remote sensing

MDPI

Article

Fast, Efficient, and Viable Compressed Sensing, Low-Rank, and Robust Principle Component Analysis Algorithms for Radar Signal Processing

Reinhard Panhuber

Fraunhofer FHR, Fraunhofer Institute for High Frequency Physics and Radar Techniques FHR, 53343 Wachtberg, Germany; reinhard.panhuber@fhr.fraunhofer.de

Abstract: Modern radar signal processing techniques make strong use of compressed sensing, affine rank minimization, and robust principle component analysis. The corresponding reconstruction algorithms should fulfill the following desired properties: complex valued, viable in the sense of not requiring parameters that are unknown in practice, fast convergence, low computational complexity, and high reconstruction performance. Although a plethora of reconstruction algorithms are available in the literature, these generally do not meet all of the aforementioned desired properties together. In this paper, a set of algorithms fulfilling these conditions is presented. The desired requirements are met by a combination of turbo-message-passing algorithms and smoothed ℓ_0-refinements. Their performance is evaluated by use of extensive numerical simulations and compared with popular conventional algorithms.

Keywords: compressed sensing; affine rank minimization; robust principle component analysis; complex valued; radar signal processing

check for updates

Citation: Panhuber, R. Fast, Efficient, and Viable Compressed Sensing, Low-Rank, and Robust Principle Component Analysis Algorithms for Radar Signal Processing. *Remote Sens.* **2023**, *15*, 2216. https://doi.org/10.3390/rs15082216

Academic Editor: Lorenzo Capineri

Received: 14 March 2023
Revised: 13 April 2023
Accepted: 18 April 2023
Published: 21 April 2023

1. Introduction

The compressive sensing (CS), affine rank minimization (ARM), and compressed robust principal component analysis (CRPCA) methods are enjoying great popularity in terms of their application in modern radar signal processing. At present, they are being applied in almost every possible field of application ranging from basic radar signal processing [1–4] to more sophisticated applications such as multiple-input multiple-output (MIMO) radar [5,6], ground moving target indication (GMTI) [7–11], synthetic aperture radar (SAR) [12–14], inverse synthetic aperture radar (ISAR) [15–18], interference and clutter mitigation [19–22], or SAR-GMTI [23–25], imaging [26], and passive radar [27], to name a few—this list is far from being exhaustive. Many more examples of CS applied to radar signal processing can be found e.g., in [28]. Alongside the specific properties of the application at hand, the success of these methods depends also on the algorithms for solving the emerging linear inverse problems. Many CS, ARM, and CRPCA algorithms found in the literature do not consider the practical requirements of radar signal processing. They either suffer from restrictions to real numbers, a slow convergence rate, or low reconstruction performance. Furthermore, many fast converging algorithms assume knowledge of generally unknown parameters, such as the precise number of sparse entries or the exact rank of a low-rank matrix. An overview of available algorithms is given in [29,30]. In this paper, a complete set of algorithms to solve general CS, ARM, and CRPCA problems is introduced that have the aforementioned properties. Due to the structural similarity of CS and ARM problems, a unified framework to solve them as well as the combined CRPCA problem can be formulated. The desirable properties are achieved by combining, augmenting, and extending turbo-message-passing algorithms and smoothed ℓ_0-refinements. The turbo-message-passing framework provides a fast converging approach to obtain an initial ℓ_1- or convex solution. The smoothed ℓ_0-approach further improves upon the

convex solution by enforcing a stricter sparsity or rank measure and as such improves the reconstruction performance. The presented algorithms are termed

- Turbo shrinkage-thresholding (TST)
- Complex successive concave sparsity approximation (CSCSA)
- Turbo singular value thresholding (TSVT)
- Complex smoothed rank approximation (CSRA)
- Turbo compressed robust principal component analysis (TCRPCA)

where TST and CSCSA apply to CS problems, TSVT and CSRA to ARM problems, and TCRPCA allows for solving combined CS and ARM problems. These algorithms are designed such that no parameters that are unknown in practice are required, and for unavoidable parameters, equations for their determination are given. The only parameter assumed to be known is noise power P_{n}, which is a reasonable assumption in the field of radar applications. Furthermore, these algorithms offer a very high convergence rate alongside low computational complexity due to the use of closed solutions of subsequent optimization problems.

In this paper, the presented algorithms are evaluated in terms of extensive numerical simulations. These focus on comparing the algorithms, as generally as possible, to popular algorithms from the literature. For this purpose, phase transition plots, convergence and computation speed, and reconstruction performance in terms of signal to noise ratio (SNR) are suitable. The application of the presented algorithms to radar applications with real measurement data is a subject for future publications.

1.1. Background

Since CS, ARM, and CRPCA frameworks are very similar in structure, it is convenient to present them in a combined way. Their corresponding objectives are to recover solutions from limited noisy observations of the respective forms [31–33]

$$\begin{aligned}\boldsymbol{y} &= \boldsymbol{A}\widetilde{\boldsymbol{s}} + \boldsymbol{n}\\ \boldsymbol{y} &= \mathcal{A}(\widetilde{\boldsymbol{L}}) + \boldsymbol{n}\\ \boldsymbol{y} &= \mathcal{A}(\widetilde{\boldsymbol{S}} + \widetilde{\boldsymbol{L}}) + \boldsymbol{n},\end{aligned}$$

where $\widetilde{\boldsymbol{s}} \in \mathbb{C}^n$ is an unknown sparse vector with $\kappa \ll n$ entries, $\widetilde{\boldsymbol{S}} \in \mathbb{C}^{N_1 \times N_2}$ is a corresponding sparse matrix with $\kappa \ll n = N_1 N_2$ entries, $\widetilde{\boldsymbol{L}} \in \mathbb{C}^{N_1 \times N_2}$ is an unknown matrix whose $\operatorname{rank}(\widetilde{\boldsymbol{L}}) = \rho \ll n_{\min} = \min(N_1, N_2)$, $\boldsymbol{A} \in \mathbb{C}^{m \times n}$ ($m < n$) and $\mathcal{A} : \mathbb{C}^{N_1 \times N_2} \to \mathbb{C}^m$ ($m < N_1 N_2 = n$) are known affine transformations, $\boldsymbol{y} \in \mathbb{C}^m$ is a measurement vector, and $\boldsymbol{n} \in \mathbb{C}^m$ is additive noise with complex normal i. i. d. coefficients of zero mean and variance P_{n}. To find a solution to these under-determined linear systems, the closest sparse and low-rank solutions consistent with the measurements are sought via

$$\min_{\boldsymbol{s}} \|\boldsymbol{s}\|_0 \text{ subject to } h(\boldsymbol{s}) \leq \varepsilon^2 \tag{1}$$

$$\min_{\boldsymbol{L}} \operatorname{rank}(\boldsymbol{L}) \text{ subject to } h(\boldsymbol{L}) \leq \varepsilon^2 \tag{2}$$

$$\min_{\boldsymbol{S},\boldsymbol{L}} \lambda \|\boldsymbol{S}\|_0 + \operatorname{rank}(\boldsymbol{L}) \text{ subject to } h(\boldsymbol{S} + \boldsymbol{L}) \leq \varepsilon^2, \tag{3}$$

where

$$h(\boldsymbol{s}) = \|\boldsymbol{A}\boldsymbol{s} - \boldsymbol{y}\|_2^2 \tag{4}$$

and

$$h(\boldsymbol{X}) = \|\mathcal{A}(\boldsymbol{X}) - \boldsymbol{y}\|_2^2 \tag{5}$$

are the data fidelity terms with $\|\cdot\|_p$ denoting the ℓ_p-norm, and $\varepsilon^2 \geq \|\boldsymbol{n}\|_2^2$ is some constant error energy. It is well known that, in general, (1)–(3) are NP-hard to solve [33]. This has given rise to the plethora of algorithms that seek approximate formulations of (1) to (3) in order to find solutions in a computationally tractable manner.

1.2. State of the Art

The most popular relaxations to the respective problems (1) to (3) can be categorized into the families of convex relaxation and greedy algorithms [34]. Further approaches comprise hard thresholding (HT), smoothed-ℓ_0, and approximated message passing (AMP) algorithms.

All of the aforementioned approaches require the restricted isometry property (RIP) and restricted rank isometry property (RRIP) conditions

$$(1-\delta_K(\boldsymbol{A}))\|\boldsymbol{s}\|_2^2 \leq \|\boldsymbol{A}\boldsymbol{s}\|_2^2 \leq (1+\delta_K(\boldsymbol{A}))\|\boldsymbol{s}\|_2^2 \tag{6}$$

$$(1-\delta_R(\mathcal{A}))\|\boldsymbol{L}\|_{\mathrm{F}}^2 \leq \|\mathcal{A}(\boldsymbol{L})\|_{\mathrm{F}}^2 \leq (1+\delta_R(\mathcal{A}))\|\boldsymbol{L}\|_{\mathrm{F}}^2, \tag{7}$$

To be fulfilled for all $\|\boldsymbol{s}\|_0 \leq K$ and $\operatorname{rank}(\boldsymbol{L}) \leq R$ by the sensing operators $\boldsymbol{A}$ and $\mathcal{A}$ for some constants δ_K and δ_R [33,35]. Basically, (6) and (7) ensure that the sensing operators $\boldsymbol{A}$ and $\mathcal{A}$ keep different sparse vectors with at most K entries and different matrices of rank $\rho \leq R$ distinguishable. Furthermore, for CRPCA, the sparse and low-rank matrices $\widetilde{\boldsymbol{S}}$ and $\widetilde{\boldsymbol{L}}$ must be distinguishable to allow for (3) to have a unique solution. This means that the sparse matrix $\widetilde{\boldsymbol{S}}$ must not be low ranking and, likewise, the low-rank matrix $\widetilde{\boldsymbol{L}}$ must not be of sparse nature. This condition is known as the rank-sparsity incoherence condition, which is fulfilled if the maximum number of nonzero entries per row and column in $\widetilde{\boldsymbol{S}}$ and the incoherence of $\boldsymbol{L}$ with respect to the standard basis are sufficiently small [36]. The first condition not only limits the total number of sparse entries in $\widetilde{\boldsymbol{S}}$ but also demands that its support is not clustered. The second condition demands that the singular vectors of $\widetilde{\boldsymbol{L}}$ are not sparse, i.e., are reasonably spread out. This entails the consequence that the maximum entry in magnitude of the dyadic $\boldsymbol{U}\boldsymbol{V}^{\mathrm{H}}$ for a given upper μ-incoherence is bounded as [32]

$$\left\|\boldsymbol{U}\boldsymbol{V}^{\mathrm{H}}\right\|_{\infty} \leq \frac{\mu\rho}{\sqrt{N_1 N_2}}, \tag{8}$$

where $\widetilde{\boldsymbol{L}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{H}}$ is the singular value decomposition (SVD) of $\widetilde{\boldsymbol{L}}$. This means $\widetilde{\boldsymbol{L}}$ must not contain spiky entries. In the following, the individual approaches and their advantages and disadvantages are briefly discussed.

1.2.1. Greedy Algorithms

The idea of greedy algorithms is to apply some heuristic approach that locally optimizes the objective functions (1) to (3) in an iterative procedure. Famous examples of CS greedy algorithms are orthogonal matching pursuit (OMP) [37], stagewise OMP (StOMP) [38], regularized OMP (ROMP) [39], its improved version compressive sampling matching pursuit (CoSaMP) [40], and subspace pursuit (SP) [41], though this list is far from exhaustive. The famous OMP algorithm and its improved version StOMP have a very simple structure that, however, goes along with the disadvantage that some incorrect sample index, once added to the support (which may happen in general), cannot be removed any more [42]. The CoSaMP and SP algorithms circumvent this problem by adding a backward step to prune wrongly selected support. This, however, requires knowledge of the sparsity level κ and rank ρ. Famous ARM and CRPCA greedy algorithms are atomic decomposition for minimum rank approximation (ADMiRA) and SpaRCS, respectively, where ADMiRA was inspired by CoSaMP and SpaRCS combines CoSaMP and ADMiRA [43,44]. Depending on the problem size, greedy algorithms are fast and of low computational complexity. However, they are known to be sensitive to noise [45,46]. Another issue arises in the context of closely spaced radar targets, for which greedy algorithms tend to merge such targets into a single one.

1.2.2. Hard Thresholding Algorithms

The objective functions of HT algorithms are [47–49]

$$\min_{\boldsymbol{s}} h(\boldsymbol{s}) \text{ subject to } \|\boldsymbol{s}\|_0 \leq K \tag{9}$$

$$\min_{\boldsymbol{L}} h(\boldsymbol{L}) \text{ subject to } \operatorname{rank}(\boldsymbol{L}) \leq R \tag{10}$$

$$\min_{\boldsymbol{L}} h(\boldsymbol{S}+\boldsymbol{L}) \text{ subject to } \operatorname{rank}(\boldsymbol{L}) \leq R, \|\boldsymbol{S}\|_0 \leq K, \tag{11}$$

where $K \in \mathbb{N}$ and $R \in \mathbb{N}$ are some chosen constants. Famous examples are, e.g., normalized iterative hard thresholding (NIHT) for CS tasks [47], singular value projection (SVP) for ARM problems [48], and an extended combination of the aforementioned algorithm termed nonconvex free lunch (NFL) for CRPCA problems [49]. These algorithms have a very simple structure. They consist of a gradient update step followed by a HT operation as

$$\boldsymbol{s}_{i+1} = \mathcal{H}_{\mathrm{s},K}(\boldsymbol{s}_i - \mu_i \nabla_{\boldsymbol{s}} h(\boldsymbol{s}_i)) \tag{12}$$

$$\boldsymbol{L}_{i+1} = \mathcal{H}_{\mathrm{l},R}(\boldsymbol{L}_i - \mu_i \nabla_{\boldsymbol{L}} h(\boldsymbol{L}_i)). \tag{13}$$

Within (12),

$$[\mathcal{H}_{\mathrm{s},K}(\boldsymbol{x})]_i = \begin{cases} x_i, & \text{if } |x_i| \geq \boldsymbol{x}^{[K]} \\ 0, & \text{else} \end{cases} \tag{14}$$

denotes the sparsity HT operator with $[\boldsymbol{x}]_i$ being the ith entry and $\boldsymbol{x}^{[K]}$ the Kth biggest entry in magnitude in vector $\boldsymbol{x}$. As such, $\mathcal{H}_{\mathrm{s},K}(\boldsymbol{x})$ sets all but the K largest elements in magnitude of $\boldsymbol{x}$ to zero. Likewise, in (13),

$$\mathcal{H}_{\mathrm{l},R}(\boldsymbol{X}) = \sum_{i=1}^{R} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{H}} \tag{15}$$

denotes the low-rank HT operator, where $\boldsymbol{X} = \boldsymbol{U}\operatorname{diag}(\boldsymbol{\sigma})\boldsymbol{V}^{\mathrm{H}}$ is the SVD of $\boldsymbol{X}$ with $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ denoting the ith column vector of $\boldsymbol{U}$ and $\boldsymbol{V}$, and $\boldsymbol{\sigma} = \begin{bmatrix}\sigma_1 & \cdots & \sigma_{n_{\min}}\end{bmatrix}^{\mathrm{T}}$ is a vector holding the singular values of $\boldsymbol{X}$ in decreasing order. As such, $\mathcal{H}_{\mathrm{l},R}(\boldsymbol{X})$ sets all but the R largest singular values of $\boldsymbol{X}$ to zero. The step sizes μ_i in the update steps need to be updated per iteration, which is crucial for reconstruction success [47]. This approach shows a high convergence rate and offers guaranteed reconstruction performance provided $K \approx \kappa$ and $R \approx \rho$. In the case of wrongly chosen parameters K and R, however, the reconstruction performance deteriorates significantly, as will be illustrated further below. The convergence rate was found to be higher than for the abovementioned greedy algorithms, although its reconstruction performance is slightly worse [50].

