Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 9, No. 1, pp. 137–154, January 2013

# A SELF-ADAPTIVE PROJECTION AND CONTRACTION METHOD FOR OPTIMIZING THE SUM OF A CONVEX AND A QUADRATIC FUNCTION

Yuan Shen and Chengjing Wang*

**Abstract:** In some convex optimization problems, the objective function is a sum of two convex functions, of which the one is a quadratic term, and the other is a convex one with some special structure. In this paper, we present some new contraction methods. They are closely related to the some existing methods such as Proximal Point Algorithm (PPA), but they have better performance compared with the classic PPA. We applied these novel methods into Compressive Sensing (CS) problems, and achieved satisfactory numerical results.

**Key words:** *convex optimization, projection and contraction method, proximal point algorithm, forward-backward splitting, compressed sensing, iterative shrinkage thresholding*

**Mathematics Subject Classification:** *65K05, 65K10, 65K15, 90C25*

## 1 Introduction

### 1.1 The Model

We consider such an optimization problem which involves two terms in objective function:

$$\min \{\phi(x) = \theta(x) + q(x) \mid x \in \Omega\}. \tag{1.1a}$$

In which, $\theta(x) : \mathbb{R}^n \to \mathbb{R}$ is a convex (not necessarily smooth) function with some special structure, and $q(x)$ is a quadratic function such as:

$$q(x) = \frac{1}{2}x^T Hx + c^T x, \tag{1.1b}$$

where $H \in \mathbb{R}^{n\times n}$ is symmetric positive semi-definite matrix, $c \in \mathbb{R}^n$, and $\Omega \subseteq \mathbb{R}^n$ is a convex closed set.

*This author's research was supported by the National Natural Science Foundation of China under grant 11201382, the youth fund of humanities and social sciences of the ministry of education under grant 12YJC910008 and was supported by the Fundamental Research Funds for the Central Universities under grant SWJTU12CX055 and SWJTU11ZT29.

## 1.2 Application Background

In Compressive Sensing, also called Compressed Sensing or Compressive Sampling, a piece of sparse digital signal is encoded as inner products between the signal and a set of random (or random-like) vectors where the number of such inner products, or linear measurements, can be significantly less than the length of the signal. In other words, we acquire the data in a compressed way such that the measuring/observing cost can be greatly reduced. However, the decoding process is not a easy task since it requires finding a sparse solution, either exact or approximate, to an underdetermined linear system. What makes such a scheme work is the sparsity; *i.e.*, the original signal must have a sparse or compressible representation under some known basis. In this paper we work over the field of real numbers in order to simplify the notations, although the algorithms considered can adapt to complex data with minor modification. Let $\bar{x} \in \mathbb{R}^n$ be an original sparse signal that we wish to capture. Without loss of generality, we assume that $\bar{x}$ is sparse under the canonical basis, *i.e.*, the number of nonzero components in $\bar{x}$, denoted by $\|\bar{x}\|_0$, is far less than its length. Instead of sampling $\bar{x}$ directly, in CS one first obtains a set of linear measurements via

$$b = A\bar{x} \in \mathbb{R}^m, \tag{1.2}$$

where $A \in \mathbb{R}^{m\times n}$ with $m < n$. The original signal $\bar{x}$ is then reconstructed from the underdetermined linear system $Ax = b$ via certain reconstruction technique.

The critical step in CS is to recover the original signal $\bar{x}$ from the observation $b$ where $b$ is obtained by (1.2). The basic CS theory presented in [7, 8, 12] states that it is extremely probable to reconstruct $\bar{x}$ accurately or even exactly from $b$ provided that $\bar{x}$ is sufficiently sparse (or compressible) relative to the number of measurements, and the encoding matrix $A$ possesses certain desirable attributes. To make CS successful, two ingredients must be addressed carefully. First, a sensing matrix $A$ must be designed so that the compressed measurement $b = A\bar{x}$ contains enough information for a successful recovery of $\bar{x}$. Second, an efficient, stable and robust reconstruction algorithm must be available for recovering $\bar{x}$ from $A$ and $b$. In the present paper, we will only concentrate on the second aspect. In order to recover the sparse signal $\bar{x}$ from the underdetermined system (1.2), one could naturally consider seeking among all solutions of (1.2) the sparsest one, *i.e.*, solving

$$\min_x \{\|x\|_0 \ \mid \ Ax = b\}, \tag{1.3}$$

where $\|x\|_0$ is the number of nonzeros in $x$. Indeed, with overwhelming probability decoder (1.3) can recover sparse signals exactly from a very limited number of random measurements (see *e.g.*, [3]). Unfortunately, this $\ell_0$ problem is combinatorial and generally computationally intractable. A fundamental decoding model in the CS is the so-called basis pursuit (BP) problem [9]:

$$\min_x \{\|x\|_1 \ \mid \ Ax = b\}. \tag{1.4}$$

Minimizing the $\ell_1$-norm in (1.4) plays a central role in promoting solution sparsity. In fact, problem (1.4) shares the common solutions with (1.3) under some favorable conditions (see, for example, [13]). When $b$ contains i.i.d. Gaussian noise, or when $\bar{x}$ is not exactly sparse but only compressible, as are the cases in most practical applications, certain relaxation to the equality constraint in (1.4) is desirable. In such situations, common relaxations to (1.4) include the constrained "basis pursuit denoising" (BPDN) problem [9]:

$$\min_x \{\|x\|_1 \ \mid \ \|Ax - b\|_2 < \delta\}. \tag{1.5}$$

However, the more frequently used model usually is such an unconstrained convex optimization problem which has two terms in the objective function:

$$\min_x \{\tau\|x\|_1 + \frac{1}{2}\|Ax - b\|_2\}, \tag{1.6}$$

where $\delta, \tau > 0$ are fixed parameters. From the optimization theory, it is well known that problems (1.5) and (1.6) are equivalent in the sense that solving one will determine a parameter value in the other so that the two share the same solution. As $\delta$ and $\tau$ approach zero, both (1.5) and (1.6) converge to (1.4).

Note the BPDN model (1.6) is in the form of optimization problem (1.1), where $\theta(x) = \tau\|x\|_1$, $q(x) = \frac{1}{2}\|Ax - b\|_2$, and $\Omega = \mathbb{R}^n$, hence, it is simply an application of (1.1). Not limit to that, the problem (1.1) can also be applied in, such as image restoration [28].

Finally, we mention that for the CS problem, aside from $\ell_1$-related decoders, there exist alternative decoding techniques such as greedy algorithms (*e.g.*, see [23]) which, however, do not belong to optimization method, thus they are not a subject of concern in the present paper although they could perform well in experiments.

### 1.3 Optimality Condition and Subproblems

Although the application background introduced is mainly from the CS, however, we consider the general model (1.1).

In the last few years, the algorithms for finding solutions of (1.1) have been extensively studied, largely because solving such problems constitutes a critical step in an emerging methodology in digital signal processing such as CS. Before introducing the algorithm for solving it, we need to derive its optimality condition.

