Minimizing Nonconvex Functions for Sparse Vector Reconstruction

Nasser Mourad and James P. Reilly
Department of Electrical and Computer Engineering
McMaster University, Hamilton, Ont. CANADA L8S 4K1

***Abstract*— In this paper we develop a novel methodology for minimizing a class of non–convex (concave on the non–negative orthant) functions for solving under–determined linear system of equations $\boldsymbol{As} = \boldsymbol{x}$ when the solution vector $\boldsymbol{s}$ is known a priori to be sparse. The proposed technique is based on locally replacing the original objective function by a quadratic convex function which is easily minimized. The resulting algorithm is iterative and is absolutely converging to a fixed point of the original objective function. For a certain selection of convex objective functions, the class of algorithms called Iterative Reweighted Least Squares (IRLS) are shown to be a special case of the proposed methodology. Thus the proposed algorithms are a generalization and unification of the previous methods. In addition, we also propose a new class of algorithms with better convergence properties compared to the regular IRLS algorithms and hence can be considered as enhancements to these algorithms. Since the original objective functions are non-convex, the proposed algorithm is susceptible to convergence to a local minimum. To alleviate this difficulty, we propose a random perturbation technique that enhances the performance of the proposed algorithm. The numerical results show that the proposed algorithms outperform some of the well known algorithms that are usually utilized for solving the same problem.**

I. Introduction

There are many situations in science and technology that seek solutions to under–determined systems of equations, i.e. systems of linear equations with fewer equations than unknowns. Such problems are extremely common for a variety of reasons. For instance, the number of sensors may be small due to physical limitations as in breast cancer imaging, or the sensing process may be slow so that one can only measure the object a few times, as in MRI. Many other examples in inverse problems, array signal processing, and biomagnetic imaging all come to mind. This problem can be expressed mathematically as finding a unique solution to a linear under–determined systems of equations $\boldsymbol{As} = \boldsymbol{x}$, where $\boldsymbol{x} \in \mathbb{R}^m$ is the vector of measurements, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is the sensing (measuring) matrix, and $m < n$. However, it is known that a system of linear equations with fewer equations than unknowns has infinitely many solutions, and thus it is necessary to impose constraints on the candidate solution to identify which of these candidate solutions is the desired one.

This work was supported in part by the Egyptian Ministry of High Education and the Natural Sciences and Engineering Research Council of Canada (NSERC).

A powerful constraint that can be used in this regard is the "*sparsity*" of the solution vector. Sparsity means that the solution vector has few nonzero elements. The problem can be readily extended to the case of non–sparse solution vectors that are piecewise constant (hence its gradient is sparse [1]), or can be sparsely represented in some basis (e.g., Fourier basis and wavelet related basis [2]). The problem of reconstruction of sparse signals from a limited number of linear measurements is known in the literature as "*compressed sensing*" or "*compressive sampling*" (CS). Compressed sensing has a wide range of applications and it has been studied in the literature under many different names such as subset selection [3], sparse coding [4], sparse component analysis (SCA) with application to blind source separation of more sources than sensors [5], [6]. Other applications are biomagnetic inverse problems [7], [8], dictionary learning [9], [10], and image restoration [1], [11]. See [12] and the references therein for a list of more applications. In all these applications, the underlying linear inverse problem is the same and can be stated as follows: Represent a signal of interest using the minimum number of "*atoms*" (vectors) from an overcomplete dictionary.

Restricting the solution vector to be sparse converts the underlying problem from being impossible into being a tractable, but nevertheless still difficult, problem. There are many algorithms that have been developed in the literature for solving the under–determined linear system of equations under the constraint that the candidate solution vector is sparse. Examples include greedy algorithms [13], [14], the basis pursuit [15], [16], iterative-thresholding algorithms [17], and iterative reweighted norm algorithms [1], [7], [12], [18]–[20]. A brief description of some of these approaches is presented in Section II.

In this paper a novel methodology is developed to minimize a class of non–convex (concave on the non–negative orthant) functions for solving the afore mentioned problem. The proposed technique is based on locally replacing the original objective function by a quadratic convex function which is easily minimized. The resulting algorithm is iterative, and it will be shown that the penalty imposed by the convex objective function at any iteration depends on the gradient of the original objective function evaluated at the previous solution vector. Accordingly, if the objective function is carefully chosen, the penalty imposed by the convex function on the small entries of the solution vector can increase as the algorithm gets closer to a local minimum of the original objective function. As will be shown in this paper, for a certain selection of the convex

objective function, the class of algorithms called Iterative Re–weighted Least Squares (IRLS) [1], [7], [18]–[20] can be derived from the proposed methodology. Thus the proposed algorithms are a generalization and unification of the previous methods.

In this paper we propose a straightforward technique for selecting a convex function such that, for any starting solution vector $\boldsymbol{s}_0$, the algorithm generates a sequence $\{\boldsymbol{s}_k\}_{k=1}^{\infty}$ that converges to a fixed point of the original objective function. Since the original objective functions are non-convex, the proposed algorithm is susceptible to convergence to a local minimum. To alleviate this difficulty, we propose a random perturbation technique that enhances the performance of the proposed algorithm. The proposed algorithm is called MCCR, as an abbreviation of **M**inimizing a **C**oncave function via a **C**onvex function **R**eplacement.

Our contributions in this paper can be summarized as follows. (1) we propose a technique that provides a deeper understanding of the IRLS class of algorithms and provides a natural mechanism for deriving IRLS algorithms starting from suitably chosen *diversity* (antisparsity) measures, where the diversity of a vector is defined as the number of its nonzero entries, (2) we propose a new class of algorithms with better convergence properties compared to the regular IRLS algorithms and, hence, can be considered as enhancements to these algorithms, and (3) we suggest some novel techniques for improving the performance of IRLS methods.

The paper is organized as follows. A revision of some of the previous approaches is presented in Section II. The proposed convex objective function is presented in Section III. The MCCR algorithm is derived in Section IV. Some techniques that enhance the performance of the derived algorithms are presented in Section V. Section VI presents some computer simulations for assessing the performance of the proposed algorithms. Finally, conclusion remarks are given in Section VII.

II. Review of Previous Approaches

The problem of finding a *sparse* solution to an under–determined linear system of equations can be reformulated as one of basis selection from an over–complete dictionary matrix. Let $\boldsymbol{A}$ denote an $(m \times n)$ dictionary matrix whose columns comprise an over–complete set of basis vectors, i.e. $m < n$, where it is assumed that rank $(\boldsymbol{A}) = m$. It is required to represent a vector $\boldsymbol{x} \in \mathbb{R}^m$ by the *smallest* possible number of columns of the dictionary matrix, that is

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}^*, \tag{1}$$

where $\boldsymbol{s}^* \in \mathbb{R}^n$ is an *l–sparse* vector, i.e., a vector with $l \ll n$ nonzero entries. It is also assumed that $l < m$.

The goal of a sparse–signal recovery algorithm is to obtain an estimate of $\boldsymbol{s}^*$ given only $\boldsymbol{x}$ and $\boldsymbol{A}$. This problem is non-trivial since $\boldsymbol{A}$ is overcomplete. Since the original vector $\boldsymbol{s}^*$ is sparse, the problem of finding the desired solution vector can be phrased as an optimization problem where the objective is to maximize (minimize) an appropriate measure of sparsity (diversity) of the solution vector while simultaneously satisfying the constraints defined by (1). This can be expressed mathematically as

$$\hat{\boldsymbol{s}} = \arg\min_{\boldsymbol{s}} g(\boldsymbol{s}) \quad \text{subject to} \quad \boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}, \tag{2}$$

where $g(\cdot)$ is an objective function to be minimized that encourages sparsity in the solution vector. In this section we consider this function to be of the form

$$g_p(\boldsymbol{s}) = ||\boldsymbol{s}||_p^p = \sum_i |s[i]|^p, \tag{3}$$

where $p \geq 0$, and $s[i]$ is the i-th element of $\boldsymbol{s}$. Eq. (3) expresses the p-th norm of $\mathbf{s}$ (although it is not strictly a valid norm for $0 \leq p < 1$). We now briefly discuss issues relating to solving (3) for various values of $0 \leq p \leq 1$.

The ℓ_0–norm counts the number of non-zero entries in $\boldsymbol{s}$. Therefore, the ℓ_0–norm of an *l-sparse* solution vector $\boldsymbol{s}^*$ is expressed as $||\boldsymbol{s}^*||_{\ell_0} = l$. An *l-sparse* solution vector can be obtained by searching over the $\binom{n}{l}$ possible ways in which the basis sets can be chosen to find the best solution. Unfortunately, the cost of such combinatorial searches is prohibitive, requiring investigation of other approaches.

For $p = 1$, (2) is usually called basis pursuit (BP) [15]. Since $p = 1$ is the smallest value of p for which $g_p(\boldsymbol{s})$ is convex, ℓ_1-minimization has been utilized in the context of sparse solutions for many years. See [12], [20] and the references therein for the history of ℓ_1-minimization and its applications.

It was shown in [24]–[26] that minimizing ℓ_q–norm, for values of $0 < q < 1$ performs better than minimizing the ℓ_1–norm in the sense that a smaller number of measurements are needed for exact reconstruction of the sparse solution vector.

Another class of algorithms that can be utilized for finding a sparse solution vector is called Iterative Re-weighted Least Squares (IRLS) [7], [18]–[20], [27], [28]. IRLS algorithms combine the simplicity of the least squares solution while imposing sparsity on the solution vector. IRLS has been successfully utilized for minimizing a wide class of objective functions, e.g., the ℓ_p-norm with $0 < p < 2$, the logarithmic function of the form $\sum_i \log(|s[i]|)$ [7], and the Shannon entropy [18]. See [7], [20], [29], [30] for the history of IRLS and their convergence properties. IRLS algorithms have the form

$$(P_{w\ell_2}) \quad \min_{\boldsymbol{s}} ||\boldsymbol{W}^{-1}\boldsymbol{s}||_2^2 \quad \text{subject to} \quad \boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}, \tag{4}$$

where $\boldsymbol{W}$ is a diagonal weighting matrix that reflects our prior knowledge about the solution vector $\boldsymbol{s}$. The resulting algorithm is iterative, and the estimated solution at the kth iteration can be expressed as

$$\boldsymbol{s}_k = \boldsymbol{W}_k(\boldsymbol{A}\boldsymbol{W}_k)^{\dagger}\boldsymbol{x}, \tag{5}$$

where † is the Moore-Penrose inverse [31]. The difference between different IRLS algorithms resides in the way that the diagonal matrix is defined. In [7] $\boldsymbol{W}_k$ was selected as $\boldsymbol{W}_k = diag(|\boldsymbol{s}_{k-1}|)$ which was shown in [18] to be equivalent to minimizing $g(\boldsymbol{s}) = \sum_i \log(|s[i]|)$, while it was selected as $\boldsymbol{W}_k = diag(|s_{k-1}[i]|^{1-0.5q})$ in [18] for minimizing ℓ_q–norm[1] by following the affine scaling methodology. The local rate

[1]unless explicitly stated, $0 < q < 1$.

of convergence varies according to the expression of $\boldsymbol{W}$; for instance it was shown to be quadratic in [7], while in [20] it was either linear or super–linear for the algorithm approximating ℓ_1–norm or ℓ_q–norm, respectively.