1.2.3. Convex Relaxations Algorithms

Another famous family comprises convex relaxations via the basis pursuit denoise (BPDN) and stable principal component pursuit (SPCP) approach [51,52]. The objective Functions (1) to (3) are relaxed to

$$\min_{\boldsymbol{s}} \|\boldsymbol{s}\|_1 \text{ subject to } h(\boldsymbol{s}) \leq \varepsilon^2 \tag{16}$$

$$\min_{\boldsymbol{L}} \|\boldsymbol{L}\|_* \text{ subject to } h(\boldsymbol{L}) \leq \varepsilon^2 \tag{17}$$

$$\min_{\boldsymbol{L}} \lambda_s \|\boldsymbol{S}\|_1 + \|\boldsymbol{L}\|_* \text{ subject to } h(\boldsymbol{S}+\boldsymbol{L}) \leq \varepsilon^2, \tag{18}$$

where $\|\cdot\|_*$ denotes the nuclear norm. Famous examples that solve the regularized unconstrained formulations of (16) to (18)

$$\widehat{\boldsymbol{s}} = \arg\min_{\boldsymbol{s}} \lambda_s \|\boldsymbol{s}\|_1 + h(\boldsymbol{s}) \tag{19}$$

$$\widehat{\boldsymbol{L}} = \arg\min_{\boldsymbol{L}} \lambda_l \|\boldsymbol{L}\|_* + h(\boldsymbol{L}) \tag{20}$$

$$\widehat{\boldsymbol{S}}, \widehat{\boldsymbol{L}} = \arg\min_{\boldsymbol{S},\boldsymbol{L}} \lambda_l \|\boldsymbol{L}\|_* + \lambda_s \|\boldsymbol{S}\|_1 + h(\boldsymbol{S}+\boldsymbol{L}) \tag{21}$$

are fast iterative shrinkage-thresholding algorithm (FISTA) and spectral projected gradient for ℓ_1 (SPGL1) for CS [51,53], singular value thresholding (SVT) for ARM [31], and the variational approaches from [52] termed as SPCP. The respective underlying algorithm to FISTA, iterative shrinkage-thresholding algorithm (ISTA), and SVT show a similar structure as the HT algorithms, where a solution is obtained via the iterative procedures

$$\boldsymbol{s}_{i+1} = \mathcal{S}_{\mathrm{s},\mu\lambda}(\boldsymbol{s}_i - \mu_i \nabla_{\boldsymbol{s}} h(\boldsymbol{s}_i)) \tag{22}$$

$$\boldsymbol{L}_{i+1} = \mathcal{S}_{\mathrm{l},\mu\lambda}(\boldsymbol{L}_i - \mu_i \nabla_{\boldsymbol{L}} h(\boldsymbol{L}_i)). \tag{23}$$

Within (22), $\mathcal{S}_{\mathrm{s},a}(\boldsymbol{x})$ denotes the sparsity soft thresholding (ST) operator, which in this work is defined for complex numbers as

$$[\mathcal{S}_{\mathrm{s},a}(\boldsymbol{x})]_i = (|x_i| - a)_+ \operatorname{sgn}(x_i), \tag{24}$$

where $(\cdot)_+ = \max(0, \cdot)$ and

$$\operatorname{sgn}(z) = \begin{cases} 0 & \text{if } z = 0 \\ \frac{z}{|z|} & \text{else} \end{cases}$$

is the complex sign function [54]. Likewise, in (23),

$$\mathcal{S}_{\mathrm{l},a}(\boldsymbol{X}) = \sum_{i=1}^{n_{\min}} (\sigma_i - a)_+ \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{H}} \tag{25}$$

denotes the low-rank ST operator. Similar to the HT operators, the ST operators also possess an unknown parameter, namely the shrinkage parameter a. This shrinkage parameter, however, can be heuristically determined from the noise power P_{n}, as will be shown further below. The convex relaxation approach thus does not suffer from requiring parameters that are hard to determine but, however, are known for their slow convergence and low reconstruction performance. A famous acceleration approach offers FISTA, which boosts the convergence of ISTA, $\mathcal{O}(1/i)$, up to $\mathcal{O}(1/i^2)$ without increasing the computational complexity notably [53].

1.2.4. Approximated Message-Passing Algorithms

In recent years, another form of acceleration technique has gained huge interest, namely AMP, from which the so-called turbo algorithms emerged [55–59]. This technique further improves the convergence rate of ST and HT approaches for suitable sensing operators $\boldsymbol{A}$ and $\mathcal{A}$. The turbo algorithms presented in this work are inspired by these. To the best of the author's knowledge, Refs. [55–59] presented their algorithms for generic and in particular HT operators but do not elaborate on ST operators, which are of major interest in this work. Utilizing the turbo approach in combination with a ST operator, a convex relaxed solution for CS and ARM problems can be found. Closed form solutions for required parameters of the turbo approach are presented in this work.

1.2.5. Smoothed ℓ_0-Algorithms

The ℓ_1- and nuclear norms are the tightest convex approximations of the ℓ_0-norm and rank function, respectively [45,60]. However, it was found that the reconstruction error of solutions derived from convex relaxed approaches can be further reduced by applying

smoothed ℓ_0- and smoothed rank approximation frameworks [45,46,61–64]. Their purpose is to enforce a stricter sparsity or rank measure compared to the ℓ_1- and nuclear norm. Doing so yields a refinement algorithm which allows for an improved reconstruction performance. Among the smoothed ℓ_0-approaches, successive concave sparsity approximation (SCSA) appeals through its simplicity and efficiency due to the availability of closed form solutions of subsequent optimization steps. This algorithm was extended to the complex case named CSCSA [54] and serves, in the algorithm collection presented in this work, as the final refinement approach of CS applications. In the work presented here, a similar approach is applied to the smoothed rank approximation (SRA) algorithm from [64] to extend it into the complex case and to improve its efficiency due to the use of closed form solutions of subsequent minimization steps. The result called CSRA serves as the final refinement algorithm of ARM.

1.3. Contribution

In this paper, a set of algorithms applicable to radar signal processing tasks is presented. This is achieved by combining turbo-message-passing principles and smoothed ℓ_0-refinements. Available approaches from the literature are extended to the complex case and closed form solutions for required parameters and subsequent minimization problems are derived. The resulting algorithms have a high convergence rate and low computational complexity, offer high reconstruction performance, and are easy to implement and free of unknown parameters. The performance in terms of reconstruction error, convergence rate, computation time, etc., is compared with classical standard algorithms.

1.4. Outline of the Paper

In Section 2, the ℓ_1-norm minimization algorithm TST and its smoothed ℓ_0-refinement algorithm CSCSA are presented. Section 3 provides the nuclear norm minimization algorithm TSVT and the corresponding refinement algorithm CSRA. In Section 4, the CRPCA algorithm as a combination of the aforementioned algorithm is presented. Within all aforementioned sections, numerical simulations are presented to evaluate the proposed algorithms and to compare them with popular alternative algorithms. Finally, concluding remarks are drawn in Section 5.

2. Compressed Sensing

In the following section, the TST and CSCSA algorithms are introduced. The TST algorithm obtains an initial sparse ℓ_1- or convex solution, which is further refined by the CSCSA algorithm. Their combination yields all the desired properties of fast convergence, high reconstruction performance, no knowledge of generally unknown parameters etc.

2.1. Turbo Shrinkage Thresholding

The TST algorithm is inspired by the turbo algorithms presented in [55–57,59], which describe general reconstruction algorithms for CS and ARM problems applying the message-passing principle. This principle allows for a drastic improvement in the convergence speed of right-orthogonally invariant linear (ROIL) sensing operators to which random and discrete Fourier transform (DFT) sensing operators belong (consider a linear operator $\mathcal{A}$ with matrix form $\boldsymbol{A}$, the SVD of $\boldsymbol{A}$ is $\boldsymbol{A} = \boldsymbol{U}_A \boldsymbol{\Sigma}_A \boldsymbol{V}_A^{\mathrm{H}}$. If $\boldsymbol{V}_A$ is a Haar-distributed random matrix independent of $\boldsymbol{\Sigma}_A$, then $\mathcal{A}$ is a right-orthogonally invariant linear (ROIL) operator [57]). The two main contributions presented in this paper are (1) the provision of required formulas for the complex case and (2) a closed form solution of the required divergence of the ST denoiser, hence the name TST algorithm.

The TST algorithm attempts to find a solution to (1) by solving the convex relaxed regularized optimization problem (19). This is achieved using the turbo-framework iterative procedure [57]

$$\begin{aligned} \boldsymbol{r}_i &= \boldsymbol{s}_{i-1} - \mu \nabla_{\boldsymbol{s}} h(\boldsymbol{s}_{i-1}) \\ \boldsymbol{z}_i &= \mathcal{S}_{s,\mu\lambda}(\boldsymbol{r}_i) \\ \boldsymbol{s}_i &= c_i(\boldsymbol{z}_i - \alpha_i \boldsymbol{r}_i), \end{aligned} \tag{26}$$

where μ is some step size, i is the iteration index, and $\mathcal{S}_{s,a}$ is the soft thresholding operator given by (24). The required parameters $\{\mu, \alpha_i, c_i\}$ are chosen according to the turbo principle such that

$$(\boldsymbol{r}_i - \widetilde{\boldsymbol{s}})^{\mathrm{H}}(\boldsymbol{s}_{i-1} - \widetilde{\boldsymbol{s}}) = 0, \tag{27}$$

$$(\boldsymbol{r}_i - \widetilde{\boldsymbol{s}})^{\mathrm{H}}(\boldsymbol{s}_i - \widetilde{\boldsymbol{s}}) = 0, \tag{28}$$

and further that for a given $\boldsymbol{s}_{i-1}$, $\|\boldsymbol{s}_i - \widetilde{\boldsymbol{s}}\|_2^2$ is minimized under (27) and (28) [57]. In the above, (27) ensures that the input and output error of the gradient update step are uncorrelated. Equally, (28) ensures a decorrelation of the input and output error of the denoising or ST step. This strategy allows for an improved convergence rate compared with classical gradient approaches. In order to fulfill (27) and (28), the true solution $\widetilde{\boldsymbol{s}}$ is required to determine exact values for $\{\mu, \alpha_i, c_i\}$. Since $\widetilde{\boldsymbol{s}}$ is unknown, asymptotic formulations for $\{\mu, \alpha_i, c_i\}$ for $n \to \infty$ were derived in [57] for the real valued case. In the general complex case, these parameters can be adapted to

$$\mu = \frac{n}{m\|\boldsymbol{A}\|_2^2} \tag{29}$$

$$\alpha_i = \begin{cases} \frac{1}{n}\mathrm{div}\left(\mathcal{S}_{s,\mu\lambda}(\boldsymbol{r}_i)\right) & \text{if } \boldsymbol{s} \in \mathbb{R}^n \\ \frac{1}{2n}\mathrm{div}\left(\mathcal{S}_{s,\mu\lambda}(\boldsymbol{r}_i)\right) & \text{if } \boldsymbol{s} \in \mathbb{C}^n \end{cases} \tag{30}$$

$$c_i = \frac{(\boldsymbol{z}_i - \alpha_i \boldsymbol{r}_i)^{\mathrm{H}} \boldsymbol{r}_i}{\|\boldsymbol{z}_i - \alpha_i \boldsymbol{r}_i\|_2^2}, \tag{31}$$

where $\mathrm{div}(\cdot)$ is the weak divergence operator. For small scene sizes n, it was found from simulations that the step size μ required for convergence has to be reduced to $\mu < 1/\|\boldsymbol{A}\|_2^2$, which corresponds to the stable step size of the FISTA algorithm. This phenomena was not discussed in [55–57,59]; however, since the parameter Equations (29)–(31) are asymptotic approximations, their validity requires the scene size to be sufficiently large. Nevertheless, TST still shows superior convergence speed even in the reduced step size case. The required weak divergence is derived in Appendix A in closed form as

$$\mathrm{div}(\mathcal{S}_{s,a}(\boldsymbol{r})) = \sum_{i=1}^{n} \left(2 - \frac{a}{|r_i|}\right) \mathbb{I}(|r_i| > a), \tag{32}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function. The next question of course is how to choose the regularization parameter λ and thus which shrinkage $a = \mu\lambda$ should be applied. The usage of a ST operator causes the optimal solution $\boldsymbol{s}_{\mathrm{opt}}$ generated by the turbo framework (26) to possess an offset of a for every sparse entry compared with the true solution $\widetilde{\boldsymbol{s}}$. Too much shrinkage results therefore in a large bias, while too little results in a slow convergence rate. As such, an optimal constant λ does not exist; rather, λ would need to be adjusted in every iteration. Unfortunately, to the best of the author's knowledge, a closed form solution to determine such an optimal λ is not known to exist. A reasonable choice for a constant parameter λ is a scaled version of the formula of [65]

$$\lambda = {}^1\!/_{4c_r}\sqrt{P_{\mathrm{n}}}\Phi^{-1}\left(1 - \frac{\alpha_r}{2n}\right), \tag{33}$$

which was found from simulations to perform well as illustrated further below. The parameter $c_r > 1$ in (33) is some constant, Φ is the cumulative density function (CDF) of $\mathcal{N}(0,1)$, and $\alpha_r \in [0,1]$ is some parameter. With this choice, a least absolute shrinkage and selection operator (LASSO) estimator achieve a so-called near-oracle performance with a probability of at least $1-\alpha_r$ [53]. For further details, e.g., proof of convergence, interested readers may refer to the proofs given in [55–57,59], which also hold for the TST version presented here.

Putting all the above steps together, the final TST algorithm is listed in Algorithm 1. It is aborted either after a maximum number of iterations I, or if the relative improvement from iteration to iteration

$$d_i = \|\boldsymbol{s}_i - \boldsymbol{s}_{i-1}\|_2 / \|\boldsymbol{s}_{i-1}\|_2 \tag{34}$$

drops below a certain threshold ϵ. The particular values to determine ϵ in Algorithm 1 are taken from [63] and proven to work well in every case.

Algorithm 1 The TST algorithm.

Input: $\boldsymbol{A}$, $\boldsymbol{y}$, λ, I

Initialization:

$\epsilon \leftarrow \min\left(10^{-4}, 5 \cdot 10^{-3}\lambda\right)$, $i \leftarrow 0$, $d \leftarrow \infty$, $\boldsymbol{s}_0 \leftarrow \boldsymbol{0}$

$\mu \leftarrow \left(0.99/\|\boldsymbol{A}\|_2^2, n/(m\|\boldsymbol{A}\|_2^2)\right)$

Body:

while $d > \epsilon$ **and** $i < I$ **do**

$i \leftarrow i + 1$

$\boldsymbol{r}_i \leftarrow \boldsymbol{s}_{i-1} - \mu \nabla_{\boldsymbol{s}} h(\boldsymbol{s}_{i-1})$

$\boldsymbol{z}_i \leftarrow \mathcal{S}_{s,\mu\lambda}(\boldsymbol{r}_i)$

$\alpha_i \leftarrow \begin{cases} \frac{1}{n}\mathrm{div}\left(\mathcal{S}_{s,\mu\lambda}(\boldsymbol{r}_i)\right) & \text{if } \boldsymbol{s} \in \mathbb{R}^n \\ \frac{1}{2n}\mathrm{div}\left(\mathcal{S}_{s,\mu\lambda}(\boldsymbol{r}_i)\right) & \text{if } \boldsymbol{s} \in \mathbb{C}^n \end{cases}$

$c_i \leftarrow (\boldsymbol{z}_i - \alpha_i \boldsymbol{r}_i)^{\mathrm{H}} \boldsymbol{r}_i / \|\boldsymbol{z}_i - \alpha_i \boldsymbol{r}_i\|_2^2$

$\boldsymbol{s}_i \leftarrow c_i(\boldsymbol{z}_i - \alpha_i \boldsymbol{r}_i)$

$d \leftarrow \|\boldsymbol{s}_i - \boldsymbol{s}_{i-1}\|_2 / \|\boldsymbol{s}_{i-1}\|_2$

end while

Output: $\widehat{\boldsymbol{s}} \leftarrow \boldsymbol{r}_i$

In the following, simulation results are shown to illustrate the performance of TST. For all the following simulations presented, the SNR is defined as

$$\mathrm{SNR} = \left\|\boldsymbol{A}\widetilde{\boldsymbol{s}}\right\|_2^2 / \|\boldsymbol{n}\|_2^2 \tag{35}$$

and the recovery quality is measured as squared reconstruction error (SRE)

