-}x^* \|_{A_n^* A_n} \leq 2\left(\frac{\sqrt{\kappa}\text{-}1}{\sqrt{\kappa}+1}\right)^k \| x_0\text{-}x^* \|_{A_n^* A_n}, \tag{10}$$

**Algorithm 3** Conjugate gradient method to (9)[15]

```
Require: Initialization $x_0 \in \mathcal{R}^s$, compute $r_0 = A_n^* y - A_n^* A_n x_0$, let $p_0 = r_0$
  for $k = 0, 1, 2, \ldots$
    compute $\alpha_k = \frac{\langle r_k, r_k \rangle}{\langle A_n p_k, A_n p_k \rangle}$
    $x_{k+1} = x_k + \alpha_k p_k$
    $r_{k+1} = r_k - \alpha_k A_n^* A_n p_k$
    $\beta_k = \frac{\langle r_{k+1}, r_{k+1} \rangle}{\langle r_k, r_k \rangle}$
    $p_{k+1} = r_{k+1} + \beta_k p_k$
  end for
```

where $x^*$ is the exact solution to (9), $\| x \|_{A_n^* A_n} = x^T A_n^* A_n x$ and $\kappa$ is the condition number of $A_n^* A_n$. Considering $A_n$ is a submatrix of $A$ which is restricted isometry one, it's easy to get that

$$(1-\delta_s) \| x_k\text{-}x^* \|_2^2 \leq \| A_n(x_k\text{-}x^*) \|_2^2 \leq (1+\delta_s) \| x_k\text{-}x^* \|_2^2, \tag{11}$$

and $\kappa \leq \frac{1+\delta_s}{1-\delta_s} \doteq \kappa_0$. Substituting this into (10), we derive that

$$\| x_k\text{-}x^* \|_2 \le 2\sqrt{\kappa_0}\left(\frac{\sqrt{\kappa_0}\text{-}1}{\sqrt{\kappa_0}+1}\right)^k \| x_0\text{-}x^* \|_2. \tag{12}$$

**Convergence analysis**

Assume that Algorithm $X$ is a convergent iterative scheme which can be used to solve HTP 2. Just like the FHTP, we propose a class of schemes. In Algorithm 4, Algorithm $X$ can be Algorithms 2, 3 or other algorithms.

```
Algorithm 4 Fast Hard Thresholding Pursuit via Algorithm X
Require: parameters needed in Algorithm $X$ and $l > 0$.
  Initialization: $x_0 = 0$
  for $n = 0, 1, 2, \ldots$
    $S_{n+1} = C_s(x_n + A^*(y - Ax_n))$
    $x^0_{n+1} = (x_n + A^*(y - Ax_n))_{S_{n+1}}$
    Apply Algorithm $X$ to HTP$_2$ to get $x^1_{n+1}, \ldots, x^l_{n+1}$
    $x_{n+1} = x^l_{n+1}$.
  end for
```

But no matter which one, Algorithm 4 is convergent provided Algorithm $X$ is convergent.

**Theorem 3.1** *Suppose that the sequence* $x^i_{n+1}$ *generated by Algorithm* $X$ *satisfies* $\| x^i_n\text{-}x^*_n \| \le \gamma(i) \| x^0_n\text{-}x^*_n \|$. *Then, for any* $s$*-sparse* $x$ *which obeys* $y = Ax$, *the sequence* $x_n$ *generated by Algorithm 4 satisfies*

$$\| x_n\text{-}x \|_2 \le \rho^n \| x_0\text{-}x \|_2, \tag{13}$$

*where* $\rho = (1 + \sqrt{2}\delta_{3s})(\delta_{2s} + \gamma(l))\sqrt{\frac{1+\delta_{2s}}{1-\delta_{2s}}} + (1 + \sqrt{2})\delta_{3s}\gamma(l) + \sqrt{2}\delta_{3s}$.

Because Algorithm $X$ is convergent, $\lim_{l\to\infty}\gamma(l) = 0$. Let $l \to \infty$, $\rho^0 = (1 + \sqrt{2}\delta_{3s})\delta_{2s}\sqrt{\frac{1+\delta_{2s}}{1-\delta_{2s}}} + \sqrt{2}\delta_{3s}$. Note that $0 < \delta_{2s} < \delta_{3s} < 1$, it's easy to get $\rho^0 < (1 + \sqrt{2}\delta_{3s})\delta_{3s}\sqrt{\frac{1+\delta_{3s}}{1-\delta_{3s}}} + \sqrt{2}\delta_{3s}$. If $\delta_{3s} < 0.3014$, then $\rho^0 < 1$. The numerical experiments demonstrate Algorithm 4 can work as well as FHTP in reconstruction ability.