Another approach for weighted norm minimization is the one proposed in [12], where $g(\boldsymbol{s})$ in (2) is replaced by $g_{w\ell_1}(\boldsymbol{s}) = ||\boldsymbol{W}_k^{-1}\boldsymbol{s}||_{\ell_1}$, where $\boldsymbol{W}_k = diag(|\boldsymbol{s}_{k-1}|)$ is also a diagonal weighting matrix. It was shown in [12] that this algorithm performs much better than the ℓ_1-norm minimization and converges in a few iterations. However each iteration is computationally expensive compared with an IRLS iteration.

III. Proposed Convex Function

In the absence of noise, a sparse solution vector of the under–determined system of equations, $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{x}$, can be obtained by solving the following optimization problem

$$\hat{\boldsymbol{s}} = \arg\min_{\boldsymbol{s}} g(\boldsymbol{s}) \quad \text{subject to} \quad \boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}, \tag{6}$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $m < n$, and $g(\cdot)$ is an objective function to be minimized that encourages sparsity in the solution. In this paper we restrict our attention to the class of non–convex objective functions that have the following two properties [9]:

(P1) $g(\boldsymbol{s})$ is separable, i.e., $g(\boldsymbol{s}) = \sum_i g_c(s[i])$, where $g_c(\cdot) : \mathbb{R} \to \mathbb{R}$ is a scalar function, and $s[i]$ is the ith entry of $\boldsymbol{s}$. Therefore, $g(\boldsymbol{s})$ is also permutation invariant, i.e., $g(\boldsymbol{s}) = g(\boldsymbol{P}\boldsymbol{s})$ for any permutation matrix $\boldsymbol{P}$.

(P2) The function $g_c(s) : \mathbb{R} \to \mathbb{R}$ referred to in P1 is sign invariant and concave–and–monotonically increasing on the nonnegative orthant $\mathcal{O}_1$.

Examples of $g_c(s)$ that were extensively used in the literature are $g_q(s) = |s|^q$ and $g_{log}(s) = \log(|s|)$ [18], [19], [24], [25]. More objective functions that obey property P2 are suggested in [32].

Since $g(\boldsymbol{s})$ is sign invariant and concave on $\mathcal{O}_1$, it is concave on each of the other orthants $\mathcal{O}_k, 1 \le k \le 2^n$. However, this does not imply that $g(\boldsymbol{s})$ is concave on $\mathbb{R}^n$.

Consider the following theorem.

Theorem 1 ([9] Theorem 8): Let $g(\boldsymbol{s}) : \mathbb{R}^n \to \mathbb{R}$ be permutation invariant, sign invariant, and concave on the nonnegative orthant $\mathcal{O}_1$. Then the global minimum of the optimization problem

$$\min_{\boldsymbol{s}} g(\boldsymbol{s}) \quad \text{subject to} \quad \boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}.$$

has at most m nonzero entries, where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $m < n$.

Proof: See [9]. □

Theorem 1 implies that if the purpose of solving (6) is to achieve a sparse solution vector, then the permutation invariant and sign invariant concave functions can be used as diversity measure functions.

The objective of this paper is to iteratively replace $g(\boldsymbol{s})$ by a convex smooth function $f(\boldsymbol{s})$, which can be easily minimized in an iterative fashion. The advantages of the proposed approach is that, by replacing the nonconvex function $g(\boldsymbol{s})$ by $f(\boldsymbol{s})$, each iteration becomes very tractable. However, since the original function $g(\boldsymbol{s})$ is nonconvex, the proposed algorithm may still converge to a local minimum.

For the proposed approach to be useful, the function $f(\boldsymbol{s})$ must be selected such that its global minimizer is guaranteed to reduce the original function $g(\boldsymbol{s})$. Towards that end, and to simplify the exposition, we consider replacing the one dimensional function $g_c(s)$ by a convex function $f(s)$ first. We then generalize to the n dimensional case. Consider the following theorem

Theorem 2: Given a differentiable one dimensional function $g_c(s) : \mathbb{R} \to \mathbb{R}$ that obeys Property P2, a one dimensional convex function $f(s) : \mathbb{R} \to \mathbb{R}$ with a global minimizer s^o, and an initial point s_0, then sufficient conditions for $g_c(s)$ to be reduced at s^o, i.e., $g_c(s^o) \le g_c(s_0)$, are: (1) $\text{sign}\,(f'(s_0)) = \text{sign}\,(g_c'(s_0))$, and (2) $|s^o| \le |s_0|$, where $g'(s_0)$ is the gradient of $g(s)$ at $s = s_0$.

Proof: Since the negative of the gradient of any objective function provides a valid direction for minimizing this objective function, the first condition in Theorem 2 insures that the minimizing direction of $f(s)$ is a valid minimizing direction for the original function $g(s)$. On the other hand, from P2, it is known that $g(s)$ is sign invariant and concave–and–monotonically increasing on the nonnegative orthant. Accordingly, the second condition $|s^o| \le |s_0|$ in Theorem 2 insures that that $g(s^o) \le g(s_0)$. □

Remark 1: For the special case where we select $f(s)$ such that $f(s) \ge g_c(s)$ and $f(s_0) = g_c(s_0)$, the convex objective function $f(\boldsymbol{s})$ *majorizes* $g(\boldsymbol{s})$ and the proposed technique is reduced to the Majorization–Minimization (MM) method [33]. However, in this paper we restrict $f(s)$ to satisfy the two conditions of Theorem 2 rather than the two conditions of the MM method. As will be shown in Section IV, the convex objective function proposed in Section IV-A satisfies the two conditions of the MM method, while the function proposed in Section IV-B violates these conditions.

There are many ways of selecting convex objective functions that satisfy the two conditions of Theorem 2. We propose a straightforward technique for doing this task. The proposed function has the following form

$$f(s) = g_c(s_0) + g_c'(s_0)(s - s_0) + \beta_0(s - s_0)^2, \tag{7}$$

where $g_c'(s_0)$ is the gradient of $g_c(s)$ at s_0, and $\beta_0 \ge 0$. Clearly the first condition of Theorem 2 is always satisfied, since $f'(s_0) = g_c'(s_0)$, and the second condition can be met by properly selecting the value of β_0.

Remark 2: If β_0 in (7) is selected as $\beta_0 = -0.5g_c''(s_0)$, where $g_c''(s_0)$ is the Hessian of $g_c(s)$ at $s = s_0$, then the function $f(s)$ in (7) is equivalent to the second–order Taylor series expansion of $g_c(s)$, with the sign of the quadratic term reversed. This results in $f(s)$ being a convex function. However it does not guarantee that the second condition of Theorem 2 is satisfied.

Assuming that the value of β_0 was chosen such that $f(s)$, defined in (7), satisfies the second condition of Theorem 2, then the point $s_1 = \arg\min_s f(s)$ is the global minimizer to $f(s)$ but not to $g(s)$. Accordingly, to reach the global minima of $g(s)$, the procedure is iterated, i.e., set $s_0 \leftarrow s_1$ and select β_0 such that (7) satisfies the conditions of Theorem 2. This

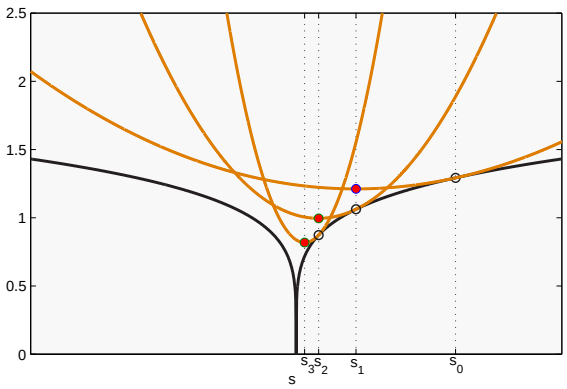


Fig. 1. The function $f(s)$ (orange curves) in successive iterations of the proposed algorithm for minimizing the nonconvex function $g_c(s)$ (black curve). The hollow circles represent the starting points, while the solid ones represent the global minima of the locally–convex functions.

procedure is repeated until $g(s)$ is minimized. The function $f(s)$ in successive iterations is shown in Fig. 1.

Since $g(\boldsymbol{s})$ is separable, the one dimensional function (7) can be readily extended to the n dimensional case by constructing the following separable convex function

$$\begin{aligned} f(\boldsymbol{s}) &= \sum_{i=1}^{n} f(s[i]) \\ &= \sum_{i=1}^{n} g_c(s_0[i]) + g_c'(s_0[i])(s[i]-s_0[i]) + \beta_0[i](s[i]-s_0[i])^2 \\ &= g(\boldsymbol{s}_0) + (\boldsymbol{s}-\boldsymbol{s}_0)^T\boldsymbol{d}_0 + (\boldsymbol{s}-\boldsymbol{s}_0)^T\boldsymbol{B}_0(\boldsymbol{s}-\boldsymbol{s}_0), \end{aligned} \tag{8}$$

where $\boldsymbol{d}_0 = \nabla g(\boldsymbol{s}_0)$ is the gradient of $g(\boldsymbol{s})$ at $\boldsymbol{s}=\boldsymbol{s}_0$, and $\boldsymbol{B}_0$ is a diagonal positive semidefinite matrix. Instead of using (8) in deriving the proposed algorithms, the following simplified function will be used

$$\tilde{f}(\boldsymbol{s}) = \boldsymbol{s}^T(\boldsymbol{d}_0 - 2\boldsymbol{B}_0\boldsymbol{s}_0) + \boldsymbol{s}^T\boldsymbol{B}_0\boldsymbol{s}, \tag{9}$$

which is equivalent to $f(\boldsymbol{s})$ after dropping the constant terms. Although the constant term in (8) does not affect the value of the solution vector of $f(\boldsymbol{s})$, it is important in proving the convergence of the derived algorithms.