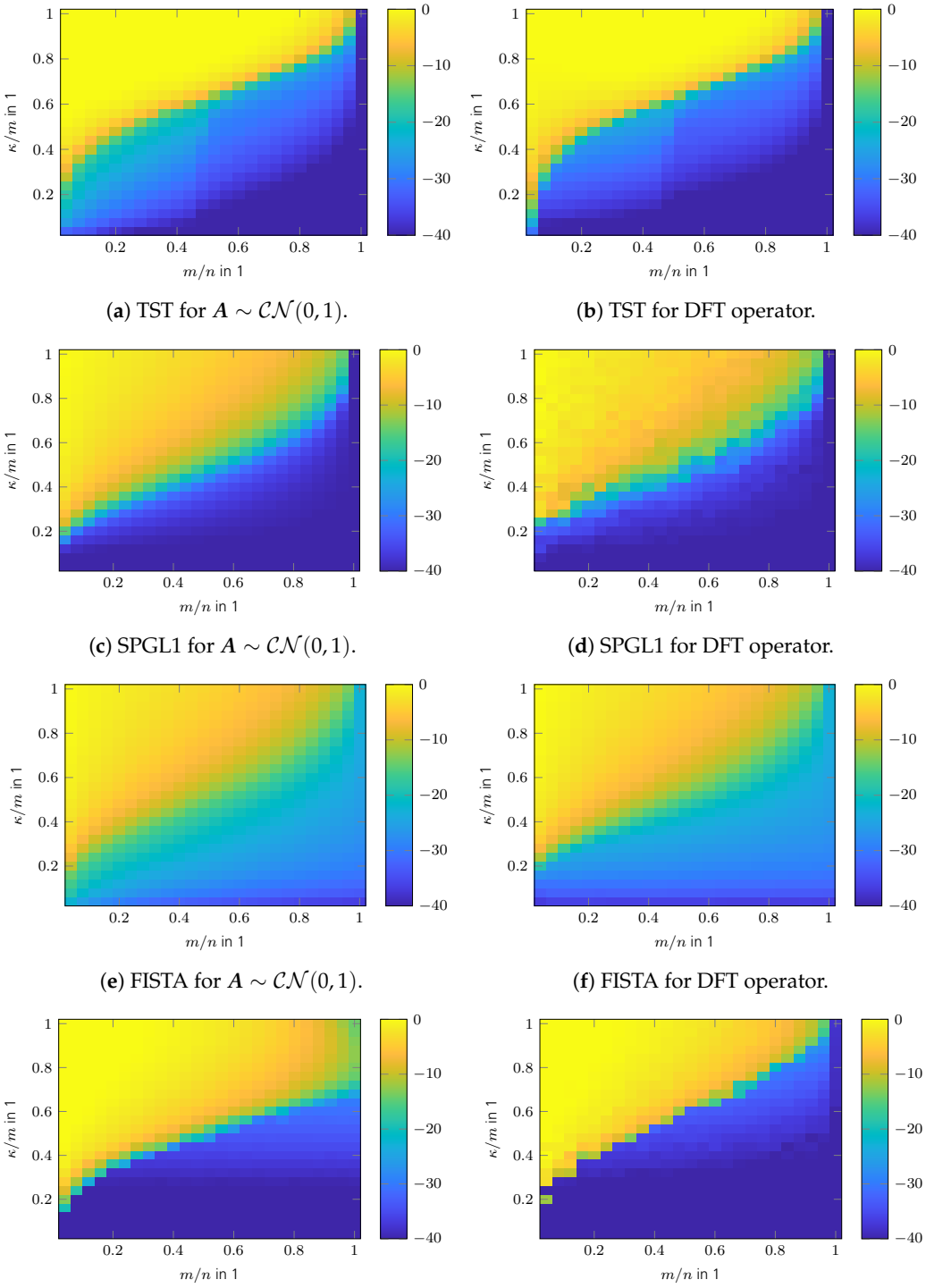
$$\mathrm{SRE} = \left\|\widetilde{\boldsymbol{s}} - \widehat{\boldsymbol{s}}\right\|_2^2 / \left\|\widetilde{\boldsymbol{s}}\right\|_2^2. \tag{36}$$

The reconstruction performance is evaluated by use of phase transition plots [66]. For a given type of sensing operator $\boldsymbol{A}$ (random or DFT), a number of sparse vectors $\widetilde{\boldsymbol{s}}$ are to be reconstructed for different measurement ratios $m/n \in (0,1]$ and sparsity ratios $\kappa/m \in (0,1]$. The resulting SRE is averaged over $N_{\mathrm{mc}} = 100$ Monte Carlo runs as $\frac{1}{N_{\mathrm{mc}}}\sum_{i=1}^{N_{\mathrm{mc}}} \mathrm{SRE}_i$ with SRE_i denoting the SRE of the ith reconstruction. In general, the higher the measurement rate m/n is, the easier it is to find a solution to (1). Likewise, the higher the sparsity ratio κ/m is, the more difficult the reconstruction problem becomes. Hence, the phase transition plot is rendered into a collection of reconstruction problems of varying difficulty. Various reconstruction algorithms now compete on how many problems of varying difficulty they can successfully reconstruct. Usually, a sharp transition of reconstructable from non-reconstructable problems arise for a given algorithm, hence the name phase transition diagram. For a single reconstruction problem, the sparse vector $\widetilde{\boldsymbol{s}}$ is set up by first deter-

mining κ and m from the given measurement and sparsity ratios. Then, κ support indices are drawn from an i. i. d. $\mathcal{U}(1, n)$ distribution. The entries at the determined indices are drawn from an i. i. d. complex Gaussian distribution. Next, the sensing operators $\boldsymbol{A}$ of size $m \times n$ are set up, where for random sensing operators $\boldsymbol{A} \sim \mathcal{CN}(0, 1)$ or for DFT operators $\boldsymbol{A}$ is set up from m randomly selected rows of an $n \times n$ DFT matrix. Finally, the elements of the noise vector $\boldsymbol{n}$ are drawn from an i. i. d. standard complex Gaussian distribution and scaled to the given SNR according to (35). For the simulation results shown in the following $n = 1250$, SNR $= 40\,\mathrm{dB}$, and $I = 300$ is used. It should be noted that random sensing matrices are of limited interest in radar engineering and serve here merely as a comparison benchmark, since many algorithms in the literature are stated for random sensing matrices alone. The DFT sensing operator servers as a representative for radar applications, since it often occurs there. Naturally, many more exist, a thorough evaluation of the TST algorithm for the CS applications mentioned in Section 1 is, however, beyond the scope of this text.

As comparison benchmarks, the aforementioned simulations are also conducted for SPGL1, FISTA, and NIHT. The SPGL1 algorithm is set up to solve the BPDN problem and as such is equipped with the true noise power P_n. Its implementation is taken from its official Github repository (https://github.com/mpf/spgl1, accessed on 20 December 2022, version v1.9). The maximum number of iterations is set to $I = 300$ iterations, and for all remaining parameters, the default setting is used. The FISTA algorithm is implemented according to [53] and its regularization parameter λ is set to (33). The NIHT algorithm is likewise implemented according to [47] and equipped once with the true number of sparse entries κ and once with twice the number of sparse entries 2κ. Both algorithms are aborted after a maximum number of $I = 300$ iterations or if the relative improvement d_i, as defined in (34), is below a threshold similar to the TST algorithm.

The phase transition diagrams of the TST algorithm for random and DFT sensing operators are shown in Figure 1a,b. Its overall reconstruction performance is higher than for SPGL1 shown in Figure 1c,d. Only for a very low sparsity ratio does SPGL1 achieve better performance. The reconstruction performance of FISTA is shown in Figure 1e,f. Its performance is inferior to that of the aforementioned algorithms. Figure 1g,h show the reconstruction performance of NIHT, where the parameter K was set to the true number of sparse entries κ. The result in the case of a wrongly chosen parameter $K = 2\kappa$ is shown in Figure 2. As can be seen, the reconstruction success depends heavily on K. Compared with FISTA and NIHT, TST exhibits better performance overall. A comparison of the convergence speed is shown in Figure 3, which shows the intermediate SREs. The convergence speed of NIHT in the case of $K = \kappa$ outperforms that of the TST algorithm; however, it drops significantly in the case of $K = 2\kappa$. A comparison of the computation time is shown in Figure 4 for a region where all algorithms perform equally well, except for FISTA. The simulations are conducted using Matlab® R2022b on an Ubuntu 20.04 LTS OS equipped with an Intel® Xeon(R) CPU E5-2687W v3 @ 3.10 GHz × 10. The fastest algorithm is TST, followed by FISTA and NIHT, provided $K = \kappa$ is used. The reason why NIHT is not the fastest algorithm, despite its superior convergence rate, is its costly line search procedure. The slowest algorithm is SPGL1; however, it should be noted that no effort in optimizing its performance was made. Finally, the SRE for various SNRs is shown in Figure 5 for all tested algorithms. As can be seen, the best SRE offers NIHT in the case of $K = \kappa$. For SNR $= 0\,\mathrm{dB}$, FISTA outperforms TST, but for higher SNR, however, TST achieves better performance. SPGL1 closely follows the available SNR level. In summary, TST is very easy to implement (in contrast to SPGL1), shows a state-of-the-art convergence rate, has low computational complexity as there are closed form solutions available for all required parameters, and finally does not generally require any unknown parameters such as NIHT. Hence, it is well suited for practical applications.



(**a**) TST for $\boldsymbol{A} \sim \mathcal{CN}(0,1)$.

(**b**) TST for DFT operator.

(**c**) SPGL1 for $\boldsymbol{A} \sim \mathcal{CN}(0,1)$.

(**d**) SPGL1 for DFT operator.

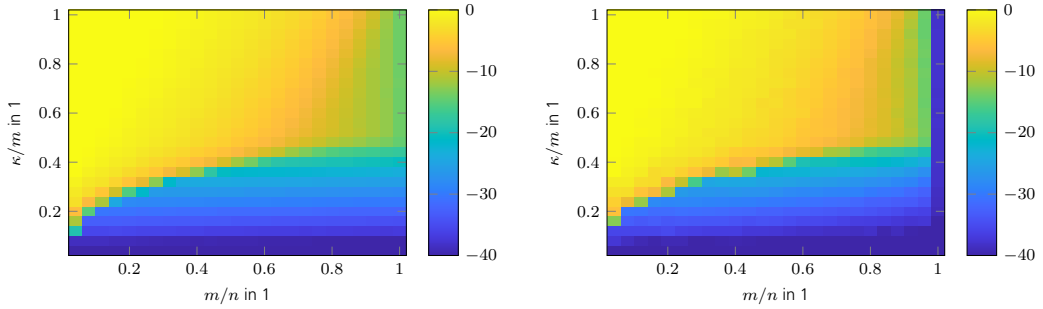
(**e**) FISTA for $\boldsymbol{A} \sim \mathcal{CN}(0,1)$.

(**f**) FISTA for DFT operator.

(**g**) NIHT for $\boldsymbol{A} \sim \mathcal{CN}(0,1)$ and $K = \kappa$.

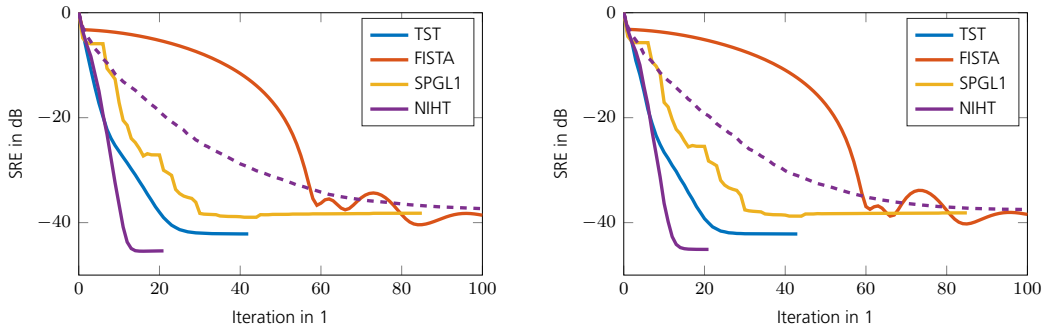
(**h**) NIHT for DFT operator and $K = \kappa$.

Figure 1. Phase transition of CS algorithms in SRE in dB.



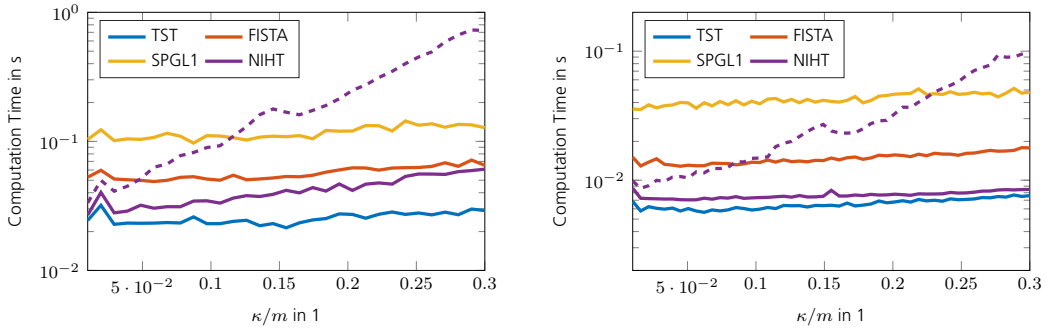
(**a**) NIHT for $\boldsymbol{A} \sim \mathcal{CN}(0,1)$. (**b**) NIHT for DFT operator.

Figure 2. Phase transition plot for NIHT in the case of wrongly chosen parameter $K = 2\kappa$.



(**a**) (**b**)

Figure 3. Comparison of convergence speed for $\kappa/m = 0.25$ and $m/n = 0.5$. The solid line shows NIHT for $K = \kappa$ and the dashed line $K = 2\kappa$. (**a**) Random sensing operators $\boldsymbol{A} \sim \mathcal{CN}(0,1)$. (**b**) DFT sensing operators with random rows.



(**a**) (**b**)

Figure 4. Comparison of computation time $m/n = 0.8$. The solid line shows NIHT for $K = \kappa$ and the dashed line $K = 2\kappa$. (**a**) Random sensing operators $\boldsymbol{A} \sim \mathcal{CN}(0,1)$. (**b**) DFT sensing operators with random rows.

The reconstruction performance of ℓ_1-relaxed minimization approaches can be further improved by the use of smoothed ℓ_0-techniques. A suitable and convenient version thereof is presented in the following section.

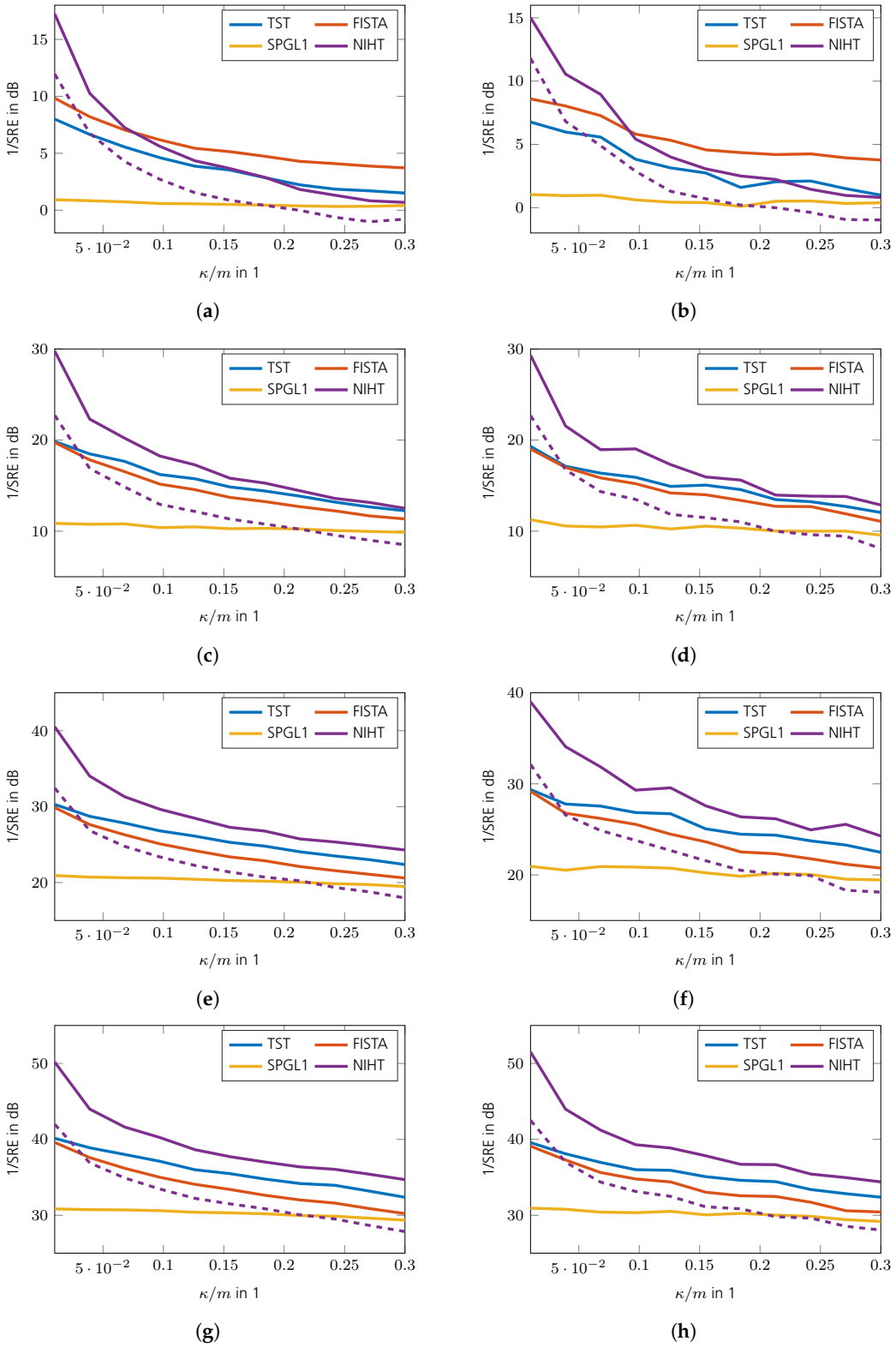


Figure 5. Comparison of SRE vs. SNR for $m/n = 0.8$. The solid line for NIHT shows $K = \kappa$ and the dashed line $K = 2\kappa$. (**a**) Random sensing operators $\boldsymbol{A} \sim \mathcal{CN}(0, 1)$ for SNR $= 0$ dB. (**b**) DFT sensing operator for SNR $= 0$ dB. (**c**) Random sensing operators $\boldsymbol{A} \sim \mathcal{CN}(0, 1)$ for SNR $= 10$ dB. (**d**) DFT sensing operator for SNR $= 10$ dB. (**e**) Random sensing operators $\boldsymbol{A} \sim \mathcal{CN}(0, 1)$ for SNR $= 20$ dB. (**f**) DFT sensing operator for SNR $= 20$ dB. (**g**) Random sensing operators $\boldsymbol{A} \sim \mathcal{CN}(0, 1)$ for SNR $= 30$ dB. (**h**) DFT sensing operator for SNR $= 30$ dB.