We assume that the solution set of (1.1), denoted by $\Omega^*$, is nonempty. For any $x^* \in \Omega^*$, we have

$$\theta(x') - \theta(x^*) + (x' - x^*)^T(Hx^* + c) \geq 0, \qquad \forall\, x' \in \Omega, \tag{1.7}$$

or equivalently:

$$(x' - x^*)^T\{s(x^*) + (Hx^* + c)\} \geq 0, \qquad \forall\, x' \in \Omega. \tag{1.8}$$

where $s(x) \in \partial(\theta(x))$ is the subgradient of $\theta(x)$ defined by

$$\xi \in \partial(\theta(x)) \;\; \Leftrightarrow \;\; \theta(y) - \theta(x) \geq \xi^T(y - x), \;\; \forall\, y \in \mathbb{R}^n.$$

For any symmetric matrix $M \in \mathbb{R}^{n\times n}$, we use $M \succ 0$ to denote that $M$ is positive definite. It is clear that, for any given $x^k$ and symmetric matrix $M$ where $M + H \succ 0$, the minimization problem

$$\min\{\phi_k(x) = \theta(x) + q(x) + \frac{1}{2}(x - x^k)^T M(x - x^k) \;\mid\; x \in \Omega\} \tag{1.9}$$

has a unique solution. Note the (1.9) can also be treated as: Finding $x \in \Omega$ such that

$$(x' - x)^T\{s(x) + (Hx + c) + M(x - x^k)\} \geq 0, \quad \forall x' \in \Omega. \tag{1.10}$$

Let the solution of (1.9) be the new iterate, the algorithm based on subproblem (1.9) is a "Proximal Point Algorithm" (PPA) when $M \succ 0$.

The subproblem (1.9) can be difficult to solve for a general $M$. However, when $\theta(x)$ is a separable function, we can exploit the separable structure of $\theta(x)$ to solve (1.9) cheaply. By choosing

$$M = rI - H \qquad \text{with} \qquad r > 0, \tag{1.11}$$

the (1.9) is reduced to such a separable problem:

$$\min \{\phi_k(x) = \theta(x) + (x - x^k)^T(Hx^k + c) + \frac{r}{2}\|x - x^k\|^2 \ \mid \ x \ \in \Omega\}, \tag{1.12}$$

which is equivalent to: Finding $x \in \Omega$ satisfying

$$(x' - x)^T\{s(x) + (Hx^k + c) + r(x - x^k)\} \geq 0, \quad \forall x' \ \in \Omega. \tag{1.13}$$

The problem (1.12) can have a closed form solution which is easy to obtain. For instance, the Basis Pursuit DeNoising (BPDN) model is in the form of (1.1) with $\theta(x) = \|x\|_1$ and $H = A^TA$, and the solution of (1.12) can be given by a soft shrinkage (see (4.6)), hence, the algorithm based on (1.12) can be called "Iterative Shrinkage/Thresholding" (IST) method [11]. Throughout this paper, we assume that the subproblem (1.12) is easy to solve, hence, it is meaningful to develop new efficient algorithms based on (1.12).

Beyond the IST viewpoint, and the PPA perspective mentioned above, the algorithm based on subproblem (1.12) can also be interpreted as a forward-backward splitting method [10]. In fact, when $\Omega = \mathbb{R}^n$ the problem (1.1) can be equivalent to: Finding $x \in \Omega$ such that

$$A(x) + B(x) \ni 0,$$

where $A(x) = \partial\theta(x)$ and $B(x) = \nabla q(x)$. The forward-backward splitting method is to iterate by:

$$x^{k+1} = (I + \frac{1}{r}A)^{-1}[(I - \frac{1}{r}B)(x^k)], \quad \text{with} \quad r > 0$$

where $(I - \frac{1}{r}B)(x^k)$ is a forward step (here it is a gradient step), and the operation associated with $(I + \frac{1}{r}A)^{-1}[\cdot]$ is a backward step (it can be a shrinkage step). It is easy to verify that the above iterating scheme for obtaining $x^{k+1}$ is equivalent to solving (1.12).

In order to derive a stopping criterion for the algorithm based on (1.12), we need to bridge (1.12) with the optimality condition of (1.1): For any given $r > 0$, let $\tilde{x}^k$ be the solution of (1.12) from given $x^k \in \mathbb{R}^n$, it is easy to verify that:

$$x^k \quad \text{is the solution of (1.1)} \quad \Longleftrightarrow \quad x^k = \tilde{x}^k.$$

Accordingly, we can define such an error function:

$$e(x^k, 1/r) := x^k - \tilde{x}^k, \tag{1.14}$$

which measures how much $x^k$ fails to be in the solution set of (1.1). Smaller $\|e(x^k, 1/r)\|$ can indicate that $x^k$ is closer to the solution set.

In addition, let $G \in \mathbb{R}^{n\times n}$ be any symmetric positive definite matrix. For any $x^* \in \Omega^*$, if the sequence $\{x^k\}$ generated by a method satisfies

$$\|x^{k+1} - x^*\|_G^2 < \|x^k - x^*\|_G^2, \tag{1.15}$$

the method is called a contraction method, and the sequence $\{x^k\}$ is said to be Fejér monotonically contractive to the solution set.

### 1.4 Projection and Contraction (PC) Methods

In this subsection, we introduce two Projection and Contraction (PC) methods from [18] which were originally proposed for the variational inequalities, and they are now applied to solve the structured constrained optimization problem (1.1). They were originally referred

as the Relaxed Method I and Relaxed Method 4 in [18]. However, they will be called PC method I and PC method Ⅱ, respectively for simplicity.

For any fixed $r > 0$, the PC method I iterates as follows:

**PC Method I** [18]

**1.** Compute the predicting point by:

$$\tilde{x}^k = \arg\min\{\phi_k(x) = \theta(x) + (x - x^k)^T(Hx^k + c) + \frac{r}{2}\|x - x^k\|^2 \mid x \in \Omega\}.$$

**2.** Generate the new iterate $x^{k+1}$ by

$$x^{k+1} = x^k - \alpha(x^k - \tilde{x}^k), \tag{1.16a}$$

where

$$\alpha = \gamma\alpha_k^*, \qquad \alpha_k^* = \frac{\|x^k - \tilde{x}^k\|_2^2}{\|x^k - \tilde{x}^k\|^2 + \frac{1}{r}\|x^k - \tilde{x}^k\|_H^2}, \qquad \text{and} \qquad \gamma \in (0, 2). \tag{1.16b}$$

Similar to the above method, paper [18] also provided another meaningful method which iterates as follows:

**PC Method Ⅱ** [18]

**1.** Compute the predicting point by:

$$\tilde{x}^k = \arg\min\{\phi_k(x) = \theta(x) + (x - x^k)^T(Hx^k + c) + \frac{r}{2}\|x - x^k\|^2 \mid x \in \Omega\}.$$

**2.** Generate the new iterate $x^{k+1}$ by

$$x^{k+1} = x^k - \gamma(I + \frac{1}{r}H)^{-1}(x^k - \tilde{x}^k), \qquad \gamma \in (0, 2). \tag{1.17}$$

The above two methods are still based on (1.12), however, compared with the classic PPA, one distinguished feature of these two methods is that they can use any $r > 0$, hence, the $M$ may not be positive definite. In addition, instead of taking the solution of (1.12) as the new iterate as in the classic PPA, they treat it as a predicting point, and generate the new iterate by (1.16a) or (1.17), respectively.