IV. The MCCR Algorithms

In this section we derive algorithms for finding a sparse solution vector of the noise–free linear underdetermined system of equations, based on replacing the nonconvex objective function $g(\boldsymbol{s})$ in (6) by the convex objective function $f(\boldsymbol{s})$ defined in (8), or equivalently $\tilde{f}(\boldsymbol{s})$ defined in (9). Since $f(\boldsymbol{s})$ satisfies the first condition of Theorem 2, the remaining part of this section considers the problem of selecting $\boldsymbol{B}_0$ such that, for a given starting point $\boldsymbol{s}_0$, the original objective function $g(\boldsymbol{s})$ satisfies $g(\boldsymbol{s}_1) \leq g(\boldsymbol{s}_0)$, where

$$\begin{aligned} \boldsymbol{s}_1 = \ & \arg\min_{\boldsymbol{s}} \quad \boldsymbol{s}^T(\boldsymbol{d}_0 - 2\boldsymbol{B}_0\boldsymbol{s}_0) + \boldsymbol{s}^T\boldsymbol{B}_0\boldsymbol{s} \\ & \text{subject to} \quad \boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}. \end{aligned} \tag{10}$$

The optimization problem (10) can be solved by following the standard method of Lagrangian multipliers, which results in the following general expression for $\boldsymbol{s}_1$ [25]

$$\boldsymbol{s}_1 = \boldsymbol{s}_0 - 0.5\boldsymbol{B}_0^{-1}\boldsymbol{d}_0 + 0.5\boldsymbol{B}_0^{-1}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{B}_0^{-1}\boldsymbol{A}^T)^{-1}\boldsymbol{A}\boldsymbol{B}_0^{-1}\boldsymbol{d}_0. \tag{11}$$

Eq. (11) represents the general expression of the proposed MCCR algorithm. In the next subsections we propose two different choices for $\boldsymbol{B}_0$ that guarantee $g(\boldsymbol{s})$ is reduced at $\boldsymbol{s}_1$, i.e., $g(\boldsymbol{s}_1) \leq g(\boldsymbol{s}_0)$.

A. First approach: Select $\boldsymbol{B}_0$ such that $\tilde{f}(\boldsymbol{s}) = ||\boldsymbol{W}^{-1}\boldsymbol{s}||_2^2$

In this subsection we select $\boldsymbol{B}_0$ such that the global minima of $\tilde{f}(\boldsymbol{s})$, and hence $f(\boldsymbol{s})$, occurs at $\boldsymbol{s}^o = 0$.[2] From (8), or equivalently (9), the global minima of $f(\boldsymbol{s})$ occurs at $\boldsymbol{s}^o$, which can be calculated from the following equation

$$\nabla_{\boldsymbol{s}} f(\boldsymbol{s})|_{\boldsymbol{s}^o} = \boldsymbol{d}_0 + 2\boldsymbol{B}_0(\boldsymbol{s}^o - \boldsymbol{s}_0) = 0,$$

which leads to

$$\boldsymbol{s}^o = \boldsymbol{s}_0 - 0.5\boldsymbol{B}_0^{-1}\boldsymbol{d}_0. \tag{12}$$

Accordingly, setting $\boldsymbol{s}^o = 0$ is equivalent to

$$\boldsymbol{s}_0 = 0.5\boldsymbol{B}_0^{-1}\boldsymbol{d}_0. \tag{13}$$

Substituting (13) into (11), and utilizing the fact that $\boldsymbol{s}_0$ is feasible, i.e., $\boldsymbol{A}\boldsymbol{s}_0 = \boldsymbol{x}$, the expression of $\boldsymbol{s}_1$ is reduced to

$$\boldsymbol{s}_1 = \boldsymbol{B}_0^{-1}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{B}_0^{-1}\boldsymbol{A}^T)^{-1}\boldsymbol{x}. \tag{14}$$

Note that $\boldsymbol{s}_1$ is the global minima of the convex problem (10) but not of the original problem (6). However, since $g(\boldsymbol{s}_1) \leq g(\boldsymbol{s}_0)$ from Theorem 3, a stable point of (6) can be obtained by iteratively repeating (14), i.e., for $k \geq 0$

$$\boldsymbol{s}_{k+1} = \boldsymbol{B}_k^{-1}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{B}_k^{-1}\boldsymbol{A}^T)^{-1}\boldsymbol{x}, \tag{15}$$

where $\boldsymbol{B}_k$ is a diagonal matrix which depends on $\boldsymbol{s}_k$ and $\boldsymbol{d}_k = \nabla g(\boldsymbol{s}_k)$, the gradient of $g(\boldsymbol{s})$ at $\boldsymbol{s}_k$. The diagonal entries of $\boldsymbol{B}_k$ can be easily calculated from (13) as follows

$$B_k[i,i] = \frac{d_k[i]}{2s_k[i]}, \quad i = 1,\ldots,n. \tag{16}$$

Note that $B_k[i,i]$ calculated from (16) is nonnegative, i.e., $\boldsymbol{B}_k$ is positive semidefinite, which is a necessary condition for $f(\boldsymbol{s})$ to be convex. This follows readily from the sign–invariant property of $g(\boldsymbol{s})$ and its monotonically increasing property on the nonnegative orthant $\mathcal{O}_1$. Note that, substituting (13) into (9), the expression of $\tilde{f}(\boldsymbol{s})$ is reduced to $\tilde{f}(\boldsymbol{s}) = ||\boldsymbol{W}^{-1}\boldsymbol{s}||_2^2$ as desired, where $\boldsymbol{W} = \boldsymbol{B}_0^{-1/2}$.

Since the original objective function $g(\boldsymbol{s})$ is nonconvex, the convergence of the proposed algorithm (15) into the "global" minima of the original problem (6) is not guaranteed. However, the proposed algorithm always converges to a fixed point of (6) and does not have undesired properties such as divergence or oscillation. The proof of convergence of (15) to a fixed point of (6) is shown by the following theorem.

[2]It should be noted that, even though the global minimizer of $\tilde{f}(\boldsymbol{s})$ is $\boldsymbol{s}^o = \boldsymbol{0}$, the solutions to the constrained problem (10) is given by (11) and is generally not $\boldsymbol{s}^o = \boldsymbol{0}$.

Theorem 3: Given a starting solution vector $\boldsymbol{s}_k \in \mathbb{R}^n$ and a separable function $g(\boldsymbol{s}) = \sum_i g_c(s[i]) : \mathbb{R}^n \rightarrow \mathbb{R}$, where $g_c(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$ is differentiable and obeys P2. Let $\boldsymbol{d}_k$ denote the gradient of $g(\boldsymbol{s})$ at $\boldsymbol{s}_k$. If $\boldsymbol{B}_k$ in (8) is calculated using (16) and a new solution vector $\boldsymbol{s}_{k+1}$ is obtained using (15), then $g(\boldsymbol{s}_{k+1}) \leq g(\boldsymbol{s}_k)$ with the equality holds only if $\boldsymbol{s}_k$ is a fixed point of the original problem (6).

Proof: Since the global minima of $f(\boldsymbol{s})$ occurs at $\boldsymbol{s}^o = 0$, it is straightforward to show that $f(\boldsymbol{s})$ is sign invariant, i.e., $f(\boldsymbol{s}) = f(|\boldsymbol{s}|)$. Since both $g(\boldsymbol{s})$ and $f(\boldsymbol{s})$ are sign invariant, it will be assumed without loss of generality that $\boldsymbol{s}_k \in \mathcal{O}_1$, the nonnegative orthant. From (8) it is clear that both $f(\boldsymbol{s})$ and $g(\boldsymbol{s})$ share a common tangent at $\boldsymbol{s}_k$, which is given by the first two terms of (8). From the convexity of $f(\boldsymbol{s})$ and the concavity of $g(\boldsymbol{s})$ on $\mathcal{O}_1$, a common tangent to $f(\boldsymbol{s})$ and $g(\boldsymbol{s})$ on $\mathcal{O}_1$ implies that $g(\boldsymbol{s})$ is upper bounded by $f(\boldsymbol{s})$ on $\mathcal{O}_1$. This is also applicable to the other $2^n - 1$ orthants. Therefore, $g(\boldsymbol{s}) \leq f(\boldsymbol{s})$, where the equality holds only at the 2^n tangent points, i.e., for a given starting point $\boldsymbol{s}_k \in \mathcal{O}_1$ we have $g(\boldsymbol{s}_k) = f(\boldsymbol{s}_k)$. However, from (10), and since $f(\boldsymbol{s})$ is a quadratic convex function, we have $f(\boldsymbol{s}_{k+1}) \leq f(\boldsymbol{s}_k)$ where the equality holds if and only if $\boldsymbol{s}_{k+1} = \boldsymbol{s}_k$, i.e., $\boldsymbol{s}_{k+1}$ can not be one of the other $2^n - 1$ tangent points. In conclusion, if $\boldsymbol{s}_{k+1}$ is different from $\boldsymbol{s}_k$, then $g(\boldsymbol{s}_{k+1}) < f(\boldsymbol{s}_{k+1}) < f(\boldsymbol{s}_k) = g(\boldsymbol{s}_k)$, while $g(\boldsymbol{s}_{k+1}) = f(\boldsymbol{s}_{k+1}) = f(\boldsymbol{s}_k) = g(\boldsymbol{s}_k)$ if and only if $\boldsymbol{s}_{k+1} = \boldsymbol{s}_k$. In this last case, $\boldsymbol{s}_k$ is a fixed point of (6). □

Thus, starting from any feasible solution vector $\boldsymbol{s}_0 \in \mathbb{R}^n$, the algorithm (15) generates a sequence $\{\boldsymbol{s}_k\}_{k=1}^{\infty}$ that converges at least to a local minimum of the original problem (6). At this point, $\boldsymbol{s}$ does not vary from one iteration to the next.

Theorem 3 shows that any algorithm of the form (15) with $\boldsymbol{B}_k$ calculated using (16) is guaranteed to converge to a fixed point of the original optimization problem (6) as long as $g(\boldsymbol{s}) \in \mathcal{G}$, where $\mathcal{G}$ is the set of all objective functions that obey P1–P2. The exact expression of the objective function $g(\boldsymbol{s})$ affects only the rate of convergence of (15). The rate of convergence of some functions $g(\boldsymbol{s}) \in \mathcal{G}$ are derived in [32].

Relation with IRLS algorithms*:* Define $\boldsymbol{W}_k = \boldsymbol{B}_k^{-\frac{1}{2}}$, then (15) can be written as

$$\boldsymbol{s}_{k+1} = \boldsymbol{W}_k (\boldsymbol{A}\boldsymbol{W}_k)^{\dagger} \boldsymbol{x}, \tag{17}$$

where † is the Moore-Penrose inverse [31]. Comparing (17) with (5) we readily find that the derived algorithm has the form of the IRLS algorithms. The convergence of an IRLS algorithm to a sparse solution vector depends on the relation between $\boldsymbol{W}_k$ and the previous solution vector $\boldsymbol{s}_k$. Some IRLS algorithms may not converge to a sparse solution vector, e.g., the algorithm derived in [18] for minimizing the Shannon entropy. However, the proposed technique provides a general methodology for deriving IRLS algorithms that converge to sparse solution vectors and enjoy fixed point convergence.