2.2. Complex Successive Concave Sparsity Approximation

The reconstruction error of ℓ_1-relaxed minimization approaches can be further reduced by smoothed ℓ_0-frameworks. The key idea is to approximate the ℓ_0-quasi-norm in the original objective Function (1) by a smooth function. The CSCSA algorithm does so, which is the author's extension of the SCSA algorithm from [63], capable of handling complex numbers. The CSCSA algorithm was published in [54] and is summarized in the following in brevity. Simulation results illustrating the performance of CSCSA in combination with the TST, SPGL1, and FISTA algorithms are presented in the following. Due to the structural similarity between CS and ARM problems, the idea of CSCSA can also be applied on ARM problems as presented in Section 3.2.

The idea of CSCSA is to substitute the ℓ_0-quasi-norm of $\boldsymbol{s} = [s_1 \cdots s_n]^{\mathrm{T}}$ by a more tractable approximation. In general, the ℓ_0-quasi-norm is defined as the number of nonzero elements in $\boldsymbol{s}$. Let

$$\delta(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{else} \end{cases} \tag{37}$$

be the Kronecker delta function, then the ℓ_0-quasi-norm of $\boldsymbol{s}$ can be defined as

$$\|\boldsymbol{s}\|_0 = \sum_{i=1}^{n} [1 - \delta(|s_i|)], \tag{38}$$

where $|s_i|$ denotes the magnitude of s_i. To make (38) smooth, it may be approximated by

$$1 - \delta(|s|) \approx f_\gamma(|s|) = 1 - \exp\left(-\frac{|s|}{\gamma}\right), \tag{39}$$

where γ determines how accurately the Kronecker function is approximated. An illustration of this approximation and other common approximations to the ℓ_0-quasi-norm are shown in Figure 6. As shown in [63], the series $\{f_\gamma(|s|)\}$ converges pointwise to $1 - \delta(|s|)$ as

$$\lim_{\gamma \to 0^+} f_\gamma(|s|) = \begin{cases} 0 & \text{if } |s| = 0 \\ 1 & \text{else.} \end{cases}$$

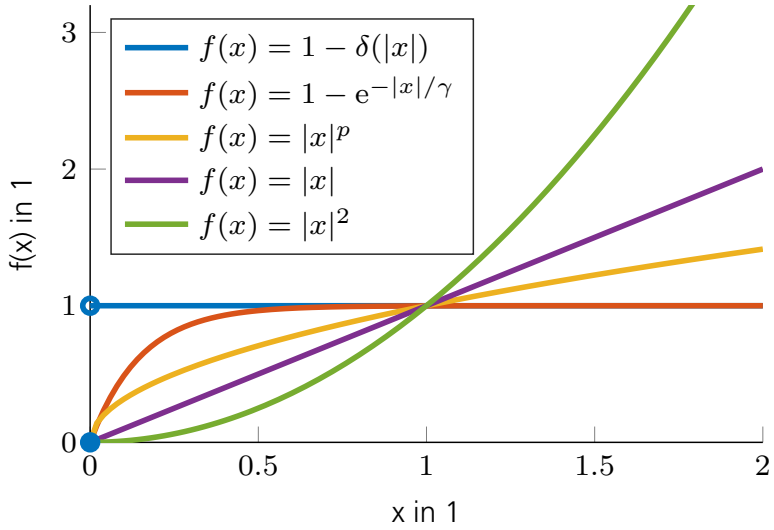


Figure 6. Common approximations to the ℓ_0-quasi-norm.

An approximation to the ℓ_0-quasi-norm is therefore

$$\|\boldsymbol{s}\|_0 \approx \sum_{i=1}^{n} f_\gamma(|s_i|) = F_\gamma(|\boldsymbol{s}|), \tag{40}$$

where $|\boldsymbol{s}|$ denotes a vector, which holds the magnitudes of the elements of $\boldsymbol{s}$. The optimization Problem (1) may now be relaxed to

$$\min_{\boldsymbol{s}} F_\gamma(|\boldsymbol{s}|) \text{ subject to } h(\boldsymbol{s}) \leq \epsilon^2, \tag{41}$$

where the residual term $h(\boldsymbol{s})$ was defined in (4) and $\epsilon^2 \geq \|\boldsymbol{n}\|_2^2$ is some constant noise energy. The constrained optimization Problem (41) can, for a fixed γ, be converted to an unconstrained optimization problem by use of regularization as

$$\widehat{\boldsymbol{s}} = \arg\min_{\boldsymbol{s}} \lambda_\gamma F_\gamma(|\boldsymbol{s}|) + h(\boldsymbol{s}). \tag{42}$$

At this point, it should be noted that the Program (42) is not convex anymore but, rather, a sum of a concave and a convex function. As such, (42) does not possess a unique minimum, and it is possible to get stuck in local minimum. To circumvent this problem, the graduated non-convexity (GNC) approach is used. The idea of GNC is to start the Program (42) with a solution $\widehat{\boldsymbol{s}}_0$ that is somewhat close to the true solution $\widetilde{\boldsymbol{s}}$ obtained from a convex algorithm, e.g., FISTA or TST. Then, (42) is minimized for a γ large enough such that the solution $\widehat{\boldsymbol{s}}$ is closer to $\widetilde{\boldsymbol{s}}$ yet does not get stuck in a wrong local minimum. Subsequently, γ is reduced by a constant factor $\gamma \leftarrow c\gamma$ to further approximate the ℓ_0-quasi-norm, and (42) is minimized using the solution from the previous iteration. The procedure is conducted until a stop criterion is met.

For a fixed γ, the optimization Problem (42) can be solved by use of the iterative thresholding (IT) approach, similar to ISTA (22), by iteratively conducting

$$\boldsymbol{s}_{i+1} = \mathcal{T}^{(\gamma)}_{\mu\lambda_\gamma}(\boldsymbol{s}_i - \mu\nabla_{\boldsymbol{s}} h(\boldsymbol{s}_i)), \tag{43}$$

where μ is some step size and

$$\mathcal{T}^{(\gamma)}_{\mu\lambda_\gamma}(x) = \begin{cases} 0 & |x| < \gamma\left(1 + \ln\left(\frac{2\lambda_\gamma\mu}{\gamma^2}\right)\right) \\ 0 & L'(0,|x|) < L'(|s_1|,|x|) \\ s_1 & \text{otherwise} \end{cases}, \tag{44}$$

is again an elementwise thresholding operator. Furthermore,

$$\begin{aligned} L'(p,q) &= \frac{1}{2\mu}(p-q)^2 + \lambda_\gamma f_\gamma(p) \\ s_1 &= \gamma W_0(z)\,\mathrm{sgn}(x) + x \\ z &= -\frac{2\lambda_\gamma\mu}{\gamma^2}\exp\left(-\frac{|x|}{\gamma}\right) \end{aligned}$$

and $W_0(\cdot)$ denotes the upper branch of the multi-valued Lambert W function [54]. It should be noted that $\mathcal{T}^{(\gamma)}_{\mu\lambda_\gamma}$ is a closed form solution of a subsequent minimization problem emerging from the IT approach. This closed form is possible only through the special choice of the ℓ_0-quasi-norm approximation (39), which leads to the possibility of applying the Lambert W function. For different choices, the subsequent minimization problem would require an additional minimization loop increasing the computational complexity. The regularization parameter is set to $\lambda_\gamma = 2\gamma\lambda$, where (33) is used for λ. Furthermore, the initial value γ_0 is set to $\gamma_0 = \max(|\boldsymbol{s}_0|)/10$, where $\boldsymbol{s}_0$ denotes the convex relaxed initial solution. More details regarding the CSCSA algorithm and the special selections of λ_γ and γ_0 are given in [54]. Finally, to further improve the convergence speed of the applied IT approach, a FISTA-like technique can be used as in [53], which does not increase the computational complexity but boosts the convergence rate from $\mathcal{O}(1/i)$ up to $\mathcal{O}(1/i^2)$. Putting all the above steps together, the final CSCSA algorithm is listed in Algorithm 2. It consists of two loops,

an inner and an outer one. In the outer loop γ is decreased gradually according to the GNC technique. The inner loop solves (42) by using a FISTA-like technique. The loops are aborted after a maximum number of iterations J and P or if the solution change, measured by the relative distance between consecutive solutions, drops below certain thresholds ϵ_o and ϵ_i. The threshold selections given in Algorithm 2 are taken from [63] and shown to work well in the following simulations.

Algorithm 2 The CSCSA algorithm.

Input: $\boldsymbol{A}$, $\boldsymbol{y}$, λ, I, J

Initialization:

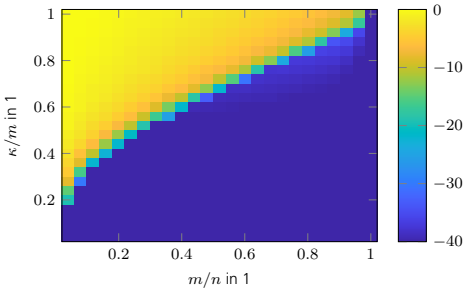
$c \leftarrow (0, 0.5)$, $\mu \leftarrow 0.99/\|\boldsymbol{A}\|_2^2$
$\epsilon_\mathrm{o} \leftarrow \min\left(10^{-4}, 5 \cdot 10^{-3}\lambda\right)$
$\epsilon_\mathrm{i} \leftarrow \min\left(10^{-3}, 5 \cdot 10^{-3}\lambda\right)$
$\widehat{\boldsymbol{s}} \leftarrow \mathrm{TST}(\boldsymbol{A}, \boldsymbol{y}, \lambda, \epsilon_\mathrm{i}, J)$, $\gamma \leftarrow \max(|\widehat{\boldsymbol{s}}|)/10$

Body:

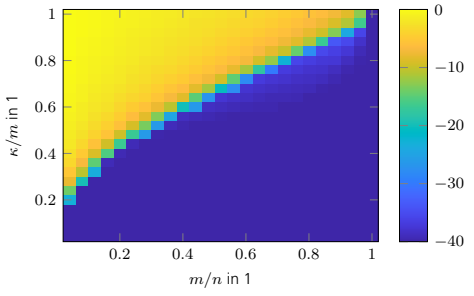
$i \leftarrow 0$, $d_\mathrm{o} \leftarrow \infty$
while $d_\mathrm{o} > \epsilon_\mathrm{o}$ **and** $i < I$ **do**
  $i \leftarrow i + 1$, $j \leftarrow 0$, $d_\mathrm{i} \leftarrow \infty$
  $t_1 \leftarrow 1$, $\boldsymbol{z}_1 \leftarrow \widehat{\boldsymbol{s}}$, $\boldsymbol{s}_0 \leftarrow \widehat{\boldsymbol{s}}$, $\lambda_\gamma \leftarrow 2\lambda\gamma$
  while $d_\mathrm{i} > \epsilon_\mathrm{i}$ **and** $j < J$ **do**
    $j \leftarrow j + 1$
    $\boldsymbol{s}_j \leftarrow \mathcal{T}_{\mu\lambda_\gamma}^{(\gamma)}\left(\boldsymbol{z}_j - \mu \boldsymbol{A}^\mathrm{H}\left(\boldsymbol{A}\boldsymbol{z}_j - \boldsymbol{y}\right)\right)$
    $t_{j+1} \leftarrow \left(1 + \sqrt{1 + 4t_j^2}\right)/2$
    $\boldsymbol{z}_{j+1} \leftarrow \boldsymbol{s}_j + \left(t_j - 1\right)\left(\boldsymbol{s}_j - \boldsymbol{s}_{j-1}\right)/t_{j+1}$
    $d_\mathrm{i} \leftarrow \left\|\boldsymbol{s}_j - \boldsymbol{s}_{j-1}\right\|_2 / \left\|\boldsymbol{s}_{j-1}\right\|_2$
  end while
  $d_\mathrm{o} \leftarrow \left\|\boldsymbol{s}_j - \widehat{\boldsymbol{s}}\right\|_2 / \|\widehat{\boldsymbol{s}}\|_2$
  $\widehat{\boldsymbol{s}} \leftarrow \boldsymbol{s}_j$
  $\gamma \leftarrow c\gamma$
end while

Output: $\widehat{\boldsymbol{s}}$

Simulation results of a CSCSA update onto FISTA, SPGL1, and TST for random and DFT sensing operators are shown in Figure 7. For λ, (33) was used and the decreasing factor for γ was set to $c = 0.1$. As can be seen, CSCSA significantly improves upon the convex results shown in Figure 1. A comparison of the reconstruction performances is shown in Figure 8, in which the curves indicate the 50 % success rate with respect to the Monte Carlo runs. Success is defined as SRE $\leq -$SNR. In addition, the 25 % and 75 % success rate confidence intervals are indicated as shaded areas. The CSCSA refinement algorithm basically obtains similar reconstruction performance for every initialization algorithm as shown in Figure 8.

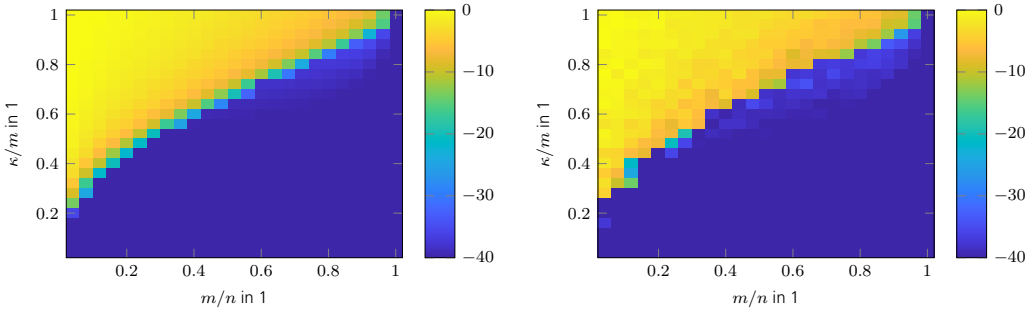


(**a**) FISTA + CSCSA for $\boldsymbol{A} \sim \mathcal{CN}(0, 1)$.

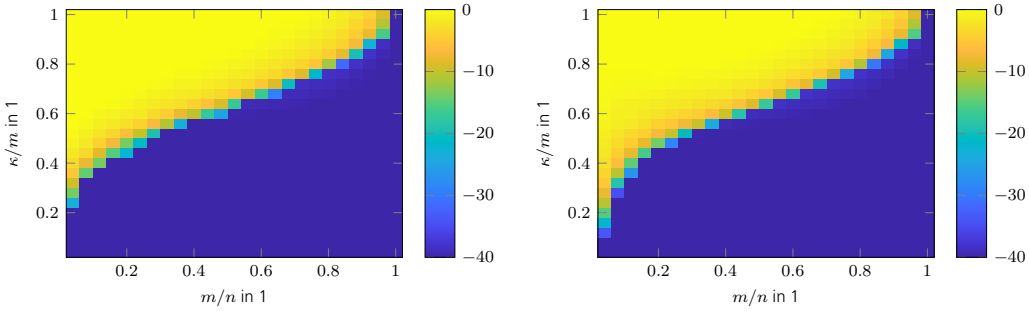


(**b**) FISTA + CSCSA for DFT operator.