**Numerical Results**

In this section, we report numerical results about Algorithm 4. In the experiments, 'X' will be substituted by the Nesterov method and CG method. All the experiments are run in MatLab 2010a on a server with 2.60GHz Intel processor and 3.90 GB RAM.

Table 1: Time cost by different algorithms. The sensing matrix $A \in \mathcal{R}^{2000\times 10000}$.

| scenario | BP | CoSaMP | HTP | FHTP | X=Nesterov | X=CG |
|---|---|---|---|---|---|---|
| Gaussian matrices and Gaussian vectors | | | | | | |
| s=100 | 690.6s | 0.7s | 15.5s | 15.3s | 15.2s | 15.3s |
| s=200 | 937.2s | 11.4s | 15.4s | 15.2s | 15.1s | 15.2s |
| s=300 | 992.3s | 51.8s | 15.6s | 15.5s | 15.3s | 15.5s |
| s=400 | 1385.8s | 136.8s | 16.1s | 15.8s | 15.4s | 15.7s |
| Gaussian matrices and Bernoulli vectors | | | | | | |
| s=100 | 409.6s | 0.2s | 13.2s | 13.9s | 11.8s | 12.2s |
| s=200 | 444.5s | 3.2s | 15.4s | 15.2s | 15.1s | 14.7s |
| s=300 | 447.2s | 12.9s | 13.7s | 14.6s | 12.8s | 13.1s |
| s=400 | 453.7s | 32.2s | 14.5s | 16.9s | 13.7s | 13.9s |
| Bernoulli matrices and Gaussian vectors | | | | | | |
| s=100 | 645.5s | 2.6s | 14.9s | 19.6s | 16.5s | 19.9s |
| s=200 | 638.2s | 52.7s | 13.5 | 15.5 | 13.7s | 15.2s |
| s=300 | 784.8s | 176.9s | 13.6s | 19.5s | 14.5s | 16.6s |
| s=400 | 1314.1s | 1507.5s | 14.3s | 24.3s | 15.4s | 19.3s |
| Bernoulli matrices and Bernoulli vectors | | | | | | |
| s=100 | 330.3s | 0.6s | 13.5s | 15.2s | 13.6s | 13.8s |
| s=200 | 351.1s | 7.5s | 12.9s | 16.8s | 14.0s | 16.3s |
| s=300 | 419.2s | 665.2s | 13.2s | 19.1s | 15.8s | 18.4s |
| s=400 | 406.1s | 1531.9s | 13.9s | 21.1s | 18.7s | 19.8s |

Table 2: Time cost by different algorithms. The sensing matrix $A \in \mathcal{R}^{4000\times 10000}$

| scenario | BP | CoSaMP | HTP | FHTP | X=Nesterov | X=CG |
|---|---|---|---|---|---|---|
| Gaussian matrices and Gaussian vectors | | | | | | |
| s=100 | 992.9s | 1.1s | 21.6s | 21.4s | 20.9s | 21.5s |
| s=200 | 1117.2s | 10.1s | 22.0s | 22.0s | 21.8s | 22.1s |
| s=300 | 1174.7s | 30.9s | 22.4s | 22.1s | 22.0s | 21.7s |
| s=400 | 1226.3s | 34.1s | 22.9s | 23.4s | 22.1s | 22.1s |
| Gaussian matrices and Bernoulli vectors | | | | | | |
| s=100 | 760.5s | 0.2s | 21.1s | 21.5s | 21.3s | 21.5s |
| s=200 | 766.9s | 9.3s | 21.2s | 21.9s | 21.4s | 21.6s |
| s=300 | 760.3.2s | 30.6s | 22.0s | 22.8s | 21.3s | 21.4s |
| s=400 | 765.5s | 32.2s | 14.5s | 16.9s | 13.7s | 13.9s |
| Bernoulli matrices and Gaussian vectors | | | | | | |
| s=100 | 944.2s | 2.1s | 21.9s | 25.1s | 22.1s | 25.2s |
| s=200 | 1116.9s | 21.2 | 22.5s | 27.5s | 23.0s | 26.4s |
| s=300 | 1175.7s | 104.0s | 22.7s | 25.9s | 23.7s | 27.2s |
| s=400 | 1220.1s | 369.1s | 27.3s | 31.8s | 24.2s | 28.5s |
| Bernoulli matrices and Bernoulli vectors | | | | | | |
| s=100 | 643.4s | 0.7s | 21.9s | 25.6s | 22.7s | 24.8s |
| s=200 | 705.7s | 8.3 | 22.6s | 25.8s | 22.3s | 25.2s |
| s=300 | 775.4s | 21.3s | 22.3s | 24.5s | 23.0s | 27.7s |
| s=400 | 856.1s | 33.2s | 25.2s | 31.6s | 24.0s | 31.4s |