The performance of the proposed algorithm in finding a sparse solution is shown in Fig. 2. The difference between Fig. 2 and Fig. 1 is that the global minimum of each convex function in Fig. 2 is at the origin. Note that this is the global minimum to the unconstrained minimization of $f(\boldsymbol{s})$ but not to the constrained problem (10), which has a global minima in this case given by (15). Fig. 2 shows the successive iterations associated with reducing a zero entry of the solution vector $\boldsymbol{s}$ towards its optimum value of zero. As shown in this figure, the algorithm takes large steps when the value of $s[j]$ is large, while the steps get smaller and smaller as $s_k[j]$ approaches zero. This performance is due to the weighting matrix $\boldsymbol{B}_k$ whose ith diagonal element is given by (16). From (16) we can see that as $s_k[i] \rightarrow 0$ the weights become large due to the fact that the function $g_c(s)$ is concave and monotonically increasing on $\mathcal{O}_1$, and hence its gradient increases as the value of s decreases. Large weights associated with small values of $\boldsymbol{s}$ are required for a sparse solution. Note that the quadratic replacement function $f(\boldsymbol{s})$ on the other hand imposes small penalties on small values of $\boldsymbol{s}$, a fact which explains why the MCCR approach can not achieve a sparse solution in one step.

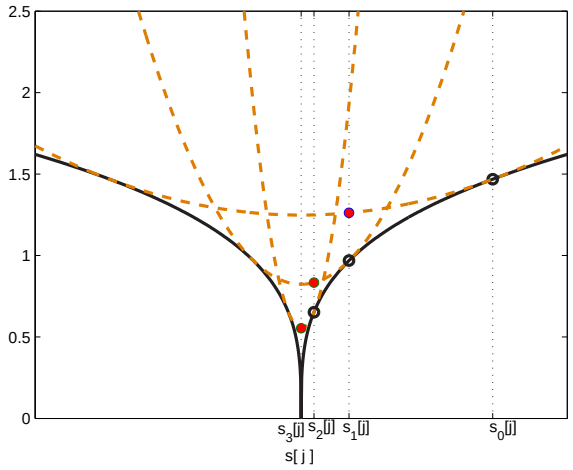


Fig. 2. Successive iterations of the proposed algorithm for minimizing the nonconvex function $g_c(s[j])$ (black curve). The hollow circles represent the starting points, while the solid ones represent the global minima of the locally–convex functions.

B. Second Approach: Select $\boldsymbol{B}_0$ such that $\tilde{f}(\boldsymbol{s}) = ||\boldsymbol{W}^{-1}(\boldsymbol{s} - \theta \boldsymbol{s}_0)||_2^2$

In Section IV-A we proved that selecting $\boldsymbol{B}_k$ as in (16) reduces (8) to $f(\boldsymbol{s}) = ||\boldsymbol{W}_k^{-1}\boldsymbol{s}||_2^2$, where $\boldsymbol{W}_k = \boldsymbol{B}_k^{-\frac{1}{2}}$. Consequently, the resulting algorithm (15), or equivalently (17), has the general form of the IRLS algorithms. Moreover, it was shown by Theorem 3 that, starting from any bounded solution vector, the proposed algorithm is guaranteed to converge to a stable solution vector of the original problem (6). However, the rate of convergence of the proposed algorithm is limited by the nature of the weighted ℓ_2–ball, as shown in Fig. 3. In contrast to the ℓ_1–ball, the ℓ_2–ball, and hence ℓ_{w2}–ball, the weighted ℓ_2–ball, have rounded tips. As a result, the proposed algorithm, and all other IRLS algorithms, may require large number of iterations to converge to a solution vector, especially when m and n are large.

In this figure, the exact and the estimated solution vectors are represented by the blue circle at $\boldsymbol{s}^* = [0 \quad 1 \quad 0]^T$ and the green circle, respectively, while $\mathcal{H}$, the set of all points $\boldsymbol{s} \in \mathbb{R}^3$ obeying $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{A}\boldsymbol{s}^*$, is represented by the red line passing

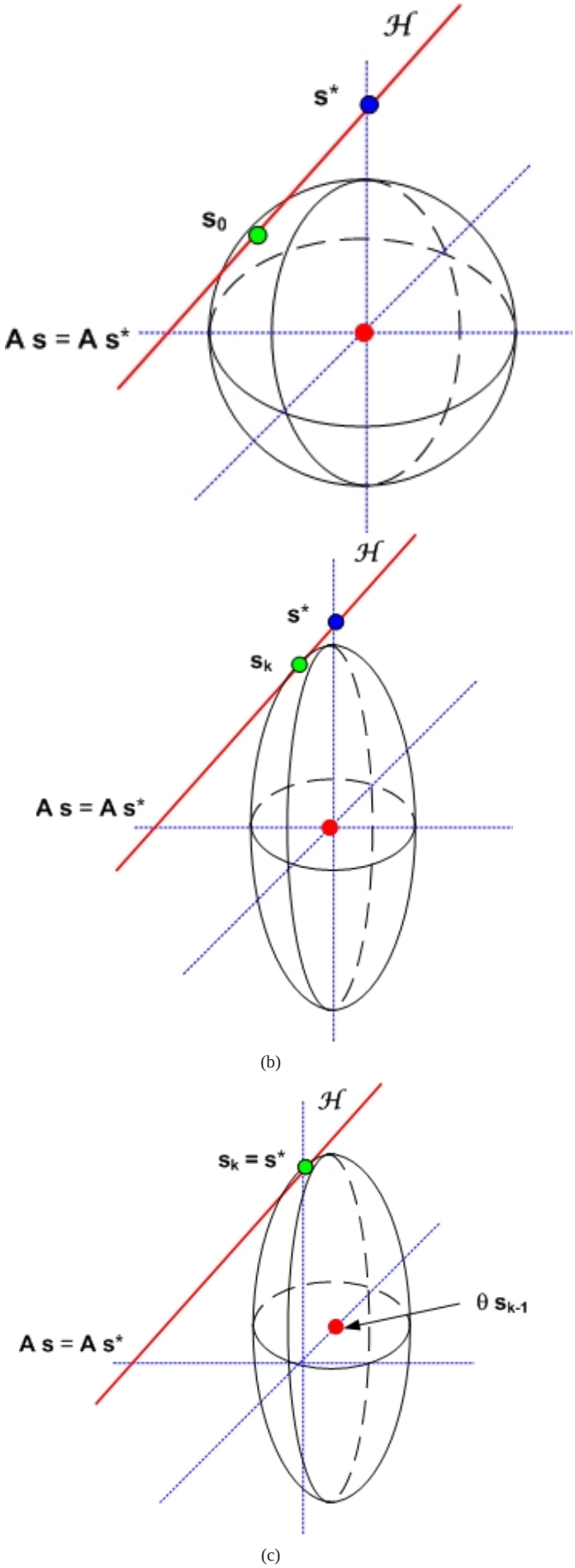


Fig. 3. Geometric interpretation of the solution vector obtained by: (a) $||\boldsymbol{s}||_2^2$ minimization, (b) $||\boldsymbol{W}_{k-1}^{-1}\boldsymbol{s}||_2^2$ minimization, and (c) $||\boldsymbol{W}_{k-1}^{-1}(\boldsymbol{s}-\theta\boldsymbol{s}_{k-1})||_2^2$ minimization .

through $\boldsymbol{s}^*$. The ℓ_2 minimizer of (6) is the point on $\mathcal{H}$ closest to the origin. This point can be found by blowing up the ℓ_2–ball, represented by the hypersphere in Fig. 3(a), until it contacts $\mathcal{H}$. Due to the randomness of the entries of $\boldsymbol{A}$, $\mathcal{H}$ is oriented at random angle. Accordingly, with high probability, the closest point $\boldsymbol{s}_0$ will live away from the coordinate axes and hence will be neither sparse nor close to the correct solution vector $\boldsymbol{s}^*$ [34].

Geometrically, incorporating a diagonal weighting matrix into the ℓ_2-norm causes the ℓ_2–ball to elongate along certain direction and no longer be symmetric. Assuming that the proposed algorithm (15) is converging to the exact solution vector $\boldsymbol{s}^*$, then at the kth iteration the ℓ_{w2}–ball will contact $\mathcal{H}$ at $\boldsymbol{s}_k$, which is closer to the exact solution vector $\boldsymbol{s}^*$ as shown in Fig. 3(b). At the next iteration, and due to the new weighting matrix, the ℓ_{w2}–ball will be further squeezed and contact $\mathcal{H}$ at a point $\boldsymbol{s}_{k+1}$ closer to $\boldsymbol{s}^*$ than the previous solution point $\boldsymbol{s}_k$. The procedure continues until the ℓ_{w2}–ball contacts $\mathcal{H}$ at $\boldsymbol{s}^*$.

Since the sparse solution vector $\boldsymbol{s}^*$ is restricted to live at the tip of the ℓ_{w2}–ball, algorithm (15) may take a large number of iterations to converge to that vector, i.e., for the ℓ_{w2}–ball to contact $\mathcal{H}$ at $\boldsymbol{s}^*$. However, the number of iterations can be significantly reduced if the sparse solution vector $\boldsymbol{s}^*$ is allowed to live on the surface of the ℓ_{w2}–ball. This can be achieved by changing the origin of the ℓ_{w2}–ball in each iteration of (15), where the origin at the kth iteration depends on the previous solution vector. The proposed modification is shown in Fig. 3(c), where the origin of the ℓ_{w2}–ball is shifted to $\theta\boldsymbol{s}_{k-1}$, where θ is a parameter to be determined. As shown in this figure, although $\mathcal{H}$ contacts the ℓ_{w2}–ball at a point away from its tip, θ can be chosen so that this point is closer to the exact solution vector $\boldsymbol{s}^*$ than the solution given by the IRLS solution.

Based on this geometric interpretation, we propose selecting $\boldsymbol{B}_0$ in (9) such that $\tilde{f}(\boldsymbol{s}) = ||\boldsymbol{W}^{-1}(\boldsymbol{s}-\theta\boldsymbol{s}_0)||_2^2$, where $\boldsymbol{W}$ is a diagonal weighting matrix and $\boldsymbol{s}_0$ is a given solution vector. Since $\boldsymbol{W}$ is diagonal, $\tilde{f}(\boldsymbol{s})$ can be written as

$$\tilde{f}(\boldsymbol{s}) = ||\boldsymbol{W}^{-1}(\boldsymbol{s}-\theta\boldsymbol{s}_0)||_2^2 = -2\theta\boldsymbol{s}^T\boldsymbol{W}^{-2}\boldsymbol{s}_0 + \boldsymbol{s}^T\boldsymbol{W}^{-2}\boldsymbol{s} + C, \tag{18}$$

where C is a constant term that does not depend on $\boldsymbol{s}$. Equating (9) and (18) we get

$$\begin{aligned} \boldsymbol{W} &= \boldsymbol{B}_0^{-\frac{1}{2}}, \\ B_0[i,i] &= \frac{d_0[i]}{2(1-\theta)s_0[i]}, \quad i = 1,\ldots,n, \end{aligned} \tag{19}$$

where the second equation results from the comparison of the first terms in (9) and (18), and utilizing the fact that $\boldsymbol{B}_0$ is diagonal.