Figure 7. *Cont.*

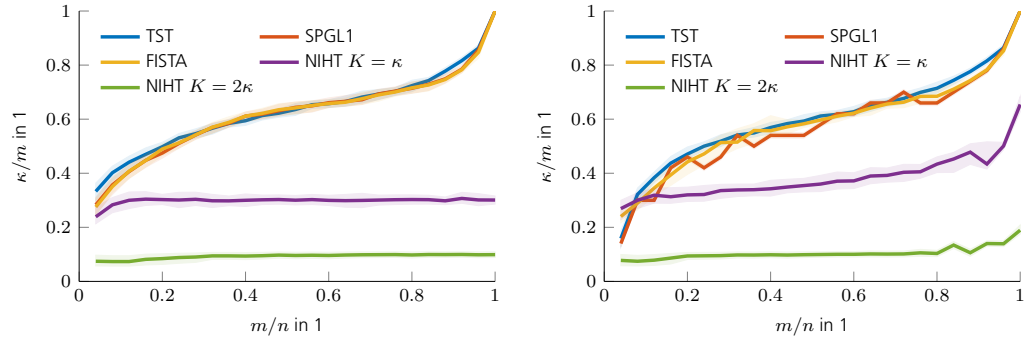


(**c**) SPGL1 + CSCSA for $\boldsymbol{A} \sim \mathcal{CN}(0,1)$. (**d**) SPGL1 + CSCSA for DFT operator.



(**e**) TST + CSCSA for $\boldsymbol{A} \sim \mathcal{CN}(0,1)$. (**f**) TST + CSCSA for DFT operator.

Figure 7. Phase transition of refined CS algorithms in SRE in dB.



(**a**) $\boldsymbol{A} \sim \mathcal{CN}(0,1)$ (**b**) Random row DFT sensing operator.

Figure 8. Comparison of reconstruction performances for CSCSA update. Reconstruction success is defined as SRE $\leq -$ SNR.

3. Affine Rank Minimization

In the following section, the TSVT and CSRA algorithms are introduced. Due to the structural similarity of CS and ARM algorithms, the TSVT and CSRA algorithms are also similar in structure to their CS counterparts from the former section. The TSVT algorithm obtains an initial low-rank convex solution, which is further refined by the CSRA algorithm. Their combination, likewise, yields all the desired properties of fast convergence and high reconstruction performance with no knowledge of generally unknown parameters, e.g., the true rank ρ of $\widetilde{\boldsymbol{L}}$ etc.

3.1. Turbo Singular Value Thresholding

The TSVT algorithm is inspired by turbo affine rank minimization (TARM) presented in [57], which is solved for ARM problems applying the turbo principle. This principle allows for a drastic improvement in convergence speed for ROIL sensing operators to which random and DFT sensing operators belong. Within TARM, a HT operator is applied.

Therefore, TARM was reworked here to use a ST operator. After finalizing TSVT, it came to the author's attention, that a similar approach was formerly published in [56] called singular value thresholding-turbo-compressive sensing (SVT-Turbo-CS), conducting the same reconstruction as TSVT. The contributions of this work beyond SVT-Turbo-CS are twofold: (1) expansion into the complex case and (2) providing an equation to select the regularization parameter λ.

The TSVT algorithm attempts to find a solution to (2) by solving the convex relaxed regularized optimization Problem (20). This is achieved using the iterative procedure

$$\begin{aligned} \boldsymbol{R}_i &= \boldsymbol{L}_{i-1} - \mu \nabla_{\boldsymbol{L}} h(\boldsymbol{L}_{i-1}) \\ \boldsymbol{Z}_i &= \mathcal{S}_{\mathrm{l},\mu\lambda}(\boldsymbol{R}_i) \\ \boldsymbol{L}_i &= c_i(\boldsymbol{Z}_i - \alpha_i \boldsymbol{R}_i) \end{aligned} \tag{45}$$

where μ is some step size, i is the iteration index, and $\mathcal{S}_{\mathrm{l},a}(\boldsymbol{X})$ is the ST operator given by (25). The required parameters $\{\mu, \alpha_i, c_i\}$ are chosen according to the turbo principle such that [57]

$$\left\langle \boldsymbol{R}_i - \widetilde{\boldsymbol{L}}, \boldsymbol{L}_{i-1} - \widetilde{\boldsymbol{L}} \right\rangle_{\mathrm{F}} = 0 \tag{46}$$

$$\left\langle \boldsymbol{R}_i - \widetilde{\boldsymbol{L}}, \boldsymbol{L}_{i} - \widetilde{\boldsymbol{L}} \right\rangle_{\mathrm{F}} = 0 \tag{47}$$

and that, furthermore, for a given $\boldsymbol{L}_{i-1}$, $\left\| \boldsymbol{L}_i - \widetilde{\boldsymbol{L}} \right\|_{\mathrm{F}}^2$ is minimized under (46) and (47). In the above equations, $\langle \boldsymbol{X}, \boldsymbol{Y} \rangle_{\mathrm{F}}$ denotes the Frobenius product. Equation (46) ensures that the input and output error of the gradient update step are uncorrelated. Equally, (47) ensures a decorrelation of the input and output error of the denoising step. This strategy allows for an improved convergence rate compared to classical gradient approaches. In order to fulfill (46) and (47), the true solution $\widetilde{\boldsymbol{L}}$ is required to determine exact values for $\{\mu, \alpha_i, c_i\}$. Since $\widetilde{\boldsymbol{L}}$ is unknown, approximate formulations for $\{\mu, \alpha_i, c_i\}$ were derived in [57] for the real valued case. In the general case these parameters are

$$\mu = \frac{n}{m \|\mathcal{A}\|_2^2} \tag{48}$$

$$\alpha_i = \begin{cases} \frac{1}{n} \mathrm{div}\left(\mathcal{S}_{l,\mu\lambda}(\boldsymbol{R}_i) \right) & \text{if } \boldsymbol{X} \in \mathbb{R}^{N_1 \times N_2} \\ \frac{1}{2n} \mathrm{div}\left(\mathcal{S}_{l,\mu\lambda}(\boldsymbol{R}_i) \right) & \text{if } \boldsymbol{X} \in \mathbb{C}^{N_1 \times N_2} \end{cases} \tag{49}$$

$$c_i = \frac{\langle \boldsymbol{Z}_i - \alpha_i \boldsymbol{R}_i, \boldsymbol{R}_i \rangle_{\mathrm{F}}}{\| \boldsymbol{Z}_i - \alpha_i \boldsymbol{R}_i \|_{\mathrm{F}}^2}. \tag{50}$$

For small scene sizes $N_1 \times N_2$, it was found from simulations that the step size μ required for convergence has to be reduced to $\mu < 1/\|\mathcal{A}\|_2^2$, which corresponds to the stable step size of the SVT algorithm. This phenomena was not discussed in [57,59], however, since the parameter Equations (48) to (50) are asymptotic approximations, and their validity requires the scene size to be sufficiently large. Nevertheless, TSVT still shows superior convergence speed even in the case of reduced step size. For the required divergence operator in (49), which is to be interpreted in the weak sense, i.e., it can fail to exist on negligible sets, a closed form solution exists [67] as

$$\mathrm{div}(\mathcal{S}_{l,a}(\boldsymbol{X})) = \sum_{j=1}^{n_{\min}} \left[\mathbb{I}(\sigma_j > a) + A_j \right] + 2B \tag{51}$$

for $\boldsymbol{X} \in \mathbb{R}^{N_1 \times N_2}$ or

$$\mathrm{div}(\mathcal{S}_{l,a}(\boldsymbol{X})) = \sum_{i=j}^{n_{\min}} \left[\mathbb{I}(\sigma_j > a) + A_j \right] + 4B \tag{52}$$

for $\boldsymbol{X} \in \mathbb{C}^{N_1 \times N_2}$, when $\boldsymbol{X}$ is simple, i.e., $\boldsymbol{X}$ has no repeated singular values, and 0 otherwise. In (51) and (52), $\mathbb{I}(\cdot)$ denotes the indicator function and

$$A_j = \begin{cases} |N_1 - N_2| \left(1 - \frac{a}{\sigma_i}\right)_+ & \text{if } \boldsymbol{X} \in \mathbb{R}^{N_1 \times N_2} \\ (2|N_1 - N_2| + 1)\left(1 - \frac{a}{\sigma_i}\right)_+ & \text{if } \boldsymbol{X} \in \mathbb{C}^{N_1 \times N_2} \end{cases}$$

$$B = \sum_{i \neq j, i, j = 1}^{n_{\min}} \frac{\sigma_i (\sigma_i - a)_+}{\sigma_i^2 - \sigma_j^2}.$$

For more details on the derivation of $\{\mu, \alpha_i, c_i\}$, the reader is referred to [57]. The next question of course is how to choose the regularization parameter λ and thus which shrinkage $a = \mu\lambda$ should be applied. The usage of a ST operator causes the singular values of the optimal solution $\boldsymbol{L}_{\text{opt}}$ generated by the turbo framework (45) to possess an offset of a for every singular value compared with the true solution $\widetilde{\boldsymbol{L}}$. Too much shrinkage results in a large bias, while too little results in a slow convergence rate. As such, an optimal constant λ does not exists, rather λ would need to be adjusted in every iteration. Unfortunately, to the best of the author's knowledge, a closed form solution to determine such a λ is known to exist only for the signal model $\boldsymbol{X} = \widetilde{\boldsymbol{L}} + \boldsymbol{E}$ with the entries of $\boldsymbol{E}$ being i. i. d. normally distributed [67]. However, this does not apply here, since the gradient update step as input to $\mathcal{S}_{\boldsymbol{L},a}(\cdot)$ is not of such form. We therefore follow another approach and set the regularization parameter λ in a similar manner to (33) as

$$\lambda = {}^1\!/_4 c_{\text{r}} \Phi^{-1}\left(1 - \frac{\alpha_{\text{r}}}{2n}\right) \sqrt{P_{\text{n}} \max(N_1, N_2)}, \tag{53}$$

where $c_{\text{r}} > 1$ is some constant, Φ is the CDF of $\mathcal{N}(0,1)$, and $\alpha_r \in [0,1]$ is some parameter. The additional factor of $\sqrt{\max(N_1, N_2)}$ adjusts the threshold from (33) for singular values. For proofs of convergence, unique solutions, etc., of TARM and thus TSVT, the interested reader may have a look into [57].

Putting all the above steps together, the final TSVT algorithm is listed in Algorithm 3. It is aborted if an upper limit of iterations I is reached or if the relative change of the intermediate solutions

$$d_i = \|\boldsymbol{L}_i - \boldsymbol{L}_{i-1}\|_{\text{F}} / \|\boldsymbol{L}_{i-1}\|_{\text{F}} \tag{54}$$

is below a predefined threshold ϵ. The threshold selection given in Algorithm 3 was evaluated numerically and proven to work well in every case.

In the following, simulation results are shown to illustrate the performance of TSVT. For all the following ARM algorithms presented, the SNR is defined as

$$\text{SNR} = \|\mathcal{A}(\widetilde{\boldsymbol{L}})\|_2^2 / \|\boldsymbol{n}\|_2^2 \tag{55}$$

and the recovery success is measured as SRE

$$\text{SRE} = \|\widetilde{\boldsymbol{L}} - \widehat{\boldsymbol{L}}\|_{\text{F}}^2 / \|\widetilde{\boldsymbol{L}}\|_{\text{F}}^2. \tag{56}$$

The reconstruction performance is evaluated by use of phase transition plots, where for a given type of sensing operator $\mathcal{A}$ (random or DFT) a number of low-rank matrices $\widetilde{\boldsymbol{L}}$ are to be reconstructed for different measurement ratios ${}^m\!/_n \in (0,1]$ and degree of freedom ratios ${}^{d_\rho}\!/_m \in (0,1]$, where $d_\rho = \rho(N_1 + N_2 - \rho)$ denotes the number of degrees of freedom in a rank-ρ matrix. The low-rank matrices $\widetilde{\boldsymbol{L}}$ are set up by first determining ρ and m from the given ratios. Here, it should be mentioned that since the rank ρ has to be a whole number, it is not possible to clearly find a suitable integer ρ for every possible d_ρ. Instead, ρ is determined by the closest integer to give the desired d_ρ. Obviously, this only approximates the desired d_ρ, where the approximation becomes better for larger dimensions $N_1 \times N_2$. This, however, results in very high computation time. With this

approach, the resulting ρ may also result to zero, especially for small d_ρ. In this case, no simulations were conducted, which are indicated by blank white entries in the following phase transition plots. A true low-rank matrix is set up as $\widetilde{\boldsymbol{L}} = \boldsymbol{X}_r \boldsymbol{X}_l^{\mathrm{H}}$, where $\boldsymbol{X}_r \in \mathbb{C}^{N_1 \times \rho}$ and $\boldsymbol{X}_l \in \mathbb{C}^{N_2 \times \rho}$ are two orthonormalized matrices with elements drawn from an i. i. d. complex Gaussian distribution. Next, the sensing operators $\mathcal{A}$ are set up, where for random sensing operators $\mathcal{A} \sim \mathcal{CN}(0,1)$ or for DFT operators $\mathcal{A}$ is set up from m randomly selected rows of an $N_1 N_2 \times N_1 N_2$ DFT matrix following the identity $\boldsymbol{y} = \mathcal{A}(\boldsymbol{L}) = \boldsymbol{A}\,\mathrm{vec}(\boldsymbol{L})$. Finally, the elements of the noise vector $\boldsymbol{n}$ are drawn from an i. i. d. standard complex Gaussian distribution and scaled the given SNR according to (55). For every combination of measurement and rank ratio, 100 Monte Carlo runs were conducted and averaged to generate the phase transition plot. For the simulation results shown in the following $N_1 = 80$, $N_2 = 24$, SNR $= 40$ dB, and $I = 300$ is used.

Algorithm 3 The TSVT algorithm.

Input: $\mathcal{A}$, $\boldsymbol{y}$, λ, I

Initialization:

$\epsilon \leftarrow \min\left(10^{-4}, 5 \cdot 10^{-3}\lambda\right)$, $i \leftarrow 0$, $d \leftarrow \infty$, $\boldsymbol{L}_0 \leftarrow \boldsymbol{0}$

$\mu \leftarrow \left(0.99/\|\mathcal{A}\|_2^2, n/(m\|\mathcal{A}\|_2^2)\right)$

Body:

while $d > \epsilon$ **and** $i < I$ **do**

$i \leftarrow i + 1$

$\boldsymbol{R}_i \leftarrow \boldsymbol{L}_{i-1} - \mu \nabla_{\boldsymbol{L}} h(\boldsymbol{L}_{i-1})$

$\boldsymbol{Z}_i \leftarrow \mathcal{S}_{\mathrm{l},\mu\lambda}(\boldsymbol{R}_i)$

$$\alpha_i \leftarrow \begin{cases} \frac{1}{n}\mathrm{div}\left(\mathcal{S}_{l,\mu\lambda}(\boldsymbol{R}_i)\right) & \text{if } \boldsymbol{X} \in \mathbb{R}^{N_1 \times N_2} \\ \frac{1}{2n}\mathrm{div}\left(\mathcal{S}_{l,\mu\lambda}(\boldsymbol{R}_i)\right) & \text{if } \boldsymbol{X} \in \mathbb{C}^{N_1 \times N_2} \end{cases}$$

$c_i \leftarrow \langle \boldsymbol{Z}_i - \alpha_i \boldsymbol{R}_i, \boldsymbol{R}_i \rangle_{\mathrm{F}} / \|\boldsymbol{Z}_i - \alpha_i \boldsymbol{R}_i\|_{\mathrm{F}}^2$

$\boldsymbol{L}_i \leftarrow c_i(\boldsymbol{Z}_i - \alpha_i \boldsymbol{R}_i)$

$d \leftarrow \|\boldsymbol{L}_i - \boldsymbol{L}_{i-1}\|_{\mathrm{F}} / \|\boldsymbol{L}_{i-1}\|_{\mathrm{F}}$

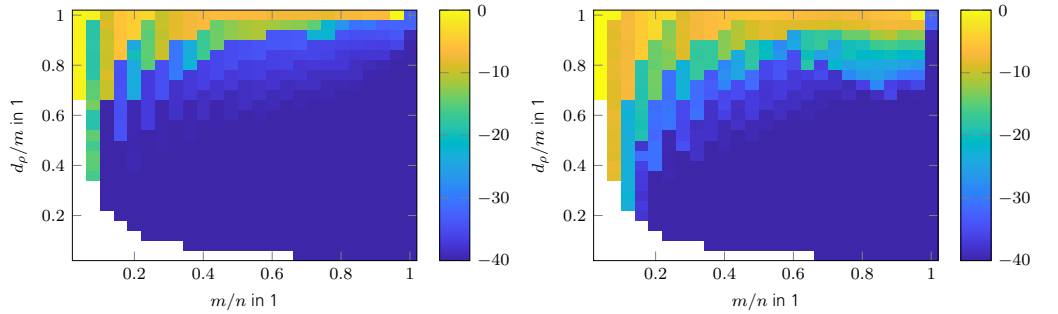
end while

Output: $\widehat{\boldsymbol{L}} \leftarrow \boldsymbol{R}_i$

As comparison benchmarks, the aforementioned simulations are also conducted for SVT, SVP, and TARM. The SVT algorithm is implemented according to [31], and its regularization parameter λ is set to (53). The SVP and TARM algorithms are likewise implemented according to [48,57] and equipped once with the true rank ρ and once with 2ρ. Both algorithms are aborted after a maximum number of $I = 300$ iterations or if the relative improvement d_{i} as defined in (54) is below a threshold similar to the TSVT algorithm.