The convergence results of the above methods can be obtained from the following theorems whose proofs are similar to that in [17, 18], thus omitted. In fact, these theorems are also the base of our new algorithm.

**Theorem 1.1.** *For any given $r > 0$, let $\tilde{x}^k$ be the solution of* (1.12) *from given $x^k$. Then for any $x^* \in \Omega^*$, we have*

$$(x^k - x^*)^T(I + \tfrac{1}{r}H)(x^k - \tilde{x}^k) \geq \|x^k - \tilde{x}^k\|_2^2 + \frac{1}{r}\|x^k - x^*\|_H^2. \tag{1.18}$$

**Theorem 1.2.** *For any fixed $r > 0$, let $\tilde{x}^k$ be the solution of* (1.12) *from given $x^k$, and the new iterate $x^{k+1}$ is given by PC method I, then for any $x^* \in \Omega^*$, we have*

$$\|x^{k+1} - x^*\|^2_{(I+\frac{1}{r}H)} \leq \|x^k - x^*\|^2_{(I+\frac{1}{r}H)} - \gamma(2-\gamma)\alpha_k^*\|x^k - \tilde{x}^k\|^2, \tag{1.19}$$

*where $\alpha_k^*$ was defined in* (1.16b).

**Theorem 1.3.** *For any given $r > 0$, if $\tilde{x}^k$ is the solution of* (1.12) *from given $x^k$, and the new iterate $x^{k+1}$ is given by PC Method II* (1.17)*, then for any $x^* \in \Omega^*$, we have*

$$\|x^{k+1} - x^*\|^2_N \leq \|x^k - x^*\|^2_N - \gamma(2-\gamma)\|x^k - \tilde{x}^k\|^2 \quad \textit{with} \quad N = (I + \tfrac{1}{r}H)^2. \tag{1.20}$$

For an algorithm based on (1.12), our limited numerical experience shows that the choice of $r$ can be critical to the convergence rate. In general, smaller $r$ can yield faster convergence rate. However, in the classic PPA, $M$ is required to be positive definite, or equivalently $r > \lambda_{\max}(H)$ according to (1.11), where this restriction on $r$ can be too conservative. In the PC methods above, the condition on $r$ is relaxed such that we can use any $r > 0$, however, they still have limitations. For instance, they still adopt a fixed $r$, while the "optimal" choice of $r$ in each iteration can vary. In addition, it is still unclear how to choose $r$ in the PC methods, *e.g.*, in the PC method I, although we can use arbitrary small $r > 0$, however, smaller $r$ does not necessarily produce better performance since $\alpha_k^*$ in (1.16b) can be small when $r$ is small, and this could, to large extent, offset the advantage of using small $r$.

In order to exploit the potential ability of the algorithm based on (1.12), we propose a new method called "Self-Adaptive Projection and Contraction" (SA-PC) method which allows $r$ to vary, and this is the main contribution of this paper. Like the above PC methods, the new algorithm is not a classic PPA since the matrix $M$ may not be positive definite. Since its convergence proof is related to that of the PC methods above, therefore, this method is still refereed as a PC method.

The rest of this paper is organized as follows. First, we give some preliminaries about projection mapping in Section 2. In Section 3, we present our new algorithm as well as its convergence result. In Section 4, we elaborate the implementation of our new algorithm in the application of CS problems. In Section 5, we report the numerical results of our method in CS problems. In the last section, we draw some concluding remarks. Finally, two Matlab codes of our algorithms for the CS problem were shown in the appendix section.

## 2 Preliminaries

In this section, we summarize some basic properties and related definitions that will be used in the following discussions. For given $v \in \mathbb{R}^n$, the solution of problem

$$\min\{\|u - v\|_2 \mid u \in \Omega\}$$

is called the projection of $v$ on $\Omega$ under Euclidean-norm, denoted by $P_\Omega(v)$. In other words,

$$P_\Omega(v) = \operatorname{argmin}\{\|u - v\|_2 \mid u \in \Omega\}.$$

Since $\Omega$ is convex and closed, the projection on to $\Omega$ is unique. From the above definition, it follows that

$$(v - P_\Omega(v))^T(u - P_\Omega(v)) \leq 0, \quad \forall v \in \mathbb{R}^n, \forall u \in \Omega.$$

Consequently, we have

$$\|P_\Omega(v) - P_\Omega(w)\|_2 \leq \|v - w\|_2, \qquad \forall v, w \in \mathbb{R}^n$$

and

$$\|u - P_\Omega(v)\|_2^2 \le \|v - u\|_2^2 - \|v - P_\Omega(v)\|_2^2, \qquad \forall v \in \mathbb{R}^n, \forall u \in \Omega.$$

These properties of the projection mapping can be found in textbooks, e.g., [2].

**Lemma 2.1.** *Let $\Omega$ be a nonempty closed convex set of $\mathbb{R}^n$ and $F$ be a mapping from $\mathbb{R}^n$ into itself. Then $x$ is a solution of the variational inequality*

$$x \in \Omega, \quad (x' - x)^T F(x) \ge 0, \quad \forall x' \in \Omega$$

*if and only if*

$$x = P_\Omega[x - \beta F(x)], \quad \forall\, \beta > 0.$$

*Proof.* The result is a special case of Theorem 1 in [19]. □

According to Lemma 2.1, let $\tilde{x}^k$ be the solution of (1.12) from given $x^k$, we have

$$\tilde{x}^k = P_\Omega[x^k - \frac{1}{r}(s(\tilde{x}^k) + (Hx^k + c))]. \tag{2.1}$$

It should be mentioned:

- If $\theta(x)$ is a smooth function and we take $x^{k+1} = \tilde{x}^k$ as the new iterate, from (2.1) we have
$$x^{k+1} = P_\Omega[x^k - \frac{1}{r}(\nabla\theta(x^{k+1}) + \nabla q(x^k))].$$
For the problem (1.1), it is different either from the PPA
$$x^{k+1} = P_\Omega[x^k - \frac{1}{r}(\nabla\theta(x^{k+1}) + \nabla q(x^{k+1}))],$$
or from the projected gradient method [21]
$$x^{k+1} = P_\Omega[x^k - \frac{1}{r}(\nabla\theta(x^k) + \nabla q(x^k))].$$
- If $q(x) = 0$, (2.1) is reduced to $\tilde{x}^k = P_\Omega[x^k - \frac{1}{r}s(\tilde{x}^k)]$. The method adopts $\tilde{x}^k$ as a new iterate, which is the PPA for the problem $\min\{\theta(x) \,|\, x \in \Omega\}$. In this case, for any $r > 0$, the PPA is convergent.
- If $\theta(x) = 0$, (2.1) is reduced to $\tilde{x}^k = P_\Omega[x^k - \frac{1}{r}(Hx^k + c)]$. The method adopts $\tilde{x}^k$ as a new iteration, which is the projected gradient method for the problem $\min\{q(x) \,|\, x \in \Omega\}$. In this case, $r > \|H\|/2$ will guarantee the convergence, and $e(x, 1/r) = x - \tilde{x}$ (see (1.14)) coincides with the definition in the PC methods for linear variational inequalities [17, 18].