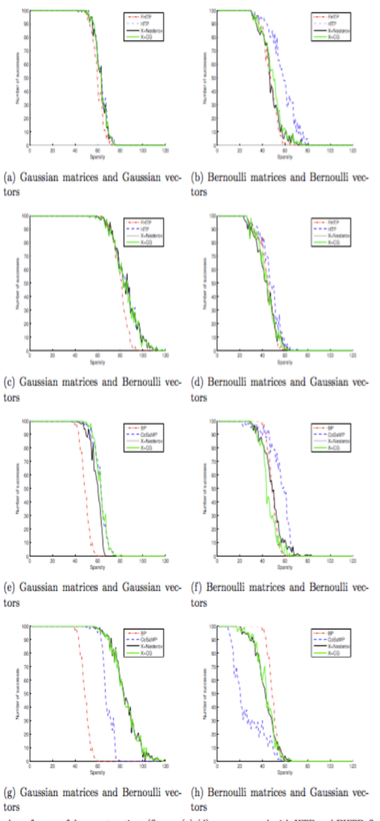
Table 3: Time cost by different algorithms. The sensing matrix $A \in \mathcal{R}^{2000\times 12000}$.

| scenario | BP | CoSaMP | HTP | FHTP | X=Nesterov | X=CG |
|---|---|---|---|---|---|---|
| Gaussian matrices and Gaussian vectors | | | | | | |
| s=100 | 897.4s | 1.1s | 18.4s | 17.0s | 16.4s | 16.6s |
| s=200 | 1099.7s | 11.2s | 17.1s | 17.5s | 16.8s | 17.0s |
| s=300 | 1302.8s | 42.6s | 17.9s | 18.4s | 17.6s | 18.1s |
| s=400 | 1506.2s | 138.8s | 18.1s | 18.2s | 17.7s | 18.1s |
| Gaussian matrices and Bernoulli vectors | | | | | | |
| s=100 | 459.1s | 0.2s | 16.3s | 16.8s | 16.4s | 16.6s |
| s=200 | 520.4s | 3.3s | 16.1s | 17.3s | 16.4s | 16.6s |
| s=300 | 610.7s | 13.0s | 16.7s | 23.8s | 17.1s | 17.8s |
| s=400 | 680.0s | 30.1s | 16.9s | 21.9s | 18.3s | 18.4s |
| Bernoulli matrices and Gaussian vectors | | | | | | |
| s=100 | 877.9s | 5.1s | 16.1s | 20.2s | 17.6s | 20.2s |
| s=200 | 1045.0s | 254.3 | 17.8s | 22.7s | 19.0s | 22.3s |
| s=300 | 1414.1s | 994.1s | 18.2s | 24.9s | 22.3s | 26.4s |
| s=400 | 1773.9s | 1722.8s | 18.7s | 28.7s | 21.7s | 25.3s |
| Bernoulli matrices and Bernoulli vectors | | | | | | |
| s=100 | 472.6s | 0.7s | 16.7s | 21.7s | 17.7s | 19.9s |
| s=200 | 468.7s | 9.1 | 16.7s | 23.6s | 19.2s | 22.1s |
| s=300 | 656.8s | 727.0s | 17.6s | 24.6s | 22.2s | 24.7s |
| s=400 | 1500.0s | 1363.0s | 19.7s | 32.3s | 24.3s | 31.2s |