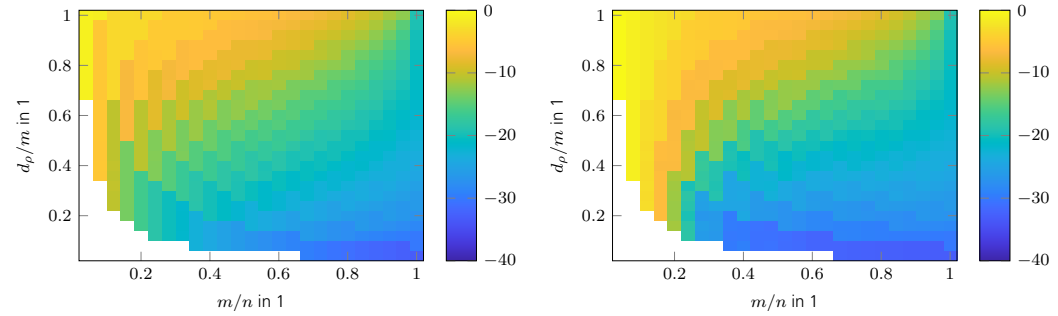
The phase transition diagrams of the TSVT algorithm for random and DFT sensing operators are shown in Figure 9a,b. Its reconstruction performance is higher than SVT and SVP as shown in Figure 9c–f. The reconstruction performance of TARM in the case of the correctly chosen parameter $R = \rho$ is comparable to TSVT, as can be seen in Figure 9g,h. Interestingly, the DFT sensing operator reveals unfavorable properties for TARM in the upper right part of the phase transition plot shown in Figure 9h. This effect is also visible for TSVT as shown in Figure 9b, though less pronounced. The reason therefore is the rather structured nature of the DFT sensing operator, especially in the case of only a few randomly removed rows for a high measurement ratio m/n. In [56], a similar effect is shown, but for CS and for correlated sparse signals. To fix this issue, the sensing operator was "improved" in [56] by adding more "randomness" to the DFT transformation by applying random signs to the columns of the glsdft matrix. In [57], such an augmented DFT matrix was also directly chosen. For radar, however, this approach is usually not possible, since the Fourier transform is inherently contained in the signal model. The reconstruction performance of TARM in the case of wrongly chosen parameter $R = 2\rho$ is shown in Figure 10. As can be seen, it depends heavily on R. The same holds for the convergence speed shown in

Figure 11, which shows the intermediate SREs. The convergence speed of SVT is inferior and merely servers as a comparison benchmark. The convergence speed of SVP and TARM depend on the parameter R, where TARM outperforms SVP in any case. The convergence speed of TARM in the case of $R = \rho$ outperforms the TSVT algorithm, although it drops significantly in the case of $R = 2\rho$. Compared with SVP, TSVT shows a faster or comparable convergence speed, depending on the sensing operator. For DFT sensing operators, the TSVT algorithm shows slower convergence speed than TARM and SVP. This is again due to the structured nature of the DFT sensing operator. While TARM and SVP can leverage the knowledge of $R = \rho$ to achieve fast convergence, TSVT requires more iterations. However, TARM and SVP are very sensitive to the selected value of R as illustrated by the dashed lines in Figure 11. A comparison of the computation time is shown in Figure 12 for a region where all algorithms perform equally well, except for SVT. The system specifications are the same as given in Section 2. The fastest algorithm is TARM provided $R = \rho$ is used. The second fastest is TSVT, followed by SVT and SVP. The computation time of TARM and SVP obviously increase in the case of $R > \rho$, rendering TSVT a good choice in the case ρ is unknown. Finally, the SRE for various SNRs is shown in Figure 13 for all tested algorithms. As can be seen, the best SREs offer TARM and SVP provided $R = \rho$. Interestingly, the higher the available SNR, the worse is the achievable SRE in the case of $R = 2\rho$. The TSVT algorithm follows the SRE of SVP and NIHT with an average loss of 2 dB. The SVT algorithm provides a comparable SRE only for SNR $= 0$ dB. In summary, TSVT is very easy to implement, shows a state-of-the-art convergence rate, has low computational complexity as there are closed form solutions available for all required parameters, and finally it does not require any unknown parameters, e.g., as the true rank ρ of $\widetilde{\boldsymbol{L}}$, in general. Hence, it is well suited for practical applications.



(**a**) TSVT for $\mathcal{A} \sim \mathcal{CN}(0,1)$.

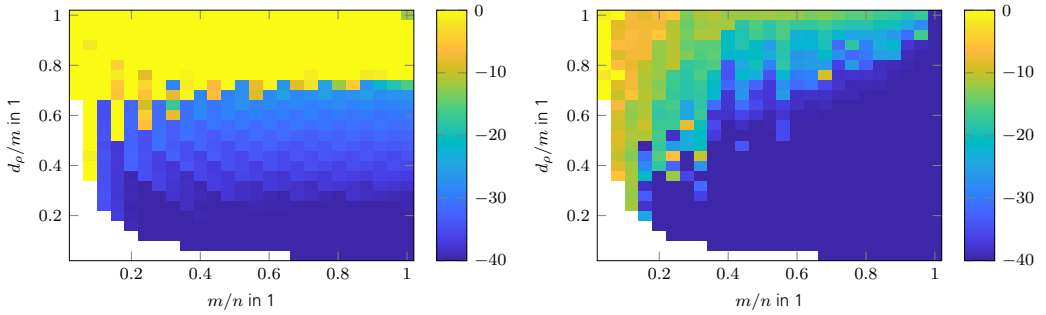
(**b**) TSVT for DFT operator.



(**c**) SVT for $\mathcal{A} \sim \mathcal{CN}(0,1)$.

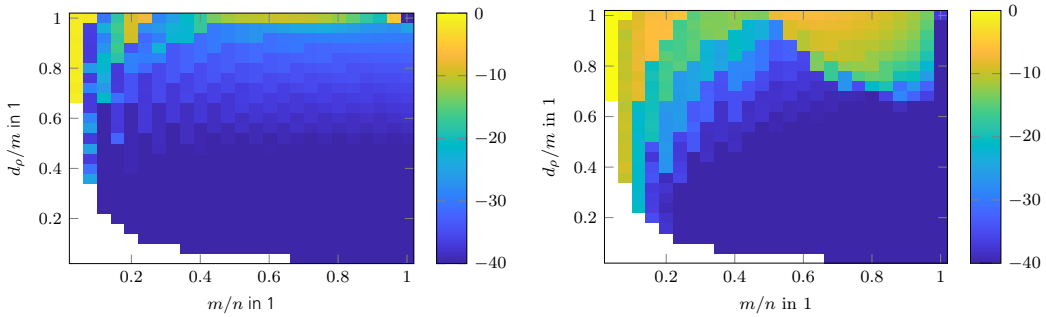
(**d**) SVT for DFT operator.

Figure 9. *Cont.*



(**e**) SVP for $\mathcal{A} \sim \mathcal{CN}(0,1)$ and $R = \rho$.

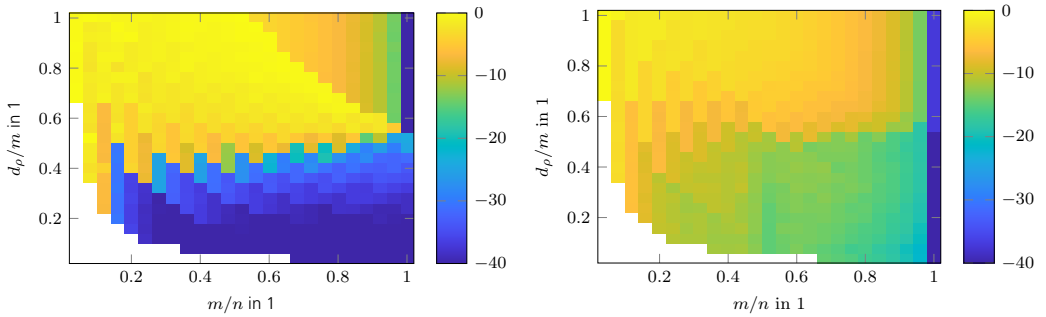
(**f**) SVP for DFT operator and $R = \rho$.



(**g**) TARM for $\mathcal{A} \sim \mathcal{CN}(0,1)$ and $R = \rho$.

(**h**) TARM for DFT operator and $R = \rho$.

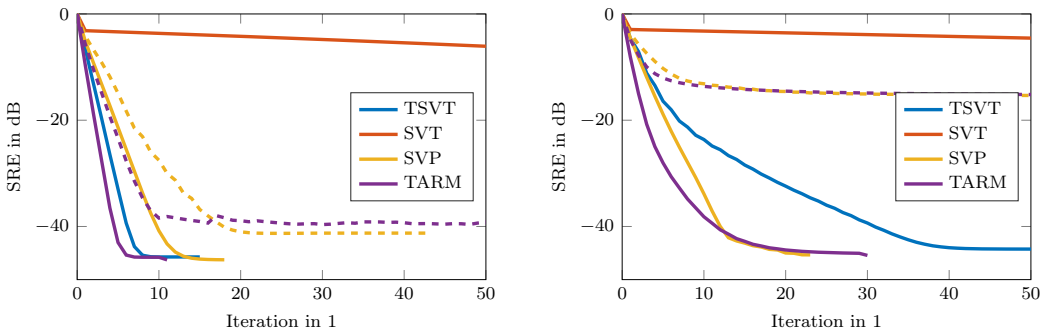
Figure 9. Phase transition of ARM algorithms in SRE in dB.



(**a**) TARM for $A \sim \mathcal{CN}(0,1)$.

(**b**) TARM for DFT operator.

Figure 10. Phase transition plot for TARM in the case of wrongly chosen parameter $R = 2\rho$.



(**a**)

(**b**)

Figure 11. Comparison of convergence speed for $d_\rho/m = 0.25$ and $m/n = 0.5$. The solid lines show SVP and TARM for $R = \rho$ and the dashed lines for $R = 2\rho$. (**a**) Random sensing operators $A \sim \mathcal{CN}(0,1)$. (**b**) DFT sensing operators with random rows.

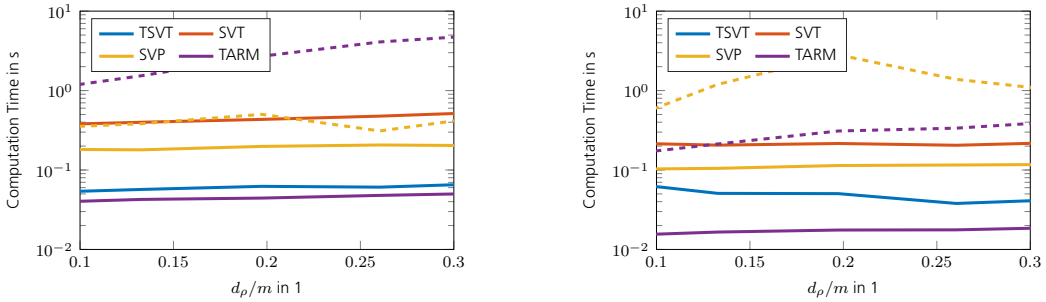


Figure 12. Comparison of computation time for $m/n = 0.8$ in an interval where all algorithms perform equally well. The solid lines show SVP and TARM for $R = \rho$ and the dashed lines for $R = 2\rho$. (**a**) Random sensing operators $A \sim \mathcal{CN}(0, 1)$. (**b**) DFT sensing operators with random rows.

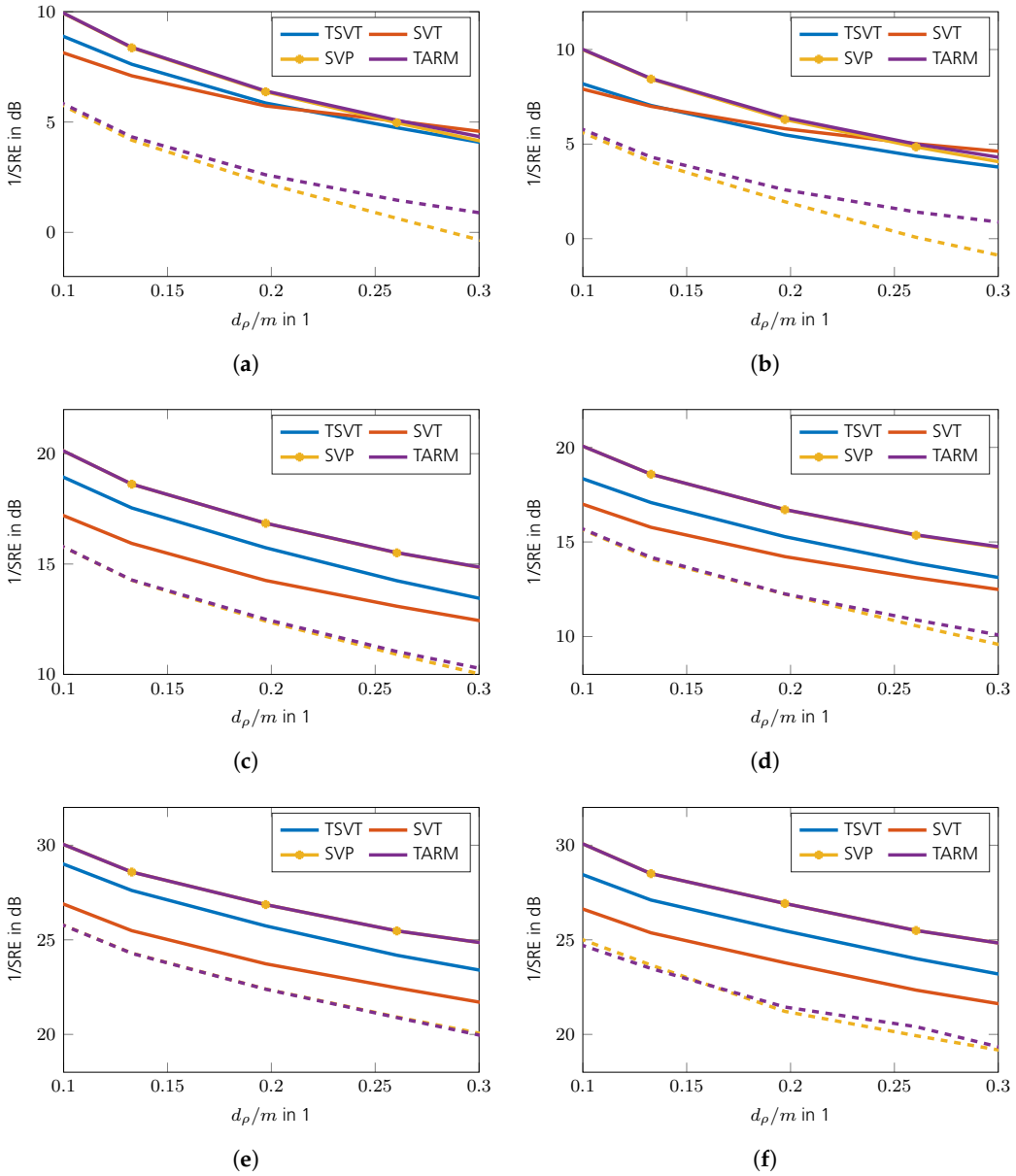
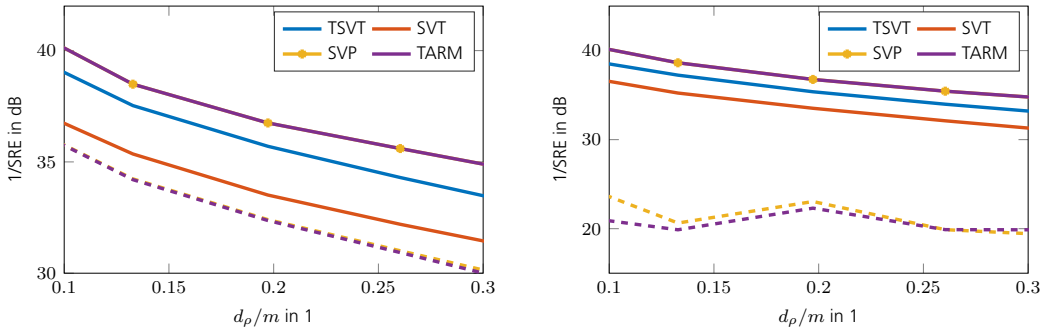


Figure 13. *Cont.*



(**g**) (**h**)

Figure 13. Comparison of SRE vs. SNR for $m/n = 0.8$. The solid line for SVP and TARM show $R = \rho$ and the dashed line $R = 2\rho$. (**a**) Random sensing operators $\boldsymbol{A} \sim \mathcal{CN}(0,1)$ for SNR $= 0$ dB. (**b**) DFT sensing operator for SNR $= 0$ dB. (**c**) Random sensing operators $\boldsymbol{A} \sim \mathcal{CN}(0,1)$ for SNR $= 10$ dB. (**d**) DFT sensing operator for SNR $= 10$ dB. (**e**) Random sensing operators $\boldsymbol{A} \sim \mathcal{CN}(0,1)$ for SNR $= 20$ dB. (**f**) DFT sensing operator for SNR $= 20$ dB. (**g**) Random sensing operators $\boldsymbol{A} \sim \mathcal{CN}(0,1)$ for SNR $= 30$ dB. (**h**) DFT sensing operator for SNR $= 30$ dB.