**Lemma 2.2.** *For given $x^k$ and symmetric matrix $M$, let $\tilde{x}^k$ be the solution of* (1.9)*. Then we have*

$$\phi(\tilde{x}^k) \le \phi(x^k) - \|x^k - \tilde{x}^k\|^2_{(M + \frac{1}{2}H)}. \tag{2.2}$$

*Proof.* . Since $\phi(x) = \theta(x) + \frac{1}{2}x^T Hx + c^T x$, we obtain

$$\begin{aligned}\phi(x^k) - \phi(\tilde{x}^k) &= (\theta(x^k) - \theta(\tilde{x}^k) + (x^k - \tilde{x}^k)^T(H\tilde{x}^k + c)) + \frac{1}{2}(x^k - \tilde{x}^k)^T H(x^k - \tilde{x}^k) \\ &\geq (x^k - \tilde{x}^k)^T\{s(\tilde{x}^k) + (H\tilde{x}^k + c)\} + \frac{1}{2}\|x^k - \tilde{x}^k\|_H^2. \end{aligned} \tag{2.3}$$

Observe the first part of the right hand side of (2.3) and set $x = x^k$ in (1.10), we have

$$(x^k - \tilde{x}^k)^T\{s(\tilde{x}^k) + (H\tilde{x}^k + c)\} \geq \|x^k - \tilde{x}^k\|_M^2. \tag{2.4}$$

The assertion (2.2) follows from (2.3) and (2.4) immediately. □

## 3 Self-Adaptive Projection and Contraction (SA-PC) Method

For any $r^0 > 0$ and fixed $\delta \in (0, 1)$, $\nu, \mu > 0$, the proposed method iterates as follows:

**Self-adaptive PC (SA-PC) Method**

**1.** Compute $\tilde{x}^k$ by:

$$\tilde{x}^k = \arg\min\{\phi_k(x) = \theta(x) + (x - x^k)^T(Hx^k + c) + \frac{r^k}{2}\|x - x^k\|^2 \mid x \in \Omega\}.$$

**2. While** the following condition does not hold:

$$t^k = \frac{\|x^k - \tilde{x}^k\|_H^2}{r^k\|x^k - \tilde{x}^k\|_2^2} \leq 2(1 - \delta). \tag{3.1}$$

**Do** backtracking by $r^k = r^k * t^k * \mu$;

recompute $\tilde{x}^k$ by

$$\tilde{x}^k = \arg\min\{\phi_k(x) = \theta(x) + (x - x^k)^T(Hx^k + c) + \frac{r^k}{2}\|x - x^k\|^2 \mid x \in \Omega\}.$$

**End**

**3.** Generate the new iterate by $x^{k+1} = \tilde{x}^k$, and prepare $r^{k+1}$ for the next iteration by

$$r^{k+1} = \min(\lambda_{\max}(H)/2, t^k/\nu) \text{ (or } r^{k+1} = t^k/\nu, \text{ when } \lambda_{\max}(H) \text{ is unknown}).$$

Compared with the classic PPA and the PC methods I and II, the most identical feature of the proposed SA-PC method is that it allows $r$ to vary dynamically in each step while other algorithms could use a fixed $r$. However, how to choose $r^k$ properly remains critical to the performance of our method. We suggest that $\tilde{x}^k$ should satisfy such a condition

$$\|x^k - \tilde{x}^k\|_H^2 \approx r^k\|x^k - \tilde{x}^k\|_2^2 \qquad \text{and} \qquad \|x^k - \tilde{x}^k\|_H^2 < 2r^k\|x^k - \tilde{x}^k\|_2^2,$$

where the latter one is a relaxed condition of $r^k > \lambda_{\max}(H)/2$ and we need to select $r^k$ dynamically to meet this condition. For doing that, in the SA-PC method, we generate $r^{k+1}$ by the Barzilai-Borwein (BB) [15] step size which was recommended in [27]. In the backtracking step, we adjust the $r^k$ by $r^k = r^k * t^k * \mu$ where the backtracking factor $t^k * \mu$ is self-adaptively determined by how fails the above condition to be satisfied. Numerical evidence shows that these two techniques can be critical to the performance of our method, although how to choose $r^k$ better remains an open question, and merits further research on it.

Then we are at the stage to derive the convergence result of the SA-PC method. First, it is easy to verify that, when $\|H\| \leq r^k(1-\delta)$ holds, the condition (3.1) can be satisfied. This means the SA-PC method can be realized. Then note that in both proofs of Theorem 1.2 and 1.3, we use the inequality (1.18) by ignoring the last term of its right hand side, *i.e.*, $\frac{1}{r}(x^k - x^*)^T H(x^k - x^*)$. However, if we retain this term in the proof, the contractive property of the generated sequence $\{\|x^k - x^*\|_2^2\}$ can still be derived under the relaxed condition (3.1).

**Theorem 3.1.** *For given symmetric matrix $M_k = r^k I - H$ with $r^k > 0$, if $\tilde{x}^k$ be the solution of* (1.9) *and the condition* (3.1) *is fulfilled. Let the sequence $\{x^k\}$ be generated by the SA-PC method. Then for any $x^* \in \Omega^*$, we have*

$$\|x^{k+1} - x^*\|_2^2 \leq \|x^k - x^*\|_2^2 - \delta\|x^k - \tilde{x}^k\|^2. \tag{3.2}$$

*Proof.* Since $M_k = r^k I - H$, similar to (1.18), we have

$$(x^k - x^*)^T(I + \tfrac{1}{r^k}H)(x^k - \tilde{x}^k) \geq \|x^k - \tilde{x}^k\|_2^2 + \tfrac{1}{r^k}\|x^k - x^*\|_H^2. \tag{3.3}$$

When the term $\frac{1}{r^k}\|x^k - x^*\|_H^2$ in (3.3) was not ignored, we get

$$\begin{aligned}
&\|x^{k+1} - x^*\|^2_{(I+\frac{1}{r^k}H)} \\
&= \|(x^k - x^*) - (x^k - \tilde{x}^k)\|^2_{(I+\frac{1}{r^k}H)} \\
&= \|x^k - x^*\|^2_{(I+\frac{1}{r^k}H)} - 2(x^k - x^*)^T(I + \tfrac{1}{r^k}H)(x^k - \tilde{x}^k) + \|x^k - \tilde{x}^k\|^2_{(I+\frac{1}{r^k}H)} \\
&\leq \|x^k - x^*\|^2_{(I+\frac{1}{r^k}H)} - 2\|x^k - \tilde{x}^k\|_2^2 - \tfrac{2}{r^k}\|x^k - x^*\|_H^2 + \|x^k - \tilde{x}^k\|^2_{(I+\frac{1}{r^k}H)} \\
&= \|x^k - x^*\|_2^2 - \|x^k - \tilde{x}^k\|_2^2 - \tfrac{1}{r^k}\|x^k - x^*\|_H^2 + \frac{1}{r^k}\|x^k - \tilde{x}^k\|_H^2.
\end{aligned} \tag{3.4}$$

Since $x^{k+1} = \tilde{x}^k$, subtracting $\frac{1}{r^k}\|\tilde{x}^k - x^*\|_H^2$ from the both sides of the above inequality, we obtain

$$\begin{aligned}
\|x^{k+1} - x^*\|_2^2 \leq\ & \|x^k - x^*\|_2^2 - \|x^k - \tilde{x}^k\|_2^2 + \tfrac{1}{r^k}\|x^k - \tilde{x}^k\|_H^2 \\
& - \tfrac{1}{r^k}(\|x^k - x^*\|_H^2 + \|\tilde{x}^k - x^*\|_H^2).
\end{aligned} \tag{3.5}$$