As described in Section IV-A, the quantity $d_0[i]/s_0[i]$ is always nonnegative. Accordingly, for $\boldsymbol{B}_0$ in (19) to be non-negative definite, the value of θ must be less than 1. As will be shown later, this is also a necessary condition for (18) to produce a sequence of solution vectors that minimize the original objective function, i.e., $g(\boldsymbol{s}_{k+1}) < g(\boldsymbol{s}_k)$, for all $k \geq 0$.

Based on this formulation, the optimization problem (10) is reduced to

$$\boldsymbol{s}_1 = \arg\min_{\boldsymbol{s}} \quad ||\boldsymbol{W}_0^{-1}(\boldsymbol{s} - \theta \boldsymbol{s}_0)||_2^2 \quad \text{subject to} \quad \boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}, \tag{20}$$

where $\boldsymbol{W}_0$ is a diagonal weighting matrix calculated from (19).

Remark 3: It is worth mentioning that the subtraction strategy in (20) has somewhat the flavour of recent Bregman methods [35], [36] for improving convergence of such constrained problems. It is not difficult to show that the objective function (20) is equivalent to the Bregman distance (based on the objective function $f(\boldsymbol{s}) = ||\boldsymbol{W}^{-1}\boldsymbol{s}||_{\ell_2}^2$) between $\boldsymbol{s}$ and $\theta\boldsymbol{s}_0$, i.e., $f_2(\boldsymbol{s}) = ||\boldsymbol{W}^{-1}(\boldsymbol{s} - \theta\boldsymbol{s}_0)||_{\ell_2}^2 = D_f^{\boldsymbol{p}}(\boldsymbol{s}, \theta\boldsymbol{s}_0)$, where $D_f^{\boldsymbol{p}}(\boldsymbol{s}, \theta\boldsymbol{s}_0)$ is the Bregman distance between $\boldsymbol{s}$ and $\theta\boldsymbol{s}_0$ based on the objective function $f(\boldsymbol{s})$, and $\boldsymbol{p} \in \partial f(\boldsymbol{s}_0)$. A topic for future research is to consider utilizing the Bregman distance based on the original objective function $g(\boldsymbol{s})$, rather than $f(\boldsymbol{s})$, to derive new algorithms for solving an under-determined liner system of equations.

Equation (20) is not in the standard form of a weighted minimum norm problem. Accordingly, for solving this problem, we follow the following steps. First, assuming that $\boldsymbol{W}_0$ is invertible, let $\boldsymbol{y}$ be defined as

$$\boldsymbol{y} \triangleq \boldsymbol{W}_0^{-1}(\boldsymbol{s} - \theta\boldsymbol{s}_0).$$

Then, $\boldsymbol{s}$ is expressed as

$$\boldsymbol{s} = \boldsymbol{W}_0\boldsymbol{y} + \theta\boldsymbol{s}_0. \tag{21}$$

Finally, substituting (21) into (20) we get

$$\tilde{\boldsymbol{y}} = \arg\min_{\boldsymbol{y}} \quad ||\boldsymbol{y}||_2^2 \quad \text{subject to} \quad \bar{\boldsymbol{x}} = \bar{\boldsymbol{A}}\boldsymbol{y}, \tag{22}$$

where $\bar{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{W}_0$ and $\bar{\boldsymbol{x}} = \boldsymbol{x} - \theta\boldsymbol{A}\boldsymbol{s}_0 = (1-\theta)\boldsymbol{x}$. Equation (22) is a minimum norm problem in the standard form and has the following solution

$$\tilde{\boldsymbol{y}} = (\bar{\boldsymbol{A}})^\dagger\bar{\boldsymbol{x}} = (1-\theta)(\boldsymbol{A}\boldsymbol{W}_0)^\dagger\boldsymbol{x}. \tag{23}$$

Substituting (23) into (21), the solution vector at the $(k+1)$th iteration is expressed as

$$\begin{aligned} \boldsymbol{s}_{k+1} &= \theta\boldsymbol{s}_k + (1-\theta)\boldsymbol{W}_k(\boldsymbol{A}\boldsymbol{W}_k)^\dagger\boldsymbol{x}. \\ &= \theta\boldsymbol{s}_k + (1-\theta)\boldsymbol{B}_k^{-1}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{B}_k^{-1}\boldsymbol{A}^T)^{-1}\boldsymbol{x}, \end{aligned} \tag{24}$$

where $\boldsymbol{B}_k$ is calculated using (19) and $\boldsymbol{s}_0$ and $\boldsymbol{d}_0$ have been replaced by $\boldsymbol{s}_k$ and $\boldsymbol{d}_k$, respectively, where $\boldsymbol{d}_k = \nabla g(\boldsymbol{s}_k)$. In the remaining part of this paper, the acronyms MCCR and IRLS will be used to refer to (24)and (15), respectively .

Selecting the value of θ: The performance of the MCCR algorithm (24) depends on the value of θ. From (24) we note that multiplying $\boldsymbol{B}_k$ by any scaling parameter does not affect the value of $\boldsymbol{s}_{k+1}$. Accordingly, the constant term in the expression of $B_k[i,i]$ in (19) can be dropped without affecting the value of $\boldsymbol{s}_{k+1}$. As a result, the dependency of $\boldsymbol{B}_k$ on the unknown parameter θ can be dropped by calculating $B_k[i,i]$, or equivalently $B_k^{-1}[i,i]$, using the following equation

$$B_k^{-1}[i,i] = \frac{s_k[i]}{d_k[i]}, \quad i = 1, \ldots, n. \tag{25}$$

TABLE I
SOME OBJECTIVE FUNCTIONS THAT CAN BE USED WITH MCCR.

Function	$d[i]$	$B^{-1}[i,i]$
$g(\boldsymbol{s}) = \sum_i \lvert s[i]\rvert^q,\ 0 < q < 1$	$qs[i]\lvert s[i]\rvert^{q-2}$	$\lvert s[i]\rvert^{2-q}$
$g(\boldsymbol{s}) = \sum_i \log(\lvert s[i]\rvert)$	$\frac{s[i]}{\lvert s[i]\rvert 2}$	$\lvert s[i]\rvert^2$
$g(\boldsymbol{s}) = \sum_i \log(1 + \lvert s[i]\rvert/\delta)$	$\frac{s[i]}{\lvert s[i]\rvert(\delta+\lvert s[i]\rvert)}$	$\lvert s[i]\rvert(\delta + \lvert s[i]\rvert)$
$g(\boldsymbol{s}) = \sum_i \operatorname{atan}(\lvert s[i]\rvert/\delta)$	$\frac{\delta s[i]}{\lvert s[i]\rvert(\delta^2+s^2[i])}$	$\lvert s[i]\rvert(\delta^2 + s^2[i])$
$g(\boldsymbol{s}) = \sum_i \frac{\lvert s[i]\rvert}{\lvert s[i]\rvert+\delta}$	$\frac{\delta s[i]}{\lvert s[i]\rvert(\delta+s[i])^2}$	$\lvert s[i]\rvert(\delta + s[i])^2$

The expression of $B^{-1}[i,i]$ derived from some objective functions of the form $g(\boldsymbol{s}) = \sum_i g_c(s[i])$, where $g_c(s[i])$ obeys P2, is presented in Table I. Note that all the constant terms are dropped from the expression of $B^{-1}[i,i]$ in the third column of Table I. The free parameter $\delta > 0$ in the last three objective functions of Table I can be selected to control convergence. Many approaches for selecting the value of δ are suggested in [32]. In this paper the value of δ at the kth iteration is calculated as $\delta_k = 0.5\ \text{mean}\,(|\boldsymbol{s}_{k-1}|)$.

After dropping the dependency of $B_k^{-1}[i,i]$ on the unknown parameter θ, Eq. (24) can be written as

$$\boldsymbol{s}_{k+1} = \theta\boldsymbol{s}_k + (1-\theta)\tilde{\boldsymbol{s}}_{k+1}, \tag{26}$$

where $\tilde{\boldsymbol{s}}_{k+1} = \boldsymbol{B}_k^{-1}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{B}_k^{-1}\boldsymbol{A}^T)^{-1}\boldsymbol{x}$ is the IRLS solution vector obtained using (15). Eq. (26) shows that the MCCR solution at the $(k+1)$th iteration is an *affine* combination of the solution vector at the kth iteration and the IRLS solution at the $(k+1)$th vector. Geometrically this means that the solution vector $\boldsymbol{s}_{k+1}$ exists at a point determined by the parameter θ, somewhere along the line connecting $\boldsymbol{s}_k$ and $\tilde{\boldsymbol{s}}_{k+1}$. Note that $\boldsymbol{s}_{k+1}$ calculated using (26) is always feasible, i.e., $\boldsymbol{A}\boldsymbol{s}_{k+1} = \boldsymbol{x}$. Accordingly, we will refer to the set of all affine combinations of $\boldsymbol{s}_k$ and $\tilde{\boldsymbol{s}}_{k+1}$ for various values of θ as the $feasible$–line.

In Section IV-A we proved that the IRLS algorithms derived in this paper are fixed point converging, i.e., for a given feasible solution vector $\boldsymbol{s}_k$, the IRLS solution vector $\tilde{\boldsymbol{s}}_{k+1}$ in (26) satisfies $g(\tilde{\boldsymbol{s}}_{k+1}) \leq g(\boldsymbol{s}_k)$. This implies that the direction from $\boldsymbol{s}_k$ to $\tilde{\boldsymbol{s}}_{k+1}$ on the $feasible$–line provides a descending direction for the original objective function $g(\boldsymbol{s})$. This in turn implies that θ must satisfy the condition ($\theta \leq 1$) in order that $g(\boldsymbol{s}_{k+1}) \leq g(\boldsymbol{s}_k)$. The optimum value of θ, denoted θ_o, is the one that minimizes $g(\boldsymbol{s})$ for all $\boldsymbol{s}$ belonging to the $feasible$–line. Accordingly, θ_o can be determined by solving the following optimization problem

$$\theta_o = \arg\min_{\theta} \quad g(\theta\boldsymbol{s}_k + (1-\theta)\tilde{\boldsymbol{s}}_{k+1}) \quad \text{subject to} \quad \theta_{min} < \theta < 1, \tag{27}$$

where θ_{min} is a lower bound on θ. This formulation is readily implemented using, e.g., the Golden-section search method, and can be solved using the Matlab function "*fminbnd.m*". In the simulation results to be shown we empirically set $\theta_{min} = -2$.

Note that if no point along the feasible line decreases $g(\boldsymbol{s})$, then the optimum value of θ is zero. In this case, the MCCR solution given by (24) coincides with the IRLS solution given by (15). Furthermore, if the IRLS solution is a fixed point, then the MCCR solution is also a fixed point.