The reconstruction performance of the nuclear norm relaxed approaches can be further improved by use of smoothed rank techniques. A suitable and convenient version thereof is presented in the following section.

3.2. Complex Smoothed Rank Approximation

Although the performance of TSVT is already good, it can be improved in a similar manner as CSCSA does for TST results. The CSRA algorithm does so by enforcing a stricter rank measure than the nuclear norm used in (20). The CSRA algorithm is our extension based on the smoothed rank function (SRF) algorithm [61] and the SRA approach in [64]. CSRA is applicable to complex valued problems and in contrast to [64], has a closed form solution to a subsequent optimization problem, hence, reduced computational complexity. In the following, an overview of CSRA is given, while additional details can be found in the Appendix B.

To enforce a stricter rank measure, a different replacement for the rank function is proposed. The rank of $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{H}}$, where $\boldsymbol{\Sigma} = \operatorname{diag}(\boldsymbol{\sigma}(\boldsymbol{L}))$, is defined as the number of nonzero elements in $\boldsymbol{\sigma}$, where the vector $\boldsymbol{\sigma}(\boldsymbol{L}) = \begin{bmatrix}\sigma_1 & \cdots & \sigma_{n_{\min}}\end{bmatrix}^{\mathrm{T}}$ holds the singular values of $\boldsymbol{L}$. The rank of $\boldsymbol{L}$ can thus be defined as

$$\operatorname{rank}(\boldsymbol{L}) = \sum_{i=1}^{n_{\min}} [1 - \delta(\sigma_i(\boldsymbol{L}))], \tag{57}$$

where $\sigma_i(\boldsymbol{L})$ is the ith largest singular value of $\boldsymbol{L}$ and $\delta(x)$ is the Kronecker delta function. Similar to CSCSA, (57) can be relaxed as

$$f_\gamma(x) = 1 - \delta(x) \approx 1 - \exp\left(-\frac{|x|}{\gamma}\right). \tag{58}$$

As can be seen, γ determines how close the rank function is approximated. Thus, an approximation to the rank function can be define as

$$\operatorname{rank}(\boldsymbol{L}) \approx \sum_{i=1}^{n_{\min}} f_\gamma(\sigma_i(\boldsymbol{L})) = F'_\gamma(\boldsymbol{\sigma}(\boldsymbol{L})) = F_\gamma(\boldsymbol{L}). \tag{59}$$

The optimization problem (2) may now be relaxed to

$$\min_{\boldsymbol{L}} F_\gamma(\boldsymbol{L}) \text{ subject to } h(\boldsymbol{L}) \leq \epsilon^2, \tag{60}$$

where the data fidelity term $h(\boldsymbol{L})$ was defined in (5). The constrained optimization problem (60) can be converted to an unconstrained one by use of regularization, which yields

$$\min_{\boldsymbol{L}} \lambda_\gamma F_\gamma(\boldsymbol{L}) + h(\boldsymbol{L}), \tag{61}$$

where λ again is some regularization parameter. The minimization in (61) constitutes an alternative to the original ℓ_1 problem given by (20). In this approach, $F_\gamma(\boldsymbol{L})$ is not concave nor convex (since $f_\gamma(x)$ is defined also for negative numbers due to evidential requirements) and not smooth, i.e., not differentiable at the origin. In order to avoid getting stuck in local minimum, the GNC approach is again applied similar to CSCSA. At first an initial solution $\boldsymbol{L}_0$ is obtained from a convex optimization algorithm such as TSVT and γ is chosen big enough such that (62) does not get stuck in a local minimum. After convergence, γ is subsequently reduced until a stopping criterion is met. For a fixed γ, the optimization Problem (61) can be solved by the IT method, by iteratively solving

$$\begin{aligned} \boldsymbol{L}_{0i} &= \boldsymbol{L}_i - \mu \nabla_{\boldsymbol{L}} h(\boldsymbol{L}_i) \\ \boldsymbol{L}_{i+1} &= \boldsymbol{U}_{0i} \operatorname{diag}\left(\mathcal{T}^{(\gamma)}_{\mu\lambda_\gamma}(\boldsymbol{\sigma}_{0i})\right) \boldsymbol{V}_{0i}^{\mathrm{H}} \end{aligned} \tag{62}$$

where $\boldsymbol{L}_{0i} = \boldsymbol{U}_{0i} \operatorname{diag}(\boldsymbol{\sigma}_{0i}) \boldsymbol{V}_{0i}^{\mathrm{H}}$ is the SVD of the output of the gradient update step $\boldsymbol{L}_{0i}$ and $\mathcal{T}^{(\gamma)}_{\mu\lambda_\gamma}$ is again a thresholding operator, which is given by (A25). In contrast to [64], a difference of convex (D.C.) optimization strategy to solve (61) is not needed; rather, the closed form solution $\mathcal{T}^{(\gamma)}_{\mu\lambda_\gamma}(\cdot)$ is available. The regularization parameter is set to $\lambda_\gamma = 16\gamma\lambda$, where (53) is used for λ. This value was found from extensive numerical simulations to nicely balance the data fidelity error $h(\boldsymbol{L})$ and the approximated rank function $F_\gamma(\boldsymbol{L})$ and perform well in any case. Furthermore, the initial value γ_0 is set to $\gamma_0 = \left\|\boldsymbol{L}_0\right\|_2 / 10$, where $\boldsymbol{L}_0$ denotes the convex relaxed initial solution. More details regarding the special selection of γ_0 are given in the Appendix B. Finally, the convergence rate of the CSRA algorithm is accelerated using a FISTA-like technique [53]. Putting all the above steps together, the final CSRA algorithm is listed in Algorithm 4. The algorithm consists of two loops, an inner and an outer one. In the outer loop γ is decreased gradually according to the GNC technique. The inner loop solves (62) by using a FISTA-like technique. The loops are aborted after a maximum number of iterations J and P or if the solution changes, measured by the relative distance between consecutive solutions, drop below certain thresholds ϵ_o and ϵ_i. The threshold selections given in Algorithm 4 were found numerically as in [63] and were proven to work well in every case.

Simulation results for random and DFT sensing operators are shown in Figure 14. The decreasing factor for γ was set to $c = 0.5$. As can be seen, CSRA improves upon the the convex results shown in Figure 9. If SVT was used as initialization algorithm for CSRA, the additional gain is dramatic for the random as well as the DFT sensing operator. If TSVT was used as an initialization algorithm, the improvements are not as high, because TSVT already achieves high reconstruction performance especially for random sensing operators. Nevertheless, especially for DFT sensing operators, CSRA helps to boost the reconstruction performance.

A comparison of the reconstruction performance of TSVT + CSRA to TARM is shown in Figure 15, in which the curves indicate a 50% success rate with respect to the Monte Carlo runs. Success is defined twofold as either SRE $\leq -$ SNR for a strict success definition and SRE $\leq -(\mathrm{SNR} - 5\,\mathrm{dB})$ for a less strict success definition. In addition, the 25 % and 75 % success rate confidence intervals are indicated as shaded areas. As can be seen, the combination of TSVT + CSRA follows and even outperforms TARM despite its non-awareness of the true rank ρ.

Algorithm 4 The CSRA algorithm.

Input: $\mathcal{A}$, $\boldsymbol{y}$, λ, J, P

Initialization:

$c \leftarrow (0, 0.5)$, $\mu \leftarrow 0.99/\|\mathcal{A}\|_2^2$

$\epsilon_{\mathrm{o}} \leftarrow \min\left(10^{-4}, 5 \cdot 10^{-3}\lambda\right)$

$\epsilon_{\mathrm{i}} \leftarrow \min\left(10^{-3}, 5 \cdot 10^{-3}\lambda\right)$

$\widehat{\boldsymbol{L}} \leftarrow \mathrm{TSVT}(\mathcal{A}, \boldsymbol{y}, \lambda, \epsilon_{\mathrm{i}}, J)$, $\gamma \leftarrow \left\|\widehat{\boldsymbol{L}}\right\|_2/10$

Body:

$p \leftarrow 0$, $d_{\mathrm{o}} \leftarrow \infty$

while $d_{\mathrm{o}} > \epsilon_{\mathrm{o}}$ **and** $p < P$ **do**

  $p \leftarrow p + 1$, $j \leftarrow 0$, $d_{\mathrm{i}} \leftarrow \infty$

  $t_1 \leftarrow 1$, $\boldsymbol{Z}_1 \leftarrow \widehat{\boldsymbol{L}}$, $\boldsymbol{L}_0 \leftarrow \widehat{\boldsymbol{L}}$, $\lambda_\gamma \leftarrow 16\lambda\gamma$

  while $d_{\mathrm{i}} > \epsilon_{\mathrm{i}}$ **and** $j < J$ **do**

    $j \leftarrow j + 1$

    $\boldsymbol{L}_j \leftarrow \mathcal{T}_{\mu\lambda_\gamma}^{(\gamma)}\left(\boldsymbol{Z}_j - \mu\mathcal{A}^{\mathrm{H}}\left(\mathcal{A}\left(\boldsymbol{Z}_j\right) - \boldsymbol{y}\right)\right)$

    $t_{j+1} \leftarrow (1 + \sqrt{1 + 4t_j^2})/2$

    $\boldsymbol{Z}_{j+1} \leftarrow \boldsymbol{L}_j + \left(t_j - 1\right)\left(\boldsymbol{L}_j - \boldsymbol{L}_{j-1}\right)/t_{j+1}$

    $d_{\mathrm{i}} \leftarrow \left\|\boldsymbol{L}_j - \boldsymbol{L}_{j-1}\right\|_{\mathrm{F}}/\left\|\boldsymbol{L}_{j-1}\right\|_{\mathrm{F}}$

  end while

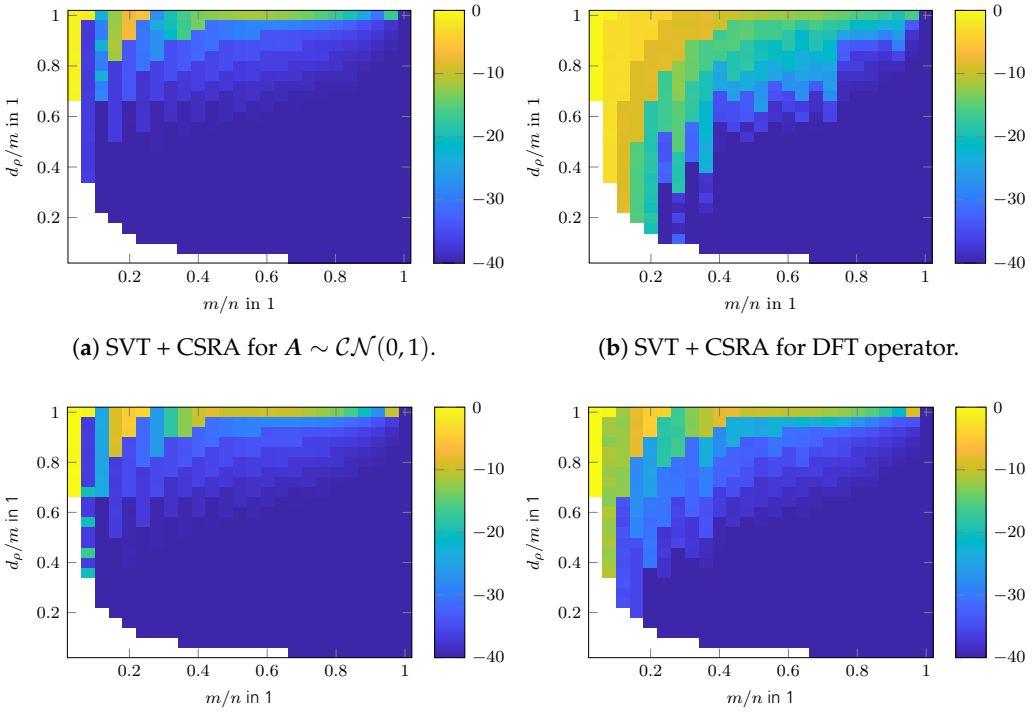
  $d_{\mathrm{o}} \leftarrow \left\|\boldsymbol{L}_j - \widehat{\boldsymbol{L}}\right\|_{\mathrm{F}}/\left\|\widehat{\boldsymbol{L}}\right\|_{\mathrm{F}}$

  $\widehat{\boldsymbol{L}} \leftarrow \boldsymbol{L}_j$

  $\gamma \leftarrow c\gamma$

end while

Output: $\widehat{\boldsymbol{L}}$



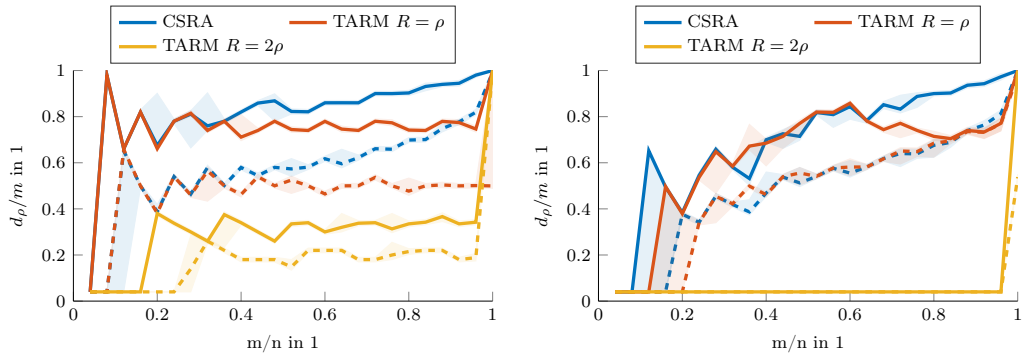
(**a**) SVT + CSRA for $\boldsymbol{A} \sim \mathcal{CN}(0, 1)$.

(**b**) SVT + CSRA for DFT operator.

(**c**) TSVT + CSRA for $\boldsymbol{A} \sim \mathcal{CN}(0, 1)$.

(**d**) TSVT + CSRA for DFT operator.

Figure 14. Phase transition of refined ARM algorithms in SRE in dB.



(**a**) $\boldsymbol{A} \sim \mathcal{CN}(0,1)$ (**b**) Random row DFT sensing operator.

Figure 15. Comparison of reconstruction performances of TSVT + CSRA to TARM. The solid lines show the 50% success rate for a strict reconstruction success, defined as SRE $\leq -$SNR, and the dashed lines the success rate for a less strict success definition, i.e., SRE $\leq -(\text{SNR} - 5\,\text{dB})$.

In summary, CSRA in combination with TSVT is very easy to implement, shows a similar convergence rate and reconstruction performance as TARM equipped with the true rank of the unknown low-rank matrix in estimation, and has low computational complexity, as there are closed form solutions available for all required parameters and subsequent optimization problems. Finally, both do not generally require any unknown parameters. Hence, they are well suited for practical applications.