By using the identity $\|u\|_2^2 + \|v\|_2^2 = \frac{1}{2}(\|u - v\|_2^2 + \|u + v\|_2^2)$, we have

$$\|x^k - x^*\|_H^2 + \|\tilde{x}^k - x^*\|_H^2 = \frac{1}{2}(\|x^k - \tilde{x}^k\|_H^2 + \|x^k + \tilde{x}^k - 2x^*)\|_H^2). \tag{3.6}$$

Substituting (3.6) to (3.5), we obtain

$$\|x^{k+1} - x^*\|_2^2 \leq \|x^k - x^*\|_2^2 - \|x^k - \tilde{x}^k\|_2^2 + \frac{1}{2r^k}\|x^k - \tilde{x}^k\|_H^2.$$

The assertion of this theorem follows from the above inequality and (3.1) immediately. □

Besides the contractive property of the sequence $\{\|x^k - x^*\|_2^2\}$, the following theorem indicates that the sequence $\{\phi(x^k)\}$ generated by the SA-PC method is monotonically decreasing.

**Theorem 3.2.** *For given $x^k$ and symmetric matrix $M_k = r^k I - H$, let $\tilde{x}^k$ be the solution of (1.9), then we have*

$$\phi(x^{k+1}) \leq \phi(x^k) - \delta r^k \|x^k - \tilde{x}^k\|^2. \tag{3.7}$$

*Proof.* First, since $x^{k+1} = \tilde{x}^k$, it follows from Lemma 2.2 that

$$\phi(x^{k+1}) \leq \phi(x^k) - \|x^k - \tilde{x}^k\|^2_{(M_k + \frac{1}{2}H)}. \tag{3.8}$$

By $M_k = r^k I - H$ and (3.1), we have

$$\|x^k - \tilde{x}^k\|^2_{(M_k+\frac{1}{2}H)} = r^k \|x^k - \tilde{x}^k\|^2 - \frac{1}{2}\|x^k - \tilde{x}^k\|^2_H \geq \delta r^k \|x^k - \tilde{x}^k\|^2. \tag{3.9}$$

Substituting (3.9) to (3.8) we get the assertion (3.7) immediately. □

## 4 Application in Compressive Sensing (CS) Problems

While the application range of problem (1.1) is potentially broad, we only focus on the BPDN model (1.6) which is often encountered in the CS problem. For convenience, we write the BPDN model (1.6) again

$$\min_x \{\phi(x) = \tau\|x\|_1 + \frac{1}{2}\|Ax - b\|_2^2\}, \tag{4.1}$$

where $\tau > 0$, $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. It is clear that the problem (4.1) has a unique solution $x^*$ which satisfies

$$0 \in \tau\partial(\|x^*\|_1) + A^T(Ax^* - b) \tag{4.2}$$

and there is a $s(x^*) \in \partial(\|x^*\|_1)$ such that

$$\tau s(x^*) + A^T(Ax^* - b) = 0. \tag{4.3}$$

In the last few years, there have been abundant works searching for efficient algorithms for solving (4.1), *e.g.*, projection gradient method [14], interior point method [20], augmented Lagrangian method [1], and the iterative shrinkage/thresholding [4, **?**, 16, 25, 26, 27]. To exploit the separable structure of $\|x\|_1$, some of the above algorithms suggest to solve

$$\arg\min_x \{(x - x^k)^T(A^T(Ax^k - b)) + \frac{r}{2}\|x - x^k\|_2^2 + \tau\|x\|_1\}. \tag{4.4}$$

In fact, the solution of (4.4) is given by

$$\tilde{x}^k = d^k - \max(\min(d^k, a), -a), \tag{4.5a}$$

where

$$d^k = x^k - \frac{1}{r}A^T(Ax^k - b) \qquad \text{and} \qquad a = \frac{\tau}{r}e, \qquad \text{and} \qquad e \text{ is all-one vector.} \tag{4.5b}$$

Note (4.5) can be rewritten into a shrinkage form

$$\tilde{x}^k = sign(d^k) .* \mathcal{S}(d^k, \frac{r}{\tau}) \tag{4.6a}$$

where

$$d^k = \frac{r}{\tau}x^k - \frac{1}{\tau}A^T(Ax^k - b). \tag{4.6b}$$

For given $x^k$ and $r > 0$, it is easy to obtain the solution of (4.4) via (4.6), hence, the algorithm based on (4.4) belongs to the Iterative Shrinkage/Thresholding method, and the main computation cost is two times of matrix-vector multiplications.

The Matlab code of the SA-PC method for the problem (4.1) can be found in the appendix section. In which the parameters such as `mu`, `nu`, and `\delta` have been assigned with proper values. The initial $r$ was set to be 1, however, when $\lambda_{\max}(AA^T)$ is known, we suggest to initialize $r$ with $\lambda_{\max}(AA^T)/2$.

In addition, our SA-PC method can converge slowly for small $\tau$, *e.g.*, when $\tau = \|A^Tb\|_\infty \times 0.01$. As a remedy, we use a simple continuation technique to overcome this issue, see [16] for details. The corresponding code can also be found in the appendix section.

## 5 Numerical Experiments

This section includes the numerical results of our SA-PC method when applied in the CS problem (4.1), which is in the form of (1.1). All the tests were done on a computer with Core 2 P8700, 4G memory, Windows Vista basic and Matlab 2009a.

### Constructing the test examples

To construct the test examples of the CS problem (4.1), we only need to give the matrix $A$, the vector $b$ and the parameter $\tau$.

- **The matrix** $A$. First, we let $A$ be an $m \times n$ matrix whose entries are randomly generated in the interval $(-1, +1)$. Then each row of matrix $A$ is normalized. The following Matlab code is used to produce the matrix $A$.

```
A=rand(m,n)*2-1; for i=1:m  t=norm(A(i,:));  A(i,:)=A(i,:)/t; end;
```

- **The vector** $b$. We let the test problem have a known spar solution *xop* and set $b$ by $A * xop$. The following Matlab code is used to produce the vector $b$.

```
xop=sprand(n,1,spar); [ii,jj,kk]=find(xop);
for i=1:size(ii) xop(ii(i))=sign(xop(ii(i))*2-1); end;
b0=A*xop; b=b0.*(ones(m,1)+0.01*randn(m,1));
```

- As in the literature, we take $\tau = \|A^Tb\|_\infty \times 0.1$ and $\tau = \|A^Tb\|_\infty \times 0.01$, respectively.

To have a quick experience of the efficiency of our SA-PC method, we first did a simple test by comparing:

1. The classic PPA where $r = 1.02 \cdot \lambda_{\max}(AA^T)$. This method was used in [6] for the CS problem in parallel many-core architectures.
2. The PC method I from [18] with fixed $r = \frac{m}{n}\lambda_{\max}(AA^T)$.
3. The SA-PC method with $\nu$=0.85, $\mu$=1.

Note the PC Method Ⅱ was not included in this test example since its subproblem may not be cheap to compute when $AA^T \neq I$.