A further advantage of the MCCR solution (24) over the IRLS solution (15) is that (24) can be easily adapted to solve the following problem

$$\hat{\boldsymbol{s}} = \arg\min_{\boldsymbol{s}} g(\boldsymbol{s}) \quad \text{subject to} \quad \boldsymbol{s} \geq 0, \quad \boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}. \tag{28}$$

This can be readily accomplished by selecting a starting feasible solution vector and properly selecting the value of $\theta_{min} \leq 1$ in (27) to ensure that $\theta \boldsymbol{s}_k + (1-\theta)\tilde{\boldsymbol{s}}_{k+1} \geq 0$ for all $\theta_{min} \leq \theta \leq 1$.

V. Performance Enhancement

In this section the performance of the derived algorithms is discussed. Since the MCCR solution (24) is an affine combination of the previous solution and the IRLS solution (15), we will consider only the factors that affect the performance of the IRLS solution (15). The discussion presented in this section is general and is applicable to all of the objective functions in Table I. There are two issues that can affect the performance of the updating equation (15). These issues, as well as their mitigating interventions, are now discussed.

A. Inversion Operation Associated with (15)

The first issue is the inversion operation associated with (15). Define $\boldsymbol{C}_k = \boldsymbol{A}\boldsymbol{B}_k^{-1}\boldsymbol{A}^T$. Then for the solution of (15) to exist, $\boldsymbol{C}_k$ must be invertible. Recall from (16) that [3]

$$B_k^{-1}[i,i] = \frac{s_k[i]}{d_k[i]}, \quad i = 1, \ldots, n. \tag{29}$$

Accordingly, $\boldsymbol{C}_k$ can be expressed as

$$\boldsymbol{C}_k = \sum_i^n \frac{s_k[i]}{d_k[i]} \boldsymbol{a}_i \boldsymbol{a}_i^T, \tag{30}$$

where $\boldsymbol{a}_i$ is the ith column of $\boldsymbol{A}$. Thus $\boldsymbol{C}_k$ is a summation of n rank-1 $(m \times m)$ matrices. Accordingly, for $\boldsymbol{C}_k$ to be invertible, at least m elements of $\boldsymbol{s}_k$ must be significantly different from zero, which contradicts the assumption that $\boldsymbol{s}$ is sparse. To overcome this difficulty, we redefine $B_k^{-1}[i,i]$ as follows

$$B_k^{-1}[i,i] = \begin{cases} \frac{s_k[i]}{d_k[i]} & \text{if} \quad |s_k[i]| \geq \epsilon \\ \frac{s_k[i]}{d_k[i]}\big|_{s_k[i] \leftarrow \epsilon}, & \text{otherwise,} \end{cases} \tag{31}$$

where ϵ is a small positive number. The second line in (31) means that, if the condition $|s_k[i]| \geq \epsilon$ is not satisfied, then $B_k^{-1}[i,i]$ is calculated by evaluating $s_k[i]/d_k[i]$ first then replacing each $s_k[i]$ by ϵ. For example, for $g_c(s) = |s|^q$ we have $d = qs|s|^{q-2}$, and if $|s| < \epsilon$ then the corresponding entry of $\boldsymbol{B}^{-1}$ will equal ϵ^{2-q}/q.

Unfortunately, it was shown in [32] that the performance of MCCR depends on the value of ϵ. To overcome this difficulty, we follow the procedure suggested in [19]. Here, ϵ is initiated to a relatively large value, e.g. $\epsilon = 1$. Its value is then reduced by a factor of 10 at iterations where the condition $||\boldsymbol{s}_k - \boldsymbol{s}_{k-1}||/||\boldsymbol{s}_k|| < \sqrt{\epsilon}/100$ is satisfied. The MCCR algorithm is summarized in Table II.

[3] In (29) the constant associated with $B_k^{-1}[i,i]$ is neglected.

TABLE II
The MCCR Algorithm

Algorithm 1: The MCCR Algorithm

Given an $(m \times n)$ matrix $\boldsymbol{A}$ of basis vectors, and a vector $\boldsymbol{x} \in \mathbb{R}^m$, select one of the value for ϵ, e.g. $\epsilon = 1$, an empirically–selected value for θ_{min}, e.g. $\theta_{min} = -2$, a small $\boldsymbol{s}_0$. This point can be selected as the least squares solution, i.e. $\boldsymbol{s}_0 = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1}\boldsymbol{x}$. T steps:

1) Repeat until convergence:
   - Calculate $\boldsymbol{B}_k^{-1}$ using (31).
   - Calculate $\tilde{\boldsymbol{s}}_{k+1} = \boldsymbol{B}_k^{-1}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{B}_k^{-1}\boldsymbol{A}^T)^{-1}\boldsymbol{x}$.
   - Calculate θ using (27).
   - Set $\boldsymbol{s}_{k+1} = \theta\boldsymbol{s}_k + (1-\theta)\tilde{\boldsymbol{s}}_{k+1}$.
   - if $||\boldsymbol{s}_{k+1} - \boldsymbol{s}_k||_2/||\boldsymbol{s}_{k+1}||_2 < \sqrt{\epsilon}/100$, set $\epsilon = \epsilon/10$ end
   - Set $k = k+1$,

End

2) Output $\boldsymbol{s}_{k+1}$ as the solution.

B. Non–convexity of $g(\boldsymbol{s})$

The second issue that affects the performance of (15), and hence (24), is the non–convexity of the objective functions considered in this paper. As a result, MCCR may converge to a local minima. This problem may be partially alleviated by the *perturbed* MCCR algorithm (PMCCR) described in Table III. At each iteration, the solution vector $\boldsymbol{s}_j^0$ is perturbed by a random noise vector $\boldsymbol{v}$, which is constrained to be in the null space of the mixing matrix; i.e. $\boldsymbol{v} = \boldsymbol{F}\boldsymbol{u}$ where $\boldsymbol{F}$ is any matrix whose range is the null space of $\boldsymbol{A}$ and $\boldsymbol{u} \in \mathbb{R}^{n-m}$ is a random noise vector. The columns of $\boldsymbol{F}$ can be chosen by first calculating the singular value decomposition (SVD) of $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, and then selecting the columns of $\boldsymbol{F}$ as the last $(n-m)$ columns of $\boldsymbol{V}$. In the simulation results, the elements of $\boldsymbol{u}$ are sampled from a zero mean uniformly distributed random variable between $\pm\alpha\bar{s}_{max}$, where $\bar{s}_{max}$ is the maximum absolute element in the MCCR solution vector $\bar{\boldsymbol{s}}$, and α is an empirically determined non–negative number in the range e.g., $1 \leq \alpha \leq 2$. Note that the perturbation noise is constrained to be in the null space of the mixing matrix to insure the feasibility of the new perturbed vector. A stopping criterion could be a maximum number of iterations, or a pre–specified value of the cardinality of the solution. Note that, in each step, the new solution vector is accepted only if its cardinality is less than the cardinality of the previous solution vector. Accordingly, by following this strategy, it is guaranteed that no performance degradation occurs.

VI. Simulation Results

In this section, a set of examples are presented in order to examine the effect of θ and the perturbation procedure on the performance of the MCCR algorithm, and to provide a comparison between the MCCR algorithm and other well known algorithms. In the first Example, we present a comparison between the proposed MCCR algorithms and the counterpart IRLS algorithms. This example reflects the effect of the parameter θ. A comparison between MCCR and PMCCR is presented by the second example. Finally, the third example presents a comparison between the MCCR algorithm and some algorithms that are usually utilized for estimating sparse vectors. In all these examples, the parameter ϵ, which

TABLE III
THE PMCCR ALGORITHM

Algorithm 2: The Perturbed MCCR Algorithm (PMCCA)

Initialization: Select one of the objective functions in Table I; select a feasible solution $\boldsymbol{s}$ $j = 0$.

1) For the selected objective function, execute MCCR with initial value $\boldsymbol{s}_0$ to obtain
2) Repeat until convergence:
   - evaluate perturbed feasible solution $\boldsymbol{s}_j^p = \bar{\boldsymbol{s}}_j^0 + \boldsymbol{Fu}$. Refer to the text for the
   - execute MCCR with initial value $\boldsymbol{s}_j^p$ to obtain $\bar{\boldsymbol{s}}_j^1$.
   - if $||\bar{\boldsymbol{s}}_j^1||_{\ell_0} \leq ||\bar{\boldsymbol{s}}_j^0||_{\ell_0}$, $\bar{\boldsymbol{s}}_{j+1}^0 = \bar{\boldsymbol{s}}_j^1$.
     else $\bar{\boldsymbol{s}}_{j+1}^0 = \bar{\boldsymbol{s}}_j^0$.
   - $j \leftarrow j + 1$

End

3) Output $\bar{\boldsymbol{s}}_{j+1}^0$ as the solution.

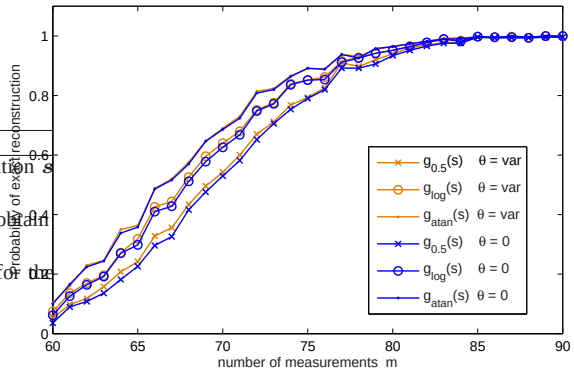
is used with the IRLS, MCCR and PMCCR algorithms, is selected as in Table II, i.e., each algorithm is started with $\epsilon = 1$. Then the value of ϵ is changed to $\epsilon \leftarrow \epsilon/10$ whenever the condition $||\boldsymbol{s}_{k+1} - \boldsymbol{s}_k||_2/||\boldsymbol{s}_{k+1}||_2 < \sqrt{\epsilon}/100$ is met.

In all these examples, two parameters are used as measures of performance. The first parameter is the probability of exact reconstruction (PER) of the solution vector. The PER is defined as the ratio between the number of runs at which the algorithm successfully estimates the sparse solution vector, to the total number of runs. The second parameter is the average number of iterations taken by each algorithm to converge to a solution vector. Each one of these two parameters is plotted as a function of the number of measurements (m).