*1)* The first experiment is about recovery ability of Algorithm 4. To show the results clearly, this experiment contains two parts: one focuses on comparing Algorithm 4 with FHTP and HTP; the other one focuses on comparing Algorithm 4 with other classical algorithms, namely the BP, CoSaMP algorithms (the matlab codes of these algorithms can be found in [18]). In this experiment, the sensing matrix $A$ is a $200\times 1000$ Gaussian matrix or Bernoulli matrix and $s$-sparse Gaussian or Bernoulli signals are employed. We run the algorithms with a stopping criterion $\| y\text{-}Ax_n \|_2 \leq 10^{-4} \| y \|_2$ or $100$ steps. And $\| x\text{-}x_n \|_2 \leq 10^{-6} \| x \|_2$ means a successful reconstruction. For each $s \in [1,120]$, we use the 100 different matrices and sparse vectors. Figure 1 presents the numerical results. The results of experiment 1 show that the Algorithm 4 have proved capable of recovering the sparse signal from a small amount of linear observations. Just as what we declared before, the Algorithm 4 can work well in numerical tests.

*2)* The second experiment is about the time cost by Algorithm 4. In this test, we use three sensing matrices: $2000\times 10000$, $4000\times 10000$, and $2000\times 12000$. In this test, we concern on comparing

Algorithm 4 with the BP, CoSaMP, HTP and FHTP algorithm. The stopping criterion can be only set as $\| y\text{-}Ax_n \|_2 \leq 10^{-4} \| y \|_2$ for that the reconstructions are all successful. Tables 1, 2 and 3 present the numerical results. Form the results of experiment 2, we can find that with almost the same accuracy Algorithm 4 with "X=Nesterov" and "X=CG" are faster than FHTP and BP. When the sparsity decreases, Algorithm 4 is also faster than CoSaMP. As the scale of the problem increases, more time can be saved.

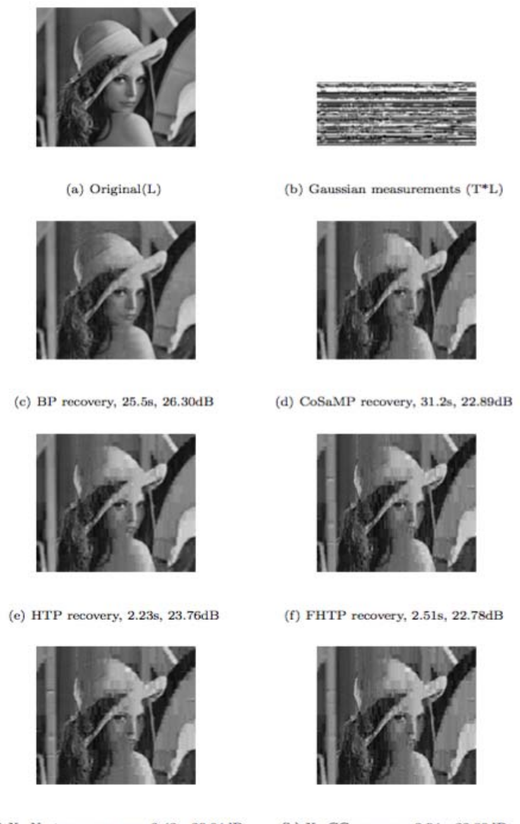

(a) Gaussian matrices and Gaussian vectors (b) Bernoulli matrices and Bernoulli vectors

(c) Gaussian matrices and Bernoulli vectors (d) Bernoulli matrices and Gaussian vectors

(e) Gaussian matrices and Gaussian vectors (f) Bernoulli matrices and Bernoulli vectors

(g) Gaussian matrices and Bernoulli vectors (h) Bernoulli matrices and Gaussian vectors

Figure 1. The number of successful reconstructions (figures (a)-(d) are compared with HTP and FHTP, figures (e)-(h) are compared with BP and CoSaMP)


(a) Original(L) (b) Gaussian measurements (T*L)