In order to compare the efficiency of the different methods fairly, all algorithms began with $x^0 = 0$, and stopped the iteration once

$$\|x^k - \tilde{x}^k\|_\infty \leq \varepsilon,$$

where $\tilde{x}^k$ is the solution of (4.4) from $x^k$. The infinity norm was used to make the tolerance $\varepsilon$ roughly irrelevant to the dimension of problem ($m$ and $n$).

### Test results

First, we ran the code with $\tau = \|A^T b\|_\infty \times 0.1$. For $\varepsilon = 10^{-3}$ and $\varepsilon = 10^{-4}$, Tables 5.1 and Table 5.2 report the number of iterations, number of matrix-vector multiplications (denoted by No.$l$ in the tables) and the CPU time (in Seconds), respectively.

Table 5.1. Test results for $\tau = \|A^T b\|_\infty \times 0.1$ and $\varepsilon = 10^{-3}$

| $m \times n$ Matrix | | No. nonzero | Classic PPA | | | PC Method I | | | SA-PC | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $m$ | $n$ | Elements | No.It | No.$l$ | CPU | No.It | No.$l$ | CPU | No.It | No.$l$ | CPU |
| 1024 | 4096 | 160 | 209 | 418 | 12.06 | 36 | 72 | 2.20 | 22 | 50 | 1.50 |
| 1600 | 8192 | 320 | 247 | 494 | 49.31 | 40 | 80 | 8.17 | 26 | 60 | 5.89 |
| 2000 | 12000 | 400 | 265 | 530 | 101.94 | 45 | 90 | 17.55 | 27 | 62 | 11.66 |

Table 5.2. Test results for $\tau = \|A^T b\|_\infty \times 0.1$ and $\varepsilon = 10^{-4}$

| $m \times n$ Matrix | | No. nonzero | Classic PPA | | | PC Method I | | | SA-PC | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $m$ | $n$ | Elements | No.It | No.$l$ | CPU | No.It | No.$l$ | CPU | No.It | No.$l$ | CPU |
| 1024 | 4096 | 160 | 316 | 632 | 18.23 | 50 | 100 | 2.98 | 29 | 67 | 1.99 |
| 1600 | 8192 | 320 | 536 | 1072 | 107.50 | 80 | 160 | 16.13 | 37 | 84 | 8.25 |
| 2000 | 12000 | 400 | 659 | 1318 | 253.36 | 99 | 198 | 38.34 | 42 | 97 | 18.20 |

From the preliminary numerical results, we see the average number of the matrix-vector multiplications in each iteration of the SA-PC method is about 2.3. In other words, its per-iteration cost is only around 15% more than that of the other two methods, while it needs much less iterations to reach a given accuracy, so its speed performance is satisfactory.

The SA-PC method is the fastest one among the tested algorithms. In details.

$$\frac{\text{The computational cost of the SA-PC}}{\text{The computational cost of the PC method I}} \approx 70\%, \qquad \text{when} \quad \varepsilon = 10^{-3}$$

and

$$\frac{\text{The computational cost of the SA-PC}}{\text{The computational cost of the PC method I}} \approx 50\%, \qquad \text{when} \quad \varepsilon = 10^{-4}.$$

The performance gap between the classic PPA and the SA-PC method can be more evident.

$$\frac{\text{The computational cost of the SA-PC}}{\text{The computational cost of the Classic PPA}} \approx 12\%, \qquad \text{when} \quad \varepsilon = 10^{-3}$$

and

$$\frac{\text{The computational cost of the SA-PC}}{\text{The computational cost of the Classic PPA}} \approx 8\%, \qquad \text{when} \quad \varepsilon = 10^{-4}.$$

We also got the test results for $\tau = \|A^T b\|_\infty \times 0.01$ as follows.

Table 5.3. Test results for $\tau = \|A^T b\|_\infty \times 0.01$ and $\varepsilon = 10^{-3}$

| $m \times n$ Matrix | | No. nonzero | Original PPA | | | PC Method I | | | SA-PC | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $m$ | $n$ | Elements | No.It | No.$l$ | CPU | No.It | No.$l$ | CPU | No.It | No.$l$ | CPU |
| 1024 | 4096 | 160 | 861 | 1722 | 50.7 | 204 | 408 | 12.0 | 90 | 206 | 6.0 |
| 1600 | 8192 | 320 | 877 | 1754 | 173.8 | 268 | 536 | 53.5 | 100 | 227 | 22.0 |
| 2000 | 12000 | 400 | 609 | 1218 | 245.2 | 174 | 348 | 70.7 | 71 | 159 | 32.1 |

Table 5.4. Test results for $\tau = \|A^T b\|_\infty \times 0.01$ and $\varepsilon = 10^{-4}$

| $m \times n$ Matrix | | No. nonzero | CLassic PPA | | | PC Method I | | | SA-PC | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $m$ | $n$ | Elements | No.It | No.$l$ | CPU | No.It | No.$l$ | CPU | No.It | No.$l$ | CPU |
| 1024 | 4096 | 160 | 967 | 1934 | 56.8 | 218 | 436 | 12.9 | 96 | 219 | 6.4 |
| 1600 | 8192 | 320 | —- | —- | —- | 622 | 1244 | 124.2 | 178 | 406 | 39.3 |
| 2000 | 12000 | 400 | —- | —- | —- | 936 | 1872 | 377.9 | 203 | 462 | 90.5 |

Under this unfavorable small setting of $\tau$, the performance gap between the classic PPA and the SA-PC method becomes a little larger. In details.

$$\frac{\text{The computational cost of the SA-PC}}{\text{The computational cost of the PC method I}} \approx 45\%, \qquad \text{when} \quad \varepsilon = 10^{-3}$$

and

$$\frac{\text{The computational cost of the SA-PC}}{\text{The computational cost of the PC method I}} \approx 30\%, \qquad \text{when} \quad \varepsilon = 10^{-4}.$$

The performance advantage of SA-PC method to the Classic PPA still remains.

$$\frac{\text{The computational cost of the SA-PC}}{\text{The computational cost of the Classic PPA}} \approx 12\%, \qquad \text{when} \quad \varepsilon = 10^{-3}$$

and

$$\frac{\text{The computational cost of the SA-PC}}{\text{The computational cost of the Classic PPA}} \approx 11\%, \qquad \text{when} \quad \varepsilon = 10^{-4}.$$

From the above experimental results, the SA-PC method has shown satisfactory performance when compared with some classic algorithms. However, in order to demonstrate it is really a competitive method, we need to compare it with some state-of-the-art algorithms.

For doing that, in the following tests, four state-of-the-art algorithms for the CS problems including GPSR-BB [14], FPC [16], FPC-AS [25, 26],and SpaRSA [27] were included. We use the latest versions of the above algorithms, *i.e.*, GPSR v6.0, FPC v2.0, FPC-AS v1.21, SpaRSA v2.0, where they were all implemented with the efficient BB step size and continuation strategy.

Note since the speed of an algorithm is, to large extent, dependent on the stopping criterion. In order to compare the speed of the algorithms in a way that is as independent as possible from the stopping criterion, the experimental protocol that we followed was the following: we first run SpaRSA with `StoppingCriterion=3` and `ToleranceA=1e-15` to obtain an "accurate" solution denoted by $\bar{x}$, and then run other algorithms until either the relative error to $\bar{x}$ defined by $\|x - \bar{x}\|/\|\bar{x}\|$ is below a prescribed *tol* or a maximal iteration number 500 is reached. To avoid the screen output which can slow down the algorithm speed, we use silent mode for all algorithms whenever possible, *i.e.*, `verbose`=0.