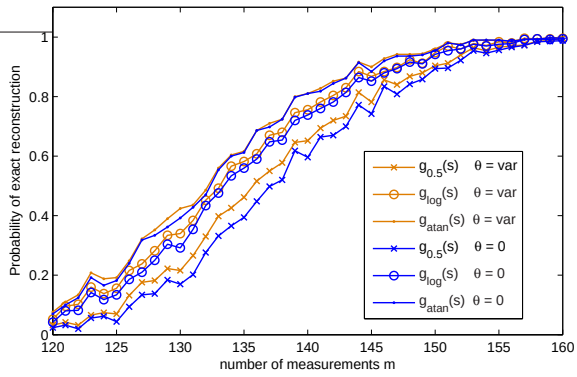
Example 1. Effect of θ*:* In this Example we compare the effect of θ on the performance of MCCR. Recall that the difference between the MCCR algorithm (24) and the IRLS algorithm (15) is the parameter θ. Three different objective functions are incorporated in this comparison. The functions are $g_{0.5}(s) = |s|^{0.5}$, $g_{log}(s)$, and $g_{atan}(s)$. Since the computation of θ requires solving a nonlinear optimization problem (27), it is important to compare not just the average number of iterations but also the average amount of time taken by each algorithm to converge to a solution vector. For doing this, the results are calculated for two different conditions. In the first condition, the low dimensional case, the parameters are $l = 30$, $n = 256$, and m increases from 60 to 90, while for the second condition, the high dimensional case, the parameters are $l = 60$, $n = 512$, and m increases from 120 to 160, where l is the number of nonzero entries in the exact solution vector.

For the two cases, the measuring matrix $\boldsymbol{A}$ and the sparse vector $\boldsymbol{s}^*$ are generated as follows. For each value of m, a random $(m \times n)$ matrix $\boldsymbol{A}$ is created whose entries are each Gaussian random variables with zero mean and unit variance. A sparse vector $\boldsymbol{s}^* \in \mathbb{R}^n$ with l nonzero entries is then created and the corresponding $\boldsymbol{x}$ is generated as $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}^*$. The indices of these l entries of $\boldsymbol{s}^*$ are randomly selected, and their values are chosen randomly from a zero mean Gaussian random variable with variance = 4. The total number of runs for each case is 500, and the results are shown in Fig. 4. The results for MCCR are presented in faint brown while the results for IRLS ($\theta = 0$) are presented in blue.

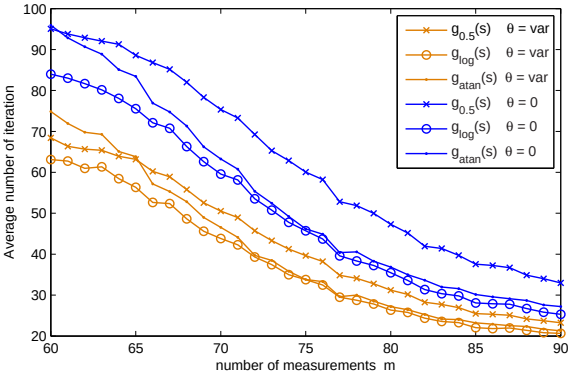
As shown in Fig. 4(a) and Fig. 4(b) for the two conditions



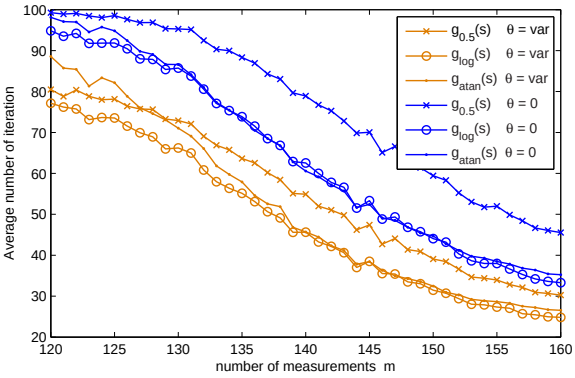
(a)



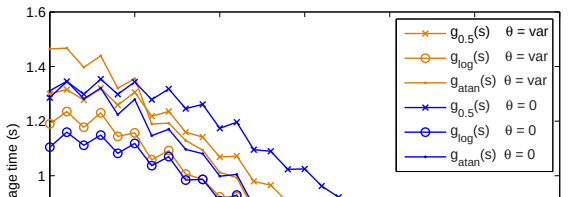
(b)



(c)



(d)

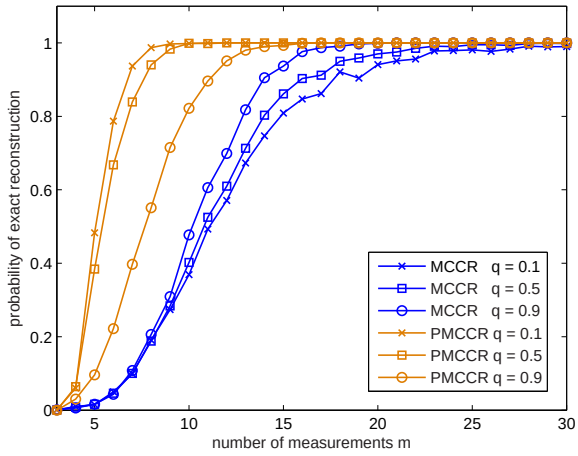


and for all the objective functions, MCCR and IRLS have almost similar PERs, meaning that they almost always converge to the same solution vectors. However, comparing the average number of iterations taken by MCCR and IRLS algorithms in Fig. 4(c) and Fig. 4(d) it is clear that MCCR takes fewer iterations than IRLS to converge to the solution vectors. For example, in Fig. 4(d), and for the objective function $g_{0.5}(s)$ and the case $m = 140$ we find that, on average, IRLS converges after 80 trials while MCCR converges after 54 trials only. However, since the computation of θ requires extra computational time, it is important to check the average time taken by each algorithm to converge to a solution vector. This comparison is shown in Fig. 4(e) and Fig. 4(f).

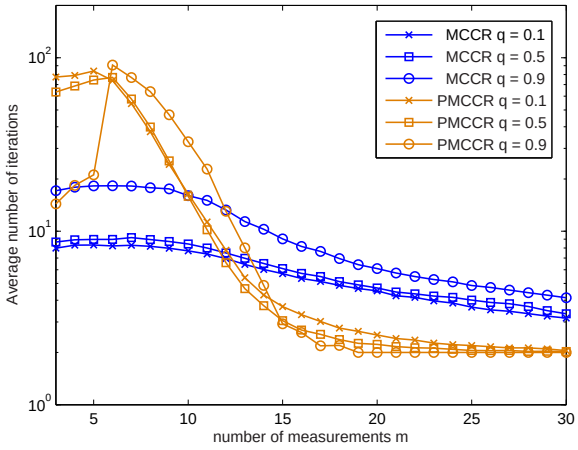
For the low dimensional case presented in Fig. 4(e) it is clear that, except for $g_{0.5}(s)$, the two algorithms take on average the same amount of time to converge to a solution vector, which implies that there is no advantage of replacing the IRLS algorithm by the MCCR algorithm for low dimensional problems. However, the average computational time could be significantly reduced if a faster algorithm is used for calculating θ, or if the computational cost of θ is low compared with the inversion operation in (24). This later case is presented in Fig. 4(f). From this figure it is clear that MCCR in this case converges in a shorter time than IRLS. Accordingly, we conclude this example by recommending the MCCR algorithm to solve large scale problems.

Example 2. Comparison between MCCR and PMCCR *:* In this example we examine the effect of the perturbation approach on the performance of the PMCCR. Specifically, we compare the performance of the MCCR and the PMCCR algorithm for the case $g(\boldsymbol{s}) = ||\boldsymbol{s}||_{\ell_q}^q$ for the following values of q; 0.1, 0.5, and 0.9. The comparison is made in terms of the probability of exact reconstruction of the sparse vector, and the average number of iterations required for convergence to that solution vector, as functions of the number of observations m, when both n and l are fixed at 40 and 3, respectively. For each value of m, $\boldsymbol{A}$ and $\boldsymbol{s}^*$ are generated as in Example 1. For the two algorithms, the values of ϵ and θ are selected as in Table II. The results are shown in Fig. 5.

As shown in Fig. 5(a) the probability of the exact reconstruction of the PMCCR algorithm is significantly better than that of the MCCR algorithm. For a low value of m, e.g., $m = 10$, the probability of exact reconstruction for PMCCR is 100% for $q = 0.1$ and 0.5. The number of iterations taken by each algorithm is presented in Fig. 5(b). The number of iterations of the PMCCR algorithm presented in Fig. 5(b) is the number of times the MCCR algorithm was called by the PMCCR algorithm, i.e., the final value of j in Table III. It is obvious that the PMCCR algorithm takes large number of iterations compared with the MCCR algorithm, especially when the number of measurements is small, e.g., $m < 10$ in this case. However, for larger m, e.g., $m \geq 15$, the performance of PMCCR in terms of successful reconstruction is 100%, while the number of outer iterations required is roughly less than or equal to 3. Therefore, PMCCR can be seen as a high-performance, more costly method for difficult cases (i.e., low values of m), whereas it is a higher–performance method at roughly the same cost, for cases which work well



(a)



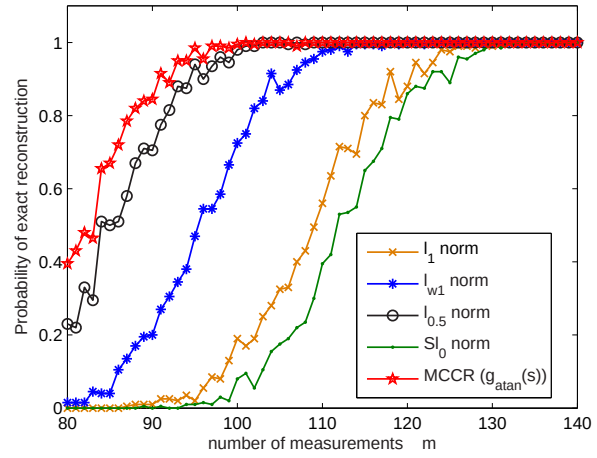
(b)

Fig. 5. Comparison between MCCR and PMCCR. (a) The probability of exact reconstruction of the original sparse vector; (b) The average number of iterations required to get a sparse solution vector. The number of iterations of the PMCCR algorithm represents the value of j in Table III.

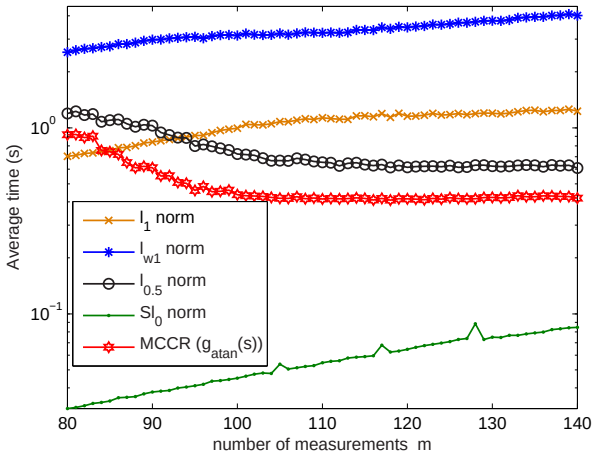
with other methods.