(c) BP recovery, 25.5s, 26.30dB (d) CoSaMP recovery, 31.2s, 22.89dB

(e) HTP recovery, 2.23s, 23.76dB (f) FHTP recovery, 2.51s, 22.78dB

(g) X=Nesterov recovery, 2.42s, 23.04dB (h) X=CG recovery, 2.34s, 22.89dB

Figure 2. Recovery from Gaussian measurements

*3)* The third experiment is conducted on a real-world example. We employ the *Lena* picture ($L \in \mathcal{R}^{256\times256}$) and the Haar wavelet frame matrix $H$[17] for test. We concern on reconstructing $L$ from linear measurements $G = TL$, where $T \in \mathcal{R}^{120\times256}$ is a Gaussian matrix. As is well known, $L$ can be expressed as $L = HX$, where $X \in \mathcal{R}^{256\times256}$ and most entries of $X$ are relative small. Therefore, $X$ can be regarded as a sparse matrix. Then, the problem reduces to recovering sparse $X$ from $G = (TH)X$.

The BP, CoSaMP, HTP, FHTP and Algorithm 4 are used to reconstruct the picture. The parameter $s$ is set as $s = 20$. Figure 2 presents the reconstructed picture, the time cost, and the Signal-to-Noise Ratio(SNR).

## Conclusion

In this paper, we propose a class of schemes for sparse recovery. The FHTP method can be regarded as a special one of this class. Theoretically, these proposed algorithms are proved to converge linearly. Finally, the numerical result validates the feasibility and efficiency of the proposed schemes.

## Acknowledgements

Partially supported by National Natural Science Foundation of China (No. 61571008). Partially supported by National Natural Science Foundation of China (No. 61402495, No. 61602166, No. 61303189, No. 61402496). Partially supported by Science Project of National University of Defense Technology (JC120201) and National Natural Science Foundation of Hunan Province in China (13JJ2001).

## References

[1] D. L. Donoho, Compressed sensing, IEEE Trans. Inf. Theory. 52 (2006) 289-306.

[2] D. L. Donoho, “For most large underdetemind systems of linear equations, the minimal $\ell_1$-norm solution is also the sparsest solution," *Commun. Pure Appl. Math.*, Vol. 59, pp. 907-934, 2006.

[3] E. Candes and T. Tao, “Decoding by linear programing," *IEEE Trans. Inf. Theory*, vol. 51, pp. 4203-4215, 2005.

[4] E. Candes, J. Romberg, and T. Tao, “Robust uncertainly principles: Exact signal reconstruction from highly incomplete frequency information", IEEE Trans. Inf. Theory. 52 (2006) 489-509.

[5] S. Foucart, Hard thresholding pursuit: an algorithm for compressive sensing, SIAM J. Numer. Anal. 49 (2011) 2543-2563.

[6] T. Blumensath and M. E. Davies, Iterative hard thresholding for compressed sensing, Appl. Comput. Harmon. Anal. 27 (2009) 265-274.

[7] D. Needell and J. A. Tropp, CoSaMP: Iterative signal recovery from incomplete and inaccurate samples, Appl. Comput. Harmon. Anal. 26 (2009) 301-320.

[8] E. Candes and T. Tao, Decoding by linear programing, IEEE Trans. Inf. Theory. 51 (2005) 4203-4215.

[9] Y. Nesterov, Introductory lectures on convex optimization: A basic course, Kluwer Academic Publishers, 2004.

[10] Y. Nesterov, Gradient methods for minimizing composite objective function, CORE discussion paper, 2007.

[11] J. H. Demmel, Applied numerical linear algebra, Beijing: Tsinghua university press, 2011.

[12] A. Beck and M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM J. Img. Sci. 2 (2009) 183-202.

[13] A. Auslender and M. Teboulle, Interior gradient and proximal methods for convex and conic optimization, SIAM J. Opt. 16 (2006) 697-725.

[14] H. Zhang and W. Yin, Gradient methods for convex minimization: better rates under weaker conditions, UCLA CAM Report, 2013.

[15] D. M. Young, Iterative solution of large linear systems, New York: Academic, 1971.

[16] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma: Robust face recognition via sparse representation. IEEE Trans. Pattern Anal. 31 (2009) 210-227.

[17] J.-F. Cai, R. H. Chan, and Z. W. Shen: A framelet-based image inpainting algorithm. Appl. Comput. Harmon. Anal. 24 (2008) 131-149.

[18] Carron I: Compressive Sensing: The Big Picture. A living document trying to paint the Big Picture in the Compressed Sensing or Compressive Sensing Framework, 2009.