We first investigate the speed performance of the algorithms under two dimensional settings when $tol = 10^{-4}$. The average result of 10 random problems with $\tau = \|A^T b\|_\infty \times 0.1$ and $\tau = \|A^T b\|_\infty \times 0.01$ are shown in Table 5.5 and Table 5.6, respectively, where the `Error` denotes the final relative error.

Table 5.5. Test results for $\tau = \|A^T b\|_\infty \times 0.1$ and $tol = 10^{-4}$

| $m \times n$ Matrix | | No. nonzero | GPSRBB | | FPC | | FPC-AS | | SpaRSA | | SA-PC | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $m$ | $n$ | Elements | Error | CPU | Error | CPU | Error | CPU | Error | CPU | Error | CPU |
| 1024 | 4096 | 160 | 8.836e-5 | 3.215 | 5.883e-6 | 3.260 | 6.587e-5 | 2.211 | 6.176e-5 | 1.997 | 7.339e-5 | 2.162 |
| 3000 | 12000 | 360 | 8.421e-5 | 12.464 | 5.377e-6 | 14.966 | 2.988e-5 | 9.554 | 6.949e-5 | 10.590 | 7.472e-5 | 9.010 |

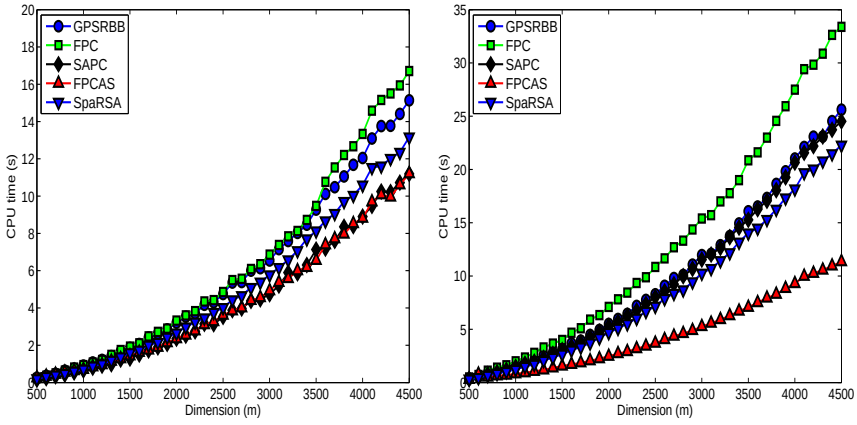
Table 5.6. Test results for $\tau = \|A^T b\|_\infty \times 0.01$ and $tol = 10^{-4}$

| $m \times n$ Matrix | | No. nonzero | GPSRBB | | FPC | | FPC-AS | | SpaRSA | | SA-PC | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $m$ | $n$ | Elements | Error | CPU | Error | CPU | Error | CPU | Error | CPU | Error | CPU |
| 1024 | 4096 | 160 | 8.750e-5 | 2.869 | 3.972e-5 | 2.920 | 6.374e-5 | 0.980 | 7.542e-5 | 1.986 | 7.354e-5 | 2.131 |
| 3000 | 12000 | 360 | 8.098e-5 | 11.967 | 8.060e-6 | 15.304 | 3.140e-6 | 5.302 | 5.657e-5 | 10.162 | 6.811e-5 | 9.574 |

We observe that, in most cases, the FPC-AS is the fastest algorithm among these algorithms. However, when $\tau = \|A^T b\|_\infty \times 0.1$, and $(m, n, k)$=(3000,12000,360), our SA-PC method becomes the fastest one. However, we also see that, when $\tau$ is small, *i.e.*, $\tau = \|A^T b\|_\infty \times 0.01$, the advantage of our method is not so evident even implemented with the continuation technique, and this could be a major drawback of our SA-PC method. But even under this unfavorable setting of $\tau$, *i.e.*, the performance of our method is still competitive.

To investigate the performance for different dimensional settings more precisely, we did another test by varying $m$ between 500 and 3500 with interval 100. For each setting of $m$, we set $n = 4m$, $k = floor(0.03n)$, $\tau = \|A^T b\|_\infty \times 0.1$, $tol = 10^{-4}$, and run ten random problems. The average results are shown in Figure 1.

Figure 1: Results of different dimensional settings when $n = 4m$, $k = floor(0.03n)$, $tol = 10^{-4}$. Left subfigure: $\tau = \|A^T b\|_\infty \times 0.1$; right subfigure: $\tau = \|A^T b\|_\infty \times 0.01$,




As can be seen from the Figure 1, the competitive performance of the SA-PC method remains as the dimension increases. Specially, when $\tau = \|A^T b\|_\infty \times 0.1$ as shown in the left subfigure, its performance can be roughly the same compared with FPC-AS when the

dimension is large. This indicates that the SA-PC method is suitable for handling large-scale problems.

In this section, we presented experimental results to compare the proposed SA-PC method with some classic methods as well as some state-of-the-art algorithms. Our numerical results show that the proposed algorithm is efficient and competitive. Specially, the proposed algorithm can handle large-scale problems well, and this can be even evident when $\tau$ is not small. However, even when $\tau$ is small, our algorithm is still competitive when compared with those state-of-the-art algorithms.

## 6 Conclusions Remarks

In this paper, we present a new method called SA-PC method for minimizing the sum of a convex and a quadratic function in the form of (1.1). Its application background is from the CS problem, but its application range is not limited to that. Its theory is based on some existing PC methods for symmetric linear variational inequalities, see [17, 18], although it can also be interpreted from, *e.g.*, the forward-backward splitting perspective. The most identical feature of the proposed algorithm is that it allows $r$ to vary in a self-adaptive way to produce a larger step size to increase the algorithm speed.

When applied in the CS problem, our experimental results show that the proposed algorithm has competitive performance to some state-of-the-art algorithms dedicated for the CS problem like FPC-AS and SpaRSA. Despite how good its performance is, our algorithm can still be instantiated by different ways of choosing $r^k$, however, this is left to the future works.

## Acknowledgement

Many thanks to Professor B. S. He for his guidance and comments which greatly improved the quality of this paper.

## References

[1] M. Afonso, J. Bioucas-Dias and M. Figueiredo, Fast image recovery using variable splitting and constrained optimization, *IEEE Trans. Image Proces.* 19 (2010) 2345–2356.

[2] E. Blum and W. Oettli, *Mathematische Optimierung, Grundlagen und Verfahren*, Ökonometrie und Unternehmensforschung, Springer-Verlag, Berlin-Heidelberg-New York, 1975.

[3] D. Baron, M. Duarte, S. Sarvotham, M.B. Wakin and R. G. Baraniuk, Distributed compressed sensing, Available at: http://dsp.rice.edu/cs/DCS112005.pdf.

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

[5] J.M. Bioucas-Dias and M.A.T. Figueiredo, A New TwIST: Two-Step Iterative Shrinkage/Thresholding Algorithms for Image Restoration, *IEEE Trans. Image Proces.* 16 (2007) 2992–3004.