Example 3. Comparison between MCCR and other algorithms*:* We conclude this paper by a comparison between MCCR and four different algorithms. The four algorithms are: the ℓ_1–norm; the weighted ℓ_1–norm [12], denoted as ℓ_{w1}–norm; the $\ell_{0.5}$–norm implemented using the algorithm presented in [19], which is equivalent to MCCR when $g_c(s) = |s|^{0.5}$ and $\theta = 0$; and the recently developed algorithm called smooth ℓ^0–norm algorithm [37], denoted as Sl_0 in Fig. 6. The objective function used with MCCR is $g_{atan}(s)$. In this example, $n = 256$, $l = 40$, and m increases from 80 to 140. For each value of m, $\boldsymbol{A}$ and $\boldsymbol{s}^*$ are generated as in Example 1. The total number of runs in this example is 200, and the results are presented in Fig. 6.

As shown in Fig. 6(a) MCCR has the best performance in terms of the PER, while the smooth ℓ^0–norm algorithm has the worst performance among the compared algorithms. Also it is clear from Fig. 6(a) that the PER of $\ell_{0.5}$–norm algorithm is very close to that of MCCR and the ℓ_{w1}–norm performs better than the regular ℓ_1–norm. However, in terms of the average



(a)



(b)

Fig. 6. Comparison between MCCR and four different algorithms. (a) The probability of exact reconstruction of the original sparse vector as a function of the number of measurements (m); (b) The average number of iterations required to get a sparse solution vector.

amount of time shown in Fig. 6(b) it is clear that, aside from the smooth ℓ^0–norm, the MCCR is the fastest algorithm, while the ℓ_{w1}–norm is the most computationally expensive one. Although both MCCR and $\ell_{0.5}$–norm algorithms estimated all the solution vectors exactly for all $m > 100$, the MCCR converged in a significantly shorter time. For example, for the case when $m = 100$, MCCR converged in 0.43 seconds while the $\ell_{0.5}$–norm algorithm converged in 0.72 seconds.

VII. Conclusion

In this paper a novel methodology was developed and employed to minimize a class of non–convex (concave on the non–negative orthant) functions for solving an under–determined system of linear equations for the case of sparse solution vector. The proposed technique is based on locally replacing the original objective function by a quadratic convex function which is easily minimized. It was shown in this paper, for a certain selection of the convex objective function, the Iterative Re–weighted Least Squares (IRLS) class of algorithms are a special case of the proposed methodology. Thus the proposed algorithms are a generalization and unification of the previous methods. In this paper we also proposed a convex objective function that produces an algorithm that can converge to the sparse solution vector in a significantly fewer number of iterations than the IRLS algorithms. Other selections of the convex objective function may produce algorithms with convergence properties better than the IRLS algorithms.

In this paper we also proposed a straightforward technique for selecting a convex function such that, for any starting solution vector $\boldsymbol{s}_0$, the algorithm generates a sequence $\{\boldsymbol{s}_k\}_{k=1}^{\infty}$ that converges to a fixed point of the original objective function. Since the original objective functions are non-convex, the proposed algorithm is susceptible to convergence to a local minimum. To alleviate this difficulty, we proposed a random perturbation technique that enhances the performance of the proposed algorithm. Simulation results were presented to examine the performance of the proposed algorithm and to compare its performance with some well known algorithms that are usually utilized for solving the same problem. The simulation results show that the proposed algorithm outperforms the existing algorithms in terms of the execution time and the accuracy of reconstructing a sparse solution vector.

References

[1] P. Rodríguez and B. Wohlberg, "An iterative reweighted norm algorithm for total variation regularization," IEEE Signal Processing Letters, vol. 14, no. 12, pp. 948–951, Dec 2007.

[2] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," IEEE Trans. Acoust., Speech, Signal Processing, vol. 41, pp. 33973415, Dec. 1993.

[3] G. H. Golub and C. F. Van Loan, "Matrix Computations." Baltimore: The Johns Hopkins University Press, 1989.

[4] S. Lesage, R. Gribonval, F. Bimbot, and L. Benaroya, "Learning unions of orthonormal bases with thresholded singular value decomposition," in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Philadelphia, PA, Mar. 1823, 2005, vol. 5, pp. v/293v/296.

[5] Y. Li, A. Cichocki, and S. Amari, "Sparse Component Analysis for Blind Source Separation With Less Sensors Than Sources," in Proc. 4th Int. Symp. Independent Component Analysis Blind Source Separation (ICA BSS), 2003, pp. 89-94.

[6] Y. Li, A. Cichocki, S. Amari, S. Shishkin, J. Cao, and F. Gu, "Sparse Representation and Its Applications in Blind Source Separation," in 7th Annual Conference on Neural Information Processing Systems (NIPS-2003), (Vancouver), Dec. 2003.

[7] I. F. Gorodnitsky and B. D. Rao, "Sparse signal reconstructions from limited data using FOCUSS: A re-weighted minimum norm algorithm," IEEE Trans. Signal Processing, vol. 45, pp. 600-616, Mar. 1997.

[8] P. Xu, Y. Tian, H. Chen, and D. Yao, "LP norm iterative sparse solution for EEG source localization," IEEE Trans. Biomed. Eng., vol. 54, no. 3, pp. 400409, Mar. 2007.

[9] K. Kreutz-Delgado and B. D. Rao, "FOCUSS-based dictionary learning algorithms," in Proc. Wavelet Appl. Signal Image Process., Bellingham, WA, Jul.-Aug. 2000, vol. 4119, pp. 459473.

[10] M. Aharon, M. Elad, and A. Bruckstein, "The K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," IEEE Trans. Signal Process., vol. 54, no. 11, pp. 43114322, Nov. 2006.

[11] B. Jeffs and M. Gunsay, "Restoration of blurred star field images by maximally sparse optimization," IEEE Trans. Image Process., vol. 2, no. 2, pp. 202211, Mar. 1993.

[12] E. J. Candès, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted ℓ_1 minimization," Journal of Fourier Analysis and Applications, vol. 14, no. 5, pp. 877-905, special issue on sparsity, December 2008.

[13] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," IEEE Trans. Signal Process., vol. 41, no. 12, pp. 33973415, Dec. 1993.

[14] J. A. Tropp, A. C. Gilbert, and M. J. Strauss, "Algorithms for simultaneous sparse approximation, Part I: Greedy pursuit," Signal Process., vol. 86, no. 3, pp. 572588, 2006.

[15] S. Chen and D. Donoho, "Basis pursuit," in Proc. Twenty-Eighth Asilomar Conf. Signals, Syst. Comput., Monterey, CA, Nov. 1994, vol. I, pp. 41-44.
[16] S. S. Chen, D. L. Donoho, and M. A. Saunder, "Atomic decomposition by basis pursuit," SIAM J. Sci. Comput., vol. 20, no. 1, pp. 3361, 1998.
[17] T. Blumensath, M. E. Davies, "Iterative Thresholding for Sparse Approximations," The Journal of Fourier Analysis and Applications, vol.14, no 5, pp. 629-654, Dec. 2008
[18] B. D. Rao and K. Kreutz-Delgado, "An affine scaling methodology for best basis selection," IEEE Trans. Signal Process., vol. 47, no. 1, Jan. 1999.
[19] R. Chartrand and W. Yin, "Iteratively Reweighted Algorithms for Compressive Sensing," in Proc. Acoustics, Speech and Signal Processing, ICASSP 2008, pp. 3869-3872.
[20] I. Daubechies, R. DeVore, M. Fornasier, and S. Gunturk, " Iteratively re-weighted least squares minimization for sparse recovery," to appear in Commun. Pure Appl. Math. 2009.
[21] D. Baron,M. F. Duarte,M. B. Wakin,S. Sarvotham, and R. G. Baraniuk, "Distributed Compressive Sensing," arXiv:0901.3403v1, Jan. 2009.
[22] E. Candes and T. Tao, "Near optimal signal recovery from random projections: universal, encoding strategies?" IEEE Trans. Inf. Theory, vol. 52, pp. 5406-5425, 2006.
[23] D. L. Donoho and J. Tanner, "Thresholds for the recovery of sparse solutions via L1 minimization," 40th Annual Conference on Information Sciences and Systems, 2006, pp 202-206.
[24] R. Chartrand, "Exact reconstruction of sparse signals via nonconvex minimization," IEEE Signal Process. Lett., vol. 14, no. 10, Oct. 2007.
[25] N. Mourad and J. Reilly, "ℓ_p Minimization for Sparse Vector Reconstruction", *ICASSP2009*, Taipei, Taiwan, Apr. 19-24, 2009.
[26] R. Chartrand and V. Staneva, "Restricted isometry properties and nonconvex compressive sensing", Inverse Problems, vol. 24, no. 035020, pp. 1–14, 2008.
[27] Y. Li, "A globally convergent method for ℓ_p problems," SIAM J. Optim. vol. 3, no. 3, pp. 609629, 1993.
[28] D. L. Donoho and Y. Tsaig, "Fast solution of ℓ_1-norm minimization problems when the solution may be sparse," preprint (2006).
[29] R. Wolke and H. Schwetlick, "Iteritevely reweighted least squares: algorithms, convergence analysis, and numerical comparisons," SIAM J. Sci. Stat. Comput., vol. 9, no. 5, pp. 907-921, 1988.
[30] R. Wolke, "Iteratively reweighted least squares: A comparison of several single step algorithms for linear models," BIT Numerical Mathematics, vol. 32, no. 3, pp. 506-524, 1992.
[31] S. L. Campbell and C. D. Meyer, "Generalized inverse of linear transformations," London, U. K.: Pitman, 1979.
[32] Nasser Mourad, "Advances in sparse signal analysis with applications to blind source separation and EEG/MEG signal processing," Ph.D. dissertation, McMaster University, Hamilton, On., Canada, 2009.
[33] D. R. Hunter and K. Lange, " A tutorial on MM algorithms," The American Statistician, vol. 58, no. 1, pp. 30-37, 2004.
[34] R. G. Baraniuk, "Compressive Sensing," IEEE Trans. Signal Proc., vol. 24, no. 4, pp. 118 - 121, July 2007.
[35] S. Osher, Y. Mao, B. Dong, and A. W. Yin, "Fast linearized bregman iteration for compressive sensing and sparse denoising," Available at ftp://ftp.math.ucla.edu/pub/camreport/cam08-37.pdf
[36] W. Yin, S. Osher, D. Goldfarb, and J. Darbon, "Bregman iterative algorithms gor ℓ_1-minimization with applications to compresed sensing," SIAM. J. Imaging Science, vol. 1, no. 1, pp. 143-168, 2008.
[37] G. H. Mohimani, M. Babaie-Zadah, and C. Jutten, "A Fast approach for overcomplete sparse decomposition based on smoothed ℓ^0 norm," IEEE Trans. Signal Processing, vol. 57, no. 1, pp. 289-301, Jan. 2009.