[6] A. Borghi, J. Darbon, S. Peyronnet, T.F. Chan and S. Osher, A simple compressive sensing algorithm for parallel many-core architectures, UCLA CAM Report 08-64, Available at: ftp://ftp.math.ucla.edu/pub/camreport/cam08-64.pdf.

[7] E. Candes, J. Romberg and T. Tao, Stable signal recovery from incomplete and inaccurate information, *Commun. Pure. Appl. Math.* 59 (2005) 1207–1233.

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

[9] S. S. Chen, D. L. Donoho and M. A. Saunders, Atomic decomposition by basis pursuit, *SIAM J. Sci. Comput.* 20 (1999) 33–61.

[10] P. Combettes and V. Wajs, Signal recovery by proximal forward-backward splitting, *Multiscale Model. Sim.* 4 (2005) 1168–1200.

[11] I. Daubechies, M. Defrise and C. De Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, *Commun. Pure. Appl. Math.* 57 (2004) 1413–1457.

[12] D. Donoho, Compressed sensing, *IEEE Trans. Inform. Theory* 52 (2006) 1289–1306.

[13] D. Donoho, For most large underdetermined systems of linear equations, the minimal $\ell_1$-norm solution is also the sparsest solution, *Commun. Pure Appl. Math.* 59 (2006) 907–934.

[14] M.A.T. Figueiredo, R.D. Nowak and S.J. Wright, Gradient Projection for Sparse Reconstruction: Application to Compressed Sensing and Other Inverse Problems, *IEEE J. Sel. Topics Signal Process.* 1 (2007) 586 – 597.

[15] R. Fletcher, On the Barzilai-Borwein method, Optimization and control with applications, *Applied Optimization* 96 (2005) 235–256.

[16] E. T. Hale, W. Yin and Y. Zhang. A Fixed-Point Continuation Method for l1-Regularized Minimization with Applications to Compressed Sensing, Rice University CAAM Technical Report, TR07-07, 2007.

[17] B. He, A new method for a class of linear variational inequalities, *Math. Program.* 66 (1994) 137–144.

[18] B. He, Solving a class of linear projection equations, *Numer. Math.* 68 (1994) 71–80.

[19] B. He, Inexact implicit methods for monotone general variational inequalities, *Math. Program.* 86 (1999) 199–217.

[20] S. Kim, K. Koh, M. Lustig, S. Boyd and D. Gorinvesky, An interior-point method for large-scale l1-regularized least squares, *IEEE Journal on Selected Topics in Signal Processing* 1 (2007) 606–617.

[21] Y. Nesterov, Gradient methods for minimizing composite objective function, CORE Discussion Paper 2007/76, Center for Operations Research and Econometrics (CORE), Catholic University of Louvain, Belgium, 2007.

[22] R.T. Rockafellar, Monotone operators and the proximal point algorithm, *SIAM J. Control Optim.* 14 (1976) 877–898.

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

[24] Y. Wang, J. Yang, W. Yin and Y. Zhang, A new alternating minimization algorithm for total variation image reconstruction, *SIAM J. Img. Sci.* 1 (2008) 248–272.

[25] Z. Wen, W. Yin, D. Goldfarb and Y. Zhang, A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization, and continuation, Rice University CAAM Technical Report, TR09-01, 2009.

[26] Z. Wen, W. Yin, H. Zhang and D. Goldfarb, On the convergence of an active set method for l1 minimization, Rice University CAAM Technical Report, TR10-22, 2010.

[27] S.J. Wright, R.D. Nowak and M.A.T. Figueiredo, Sparse reconstruction by separable approximation, *IEEE Trans. Signal Proces.* 57 (2009) 2479 – 2493.

[28] J. Yang, Y. Zhang and W. Yin, An efficient TVL1 algorithm for deblurring multichannel images corrupted by impulsive noise, *SIAM J. Sci. Comput.* 31 (2009) 2842–2865.

## Appendix

**The Matlab Code of the SA-PC Method for** (4.1)**.**

```
r = 1; stopc = 1; eps=1e-4; x = zeros(n,1);k = 0; Ax = A*x; l = 1; %%%%%%%%%
while(stopc>eps && k<maxit) k=k+1;                                  %%%%%%%%%
    x0= x;  Ax0=Ax;   d0=A'*(b-Ax0);    l=l+1;                     %% (1) %%
    d = d0 +x0*r;     x= (d - max(min(d,tau),-tau))/r;             %% (2) %%
    Ax=A*x;          l=l+1;                                       %% (3) %%
    ex= x0-x;        Aex=Ax0-Ax;                                  %% (4) %%
    T1=ex'*ex;       T2=Aex'*Aex;       TT=T2/(T1*r);             %% (5) %%
  while(TT > 1.9)   r=r*TT*1;                                     %% (6) %%
     d = d0 +x0*r;     x= (d - max(min(d,tau),-tau))/r;           %% (7) %%
     Ax=A*x;          l=l+1;                                      %% (8) %%
     ex= x0-x;        Aex=Ax0-Ax;                                 %% (9) %%
     T1=ex'*ex;       T2=Aex'*Aex;       TT=T2/(T1*r);            %%(10) %%
  end;                                                            %%(11) %%
  stopc = norm(x-x0,inf); r= T2*0.85/T1;                          %%(12) %%
end; %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

**The Matlab Code of the SA-PC Method for (4.1) with continuation.**

```
r = 1; stopc = 1; eps=1e-4; x = zeros(n,1);k = 0; Ax = A*x; l = 1; %%%%%%%%%
tauf = tau; tau = 0.1*norm(A'*b,inf); rr = exp(log(tau/tauf)/40);  %%%%%%%%%
while((stopc>eps && k<maxit)|| tau>tauf) k=k+1;                    %%%%%%%%%
    x0= x;  Ax0=Ax;   d0=A'*(b-Ax0);    l=l+1;                    %% (1) %%
    d = d0 +x0*r;     x= (d - max(min(d,tau),-tau))/r;            %% (2) %%
    Ax=A*x;          l=l+1;                                       %% (3) %%
    ex= x0-x;        Aex=Ax0-Ax;                                  %% (4) %%
    T1=ex'*ex;       T2=Aex'*Aex;       TT=T2/(T1*r);             %% (5) %%
  while(TT > 1.9)   r=r*TT*1;                                     %% (6) %%
     d = d0 +x0*r;     x= (d - max(min(d,tau),-tau))/r;           %% (7) %%
     Ax=A*x;          l=l+1;                                      %% (8) %%
     ex= x0-x;        Aex=Ax0-Ax;                                 %% (9) %%
     T1=ex'*ex;       T2=Aex'*Aex;       TT=T2/(T1*r);            %%(10) %%
  end;                                                            %%(11) %%
  stopc = norm(x-x0,inf); r = T2*0.85/T1;  tau = max(tau/rr,tauf); %%(12) %%
end; %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

*Manuscript received 8 September 2011*
*revised 2 March 2012*
*accepted for publication 11 April 2012*

Yuan Shen
Department of Mathematics, Nanjing University, Nanjing
210093, P. R. China
E-mail address: yshen7@gmail.com

Chengjing Wang
School of Mathematics, Southwest Jiaotong University
Chengdu, 610031, P. R. China
E-mail address: renascencewang@hotmail.com