4. Compressed Robust Principle Component Analysis

Turbo Compressed Robust Principle Component Analysis

For TCRPCA, all of the aforementioned reconstruction algorithms for CS and ARM problems are combined together. Following the GNC approach described in Section 2.2, the corresponding relaxed convex Problem (21) is solved via a combination of TST and TSVT by iteratively updating

$$\begin{aligned} \boldsymbol{R}_{s,i} &= \boldsymbol{S}_{i-1} - \mu_s \nabla_{\boldsymbol{S}} h(\boldsymbol{S}_{i-1}, \boldsymbol{L}_{i-1}) \\ \boldsymbol{Z}_{s,i} &= \mathcal{S}_{s,\mu_s\lambda_{s0}}(\boldsymbol{R}_{s,i}) \\ \boldsymbol{S}_i &= \mathcal{Q}_{\kappa_s}(c_{s,i}(\boldsymbol{Z}_{s,i} - \alpha_{s,i}\boldsymbol{R}_{s,i})) \end{aligned}$$

$$\begin{aligned} \boldsymbol{R}_{l,i} &= \boldsymbol{L}_{i-1} - \mu_l \nabla_{\boldsymbol{L}} h(\boldsymbol{S}_{i-1}, \boldsymbol{L}_{i-1}) \\ \boldsymbol{Z}_{l,i} &= \mathcal{S}_{l,\mu_l\lambda_l}(\boldsymbol{R}_{l,i}) \\ \boldsymbol{L}_i &= \mathcal{P}_{\varphi_l}(c_{l,i}(\boldsymbol{Z}_{l,i} - \alpha_{l,i}\boldsymbol{R}_{l,i})). \end{aligned}$$

The required parameters μ_s, $c_{s,i}$, and $\alpha_{s,i}$ are determined as for the TST algorithm explained in Section 2.1 and μ_l, λ_l, $c_{l,i}$, and $\alpha_{l,i}$ as for the TSVT algorithm explained in Section 3.1. The regularization parameter λ_{s0} is chosen as defined in (33). The combination of TST and TSVT is inspired by the turbo algorithms presented in [58,59]. In contrast to those works, only ST operators are used in the TCRPCA algorithm, whose parameters are not learned or determined via excessive grid search. Furthermore, in order to support the incoherence condition of the sparse and low-rank matrices $\boldsymbol{S}$ and $\boldsymbol{L}$, the sparsity ratio operator $\mathcal{Q}_{\kappa_s} : \mathbb{C}^{N_1 \times N_2} \to \mathbb{C}^{N_1 \times N_2}$ and infinity norm operator $\mathcal{P}_{\varphi_l} : \mathbb{C}^{N_1 \times N_2} \to \mathbb{C}^{N_1 \times N_2}$ may be applied [49]. The usage of $\mathcal{Q}_{\kappa_s}$ prevents clustering of sparse entries and is defined element-wise as

$$[\mathcal{Q}_{\kappa_s}(\boldsymbol{S})]_{ij} = \begin{cases} s_{ij} & \text{if } |s_{ij}| \geq \boldsymbol{S}_{i:}^{[\lfloor \kappa_s N_2 \rfloor]} \;\&\; |s_{ij}| \geq \boldsymbol{S}_{:j}^{[\lfloor \kappa_s N_1 \rfloor]} \\ 0 & \text{else,} \end{cases} \tag{63}$$

where $\kappa_s \in (0,1)$, s_{ij} denotes the (i,j)-th entry of $\boldsymbol{S}$, $\boldsymbol{S}_{i:}$ the i-th row, $\boldsymbol{S}_{:j}$ the j-th column of $\boldsymbol{S}$ in Matlab notation, and $\boldsymbol{x}^{[a]}$ the a-th biggest entry in magnitude in $\boldsymbol{x}$. The usage of $\mathcal{P}_\varphi$ prevents spikiness in the low-rank reconstruction and is defined element-wise as

$$\left[\mathcal{P}_\varphi(\boldsymbol{L})\right]_{ij} = \begin{cases} \varphi \exp\left(\mathrm{j}\arg(l_{ij})\right), & \text{if } |l_{ij}| \geq \varphi \\ l_{ij}, & \text{else} \end{cases}. \tag{64}$$

Unfortunately, the required parameters κ_s and φ_l are unknown in general. For κ_s, a reasonable guess is required and φ_l can be determined from

$$\varphi_l = \frac{\mu\rho c_\varphi \left\|\widetilde{\boldsymbol{L}}\right\|_2}{\sqrt{N_1 N_2}}, \tag{65}$$

where c_φ is a parameter set manually usually < 1 [32,49]. Most parameters in (65) are unknown, since knowledge of the true low-rank matrix $\widetilde{\boldsymbol{L}}$ is required. As a rough estimate, we may use $\left\|\widetilde{\boldsymbol{L}}\right\|_2 \approx \left\|\mathcal{A}^{\mathrm{H}}(\boldsymbol{y})\right\|_2$, $\mu = 1$, and $\rho \approx \min(N_1, N_2)/2$. Certainly, this estimate is not justified to be anywhere close to an optimal value, but it was found from simulations that it is sufficient to apply $\mathcal{Q}_{\kappa_s}$ and $\mathcal{P}_{\varphi_l}$ only for a limited number of iterations, e.g., the first 10 iterations. The intermediate results $\boldsymbol{S}_i$ and $\boldsymbol{L}_i$ then lie in a surrounding of a diffuse sparse and low-rank solution, and subsequent iterations do not need any further "guidance" by the projection operators. This also circumvents the problem of not knowing the optimal parameters κ_s and φ_l.

Once a suitable convex solution was obtained, a refinement is conducted by solving

$$\widehat{\boldsymbol{S}}, \widehat{\boldsymbol{L}} = \underset{\boldsymbol{S},\boldsymbol{L}}{\arg\min}\, \lambda_s F_{0,\gamma_s}(\boldsymbol{S}) + \lambda_l F_{\mathrm{r},\gamma_l}(\boldsymbol{L}) + h(\boldsymbol{S},\boldsymbol{L}), \tag{66}$$

where $F_{0,\gamma_s}(\cdot)$ is the ℓ_0-approximation function (40) and $F_{\mathrm{r},\gamma_l}(\cdot)$ is the rank approximation Function (59). Program (66) is solved via a combination of CSCSA and CSRA by iteratively updating

$$\begin{aligned} \boldsymbol{S}_i &= \mathcal{T}^{(\gamma_s)}_{0,\mu_s\lambda_{s,\gamma_s}}(\boldsymbol{Z}_{s,i} - \mu_s \nabla_{\boldsymbol{S}} h(\boldsymbol{Z}_{s,i}, \boldsymbol{Z}_{l,i})) \\ \boldsymbol{L}_i &= \mathcal{T}^{(\gamma_l)}_{\mathrm{r},\mu_l\lambda_l}(\boldsymbol{Z}_{l,i} - \mu_l \nabla_{\boldsymbol{L}} h(\boldsymbol{Z}_{s,i}, \boldsymbol{Z}_{l,i})) \\ t_{i+1} &= \frac{1 + \sqrt{1 + 4t_i^2}}{2} \\ \boldsymbol{Z}_{s,i+1} &= \boldsymbol{S}_i + \frac{(t_i - 1)}{t_{i+1}}(\boldsymbol{S}_i - \boldsymbol{S}_{i-1}) \\ \boldsymbol{Z}_{l,i+1} &= \boldsymbol{L}_i + \frac{(t_i - 1)}{t_{i+1}}(\boldsymbol{L}_i - \boldsymbol{L}_{i-1}), \end{aligned}$$

where $\mathcal{T}^{(a)}_{0,b}(\cdot)$ is the thresholding operator as defined in CSCSA in Section 2.2 and $\mathcal{T}^{(a)}_{\mathrm{r},b}(\cdot)$ is the thresholding operator as defined in CSRA in Section 3.2. The required parameters λ_s, γ_s, and γ_l are also the same as defined in CSCSA and CSRA.

Putting all the above steps together, the final TCRPCA algorithm is listed in Algorithm5, which delivers a solution to program (21), and Algorithm 6 which solves for Program (66).

Algorithm 5 Part 1 of TCRPCA algorithm delivering convex solution.

Input: $\mathcal{A}$, $\boldsymbol{y}$, λ_s, λ_l, κ_s, φ_l, I

Initialization:

$\epsilon_s \leftarrow \min\left(10^{-4}, 5 \cdot 10^{-3}\lambda_s\right)$
$\epsilon_l \leftarrow \min\left(10^{-4}, 5 \cdot 10^{-3}\lambda_l\right)$
$\boldsymbol{S}_0 \leftarrow \boldsymbol{0}$, $\boldsymbol{L}_0 \leftarrow \boldsymbol{0}$, $\mu_s \leftarrow n/(m\|\mathcal{A}\|_2^2)$
$\mu_l \leftarrow n/(m\|\mathcal{A}\|_2^2)$, $i \leftarrow 0$, $d_s \leftarrow \infty$, $d_l \leftarrow \infty$

Body:

while ($d_s > \epsilon_s$ **or** $d_l > \epsilon_l$) **and** $i < I$ **do**
  $i \leftarrow i + 1$
  $\boldsymbol{R}_{s,i} \leftarrow \boldsymbol{S}_{i-1} - \mu_s \nabla_{\boldsymbol{S}} h(\boldsymbol{S}_{i-1}, \boldsymbol{L}_{i-1})$
  $\boldsymbol{Z}_{s,i} \leftarrow \mathcal{S}_{s,\mu_s\lambda_s}(\boldsymbol{R}_{s,i})$
  $\alpha_{s,i} \leftarrow$(30)
  $c_{s,i} \leftarrow$(31)
  $\boldsymbol{S}_i \leftarrow \mathcal{Q}_{\kappa_s}(c_{s,i}(\boldsymbol{Z}_{s,i} - \alpha_{s,i}\boldsymbol{R}_{s,i}))$
  $\boldsymbol{R}_{l,i} \leftarrow \boldsymbol{L}_{i-1} - \mu_l \nabla_{\boldsymbol{L}} h(\boldsymbol{S}_{i-1}, \boldsymbol{L}_{i-1})$
  $\boldsymbol{Z}_{l,i} \leftarrow \mathcal{S}_{l,\mu_l\lambda_l}(\boldsymbol{R}_{l,i})$
  $\alpha_{l,i} \leftarrow$(49)
  $c_{l,i} \leftarrow$(50)
  $\boldsymbol{L}_i \leftarrow \mathcal{P}_{\varphi_l}(c_{l,i}(\boldsymbol{Z}_{l,i} - \alpha_{l,i}\boldsymbol{R}_{l,i}))$
  $d_s \leftarrow \|\boldsymbol{S}_i - \boldsymbol{S}_{i-1}\|_{\mathrm{F}} / \|\boldsymbol{S}_{i-1}\|_{\mathrm{F}}$
  $d_l \leftarrow \|\boldsymbol{L}_i - \boldsymbol{L}_{i-1}\|_{\mathrm{F}} / \|\boldsymbol{L}_{i-1}\|_{\mathrm{F}}$
end while

Output: $\widehat{\boldsymbol{S}} \leftarrow \boldsymbol{R}_{s,i}$, $\widehat{\boldsymbol{L}} \leftarrow \boldsymbol{R}_{l,i}$

In the following, simulation results are shown to illustrate the performance of TCRPCA for a DFT and a noiselet sensing operator instead of a random sensing operator for reasons of computational load. The sparse and low-rank matrices $\widetilde{\boldsymbol{S}}$ and $\widetilde{\boldsymbol{L}}$ are generated similar as for the aforementioned algorithms in Sections 2 and 3, however, for a size of $N_1 = N_2 = 128$. The performance evaluation follows an approach similar as in [44]. The low-rank matrices $\widetilde{\boldsymbol{L}}$ are set up for fixed ranks $\rho = \{3, 5, 7, 9\}$ and the phase transition plots are evaluated over the sparsity rate $\kappa/m > 0$; hence, the sparse matrices $\widetilde{\boldsymbol{S}}$ are accordingly set up. The resulting phase transition plots illustrated in Figure 16 indicate the 50 % success rate with respect to the Monte Carlo runs, where 20 runs are conducted. Success is defined as SRE $\leq -$ SNR, where the SRE of the reconstructed low-rank matrices $\widehat{\boldsymbol{L}}$ is used. These transition plots thus provide information of up to which sparsity rate or "corruption rate" the low-rank matrices can still be reconstructed.

As comparison benchmarks, the aforementioned simulations are also conducted for SpaRCS, NFL, and turbo message passing for CRPCA (TMP-CRPCA) algorithm, which uses HT instead of ST denoisers. Additional comparisons of CRPCA algorithms are given in [68]. The SpaRCS algorithm is equipped with the true number of sparse entries κ and rank ρ. Its implementation is taken from its official Github repository (https://github.com/image-science-lab/SpaRCS, accessed on 20 December 2022). The maximum number of iterations is set to $I = 300$, and for all remaining parameters the default setting is used. The NFL and TMP-CRPCA algorithms are implemented according to [44,59] and, likewise, equipped with the true number of sparse entries κ and rank ρ. For the NFL algorithm, only the initial phase algorithm is used as the gradient descent phase algorithm merely serves as a final fast refinement step. For the initial phase algorithm, the maximum number of iterations of the Dykstra projection step are set to $I = 5$, and its corresponding infinity threshold value φ_l, as defined in (65), is chosen similar to the TCRPCA algorithm. The abortion criteria of NFL and TMP-CRPCA are also set as for the TCRPCA algorithm.

Algorithm 6 Part 2 of TCRPCA algorithm delivering refined solution.

Input: $\mathcal{A}$, $\boldsymbol{y}$, λ_s, λ_l, $\widehat{\boldsymbol{S}}_0$, $\widehat{\boldsymbol{L}}_0$, I, J

Initialization:

$c \leftarrow (0, 0.5)$, $\mu_s \leftarrow 0.99/\|\mathcal{A}\|_2^2$, $\mu_l \leftarrow 0.99/\|\mathcal{A}\|_2^2$

$\gamma_s \leftarrow \max(|\widehat{\boldsymbol{S}}|)/10$, $\gamma_l \leftarrow \left\|\widehat{\boldsymbol{L}}\right\|_2/10$

$\epsilon_{\text{si}} \leftarrow \min\left(10^{-3}, 5 \cdot 10^{-3}\lambda_s\right)$

$\epsilon_{\text{so}} \leftarrow \min\left(10^{-4}, 5 \cdot 10^{-3}\lambda_s\right)$

$\epsilon_{l\text{i}} \leftarrow \min\left(10^{-3}, 5 \cdot 10^{-3}\lambda_l\right)$

$\epsilon_{l\text{o}} \leftarrow \min\left(10^{-4}, 5 \cdot 10^{-3}\lambda_l\right)$

Body:

$i \leftarrow 0$, $d_{\text{so}} \leftarrow \infty$, $d_{\text{lo}} \leftarrow \infty$

while ($d_{\text{so}} > \epsilon_{\text{so}}$ **or** $d_{l\text{o}} > \epsilon_{l\text{o}}$) **and** $i < I$ **do**

  $i \leftarrow i + 1$, $j \leftarrow 0$, $d_{\text{si}} \leftarrow \infty$, $d_{l\text{i}} \leftarrow \infty$

  $t_1 \leftarrow 1$, $\boldsymbol{Z}_{s,1} \leftarrow \widehat{\boldsymbol{S}}$, $\boldsymbol{Z}_{l,1} \leftarrow \widehat{\boldsymbol{L}}$, $\boldsymbol{S}_0 \leftarrow \widehat{\boldsymbol{S}}$, $\boldsymbol{L}_0 \leftarrow \widehat{\boldsymbol{L}}$

  $\lambda_{s,\gamma_s} \leftarrow 2\lambda_s\gamma_s$, $\lambda_{l,\gamma_l} \leftarrow 16\lambda_l\gamma_l$

  while ($d_{\text{si}} > \epsilon_{\text{si}}$ **or** $d_{l\text{i}} > \epsilon_{l\text{i}}$) **and** $j < J$ **do**

    $j \leftarrow j + 1$

    $\boldsymbol{S}_j \leftarrow \mathcal{T}^{(\gamma_s)}_{0,\mu_s\lambda_{s,\gamma_s}}\left(\boldsymbol{Z}_{s,j} - \mu_s\nabla_{\boldsymbol{S}}h\left(\boldsymbol{Z}_{s,j}, \boldsymbol{Z}_{l,j}\right)\right)$

    $\boldsymbol{L}_j \leftarrow \mathcal{T}^{(\gamma_l)}_{\text{r},\mu_l\lambda_{l,\gamma_l}}\left(\boldsymbol{Z}_{l,j} - \mu_l\nabla_{\boldsymbol{L}}h\left(\boldsymbol{Z}_{s,j}, \boldsymbol{Z}_{l,j}\right)\right)$

    $t_{j+1} \leftarrow (1 + \sqrt{1 + 4t_j^2})/2$

    $\boldsymbol{Z}_{s,j+1} \leftarrow \boldsymbol{S}_j + (t_j - 1)(\boldsymbol{S}_j - \boldsymbol{S}_{j-1})/t_{j+1}$

    $\boldsymbol{Z}_{l,j+1} \leftarrow \boldsymbol{L}_j + (t_j - 1)(\boldsymbol{L}_j - \boldsymbol{L}_{j-1})/t_{j+1}$

    $d_{\text{si}} \leftarrow \left\|\boldsymbol{S}_j - \boldsymbol{S}_{j-1}\right\|_{\text{F}}/\left\|\boldsymbol{S}_{j-1}\right\|_{\text{F}}$

    $d_{l\text{i}} \leftarrow \left\|\boldsymbol{L}_j - \boldsymbol{L}_{j-1}\right\|_{\text{F}}/\left\|\boldsymbol{L}_{j-1}\right\|_{\text{F}}$

  end while

  $d_{\text{so}} \leftarrow \left\|\boldsymbol{S}_j - \widehat{\boldsymbol{S}}\right\|_{\text{F}}/\left\|\widehat{\boldsymbol{S}}\right\|_{\text{F}}$

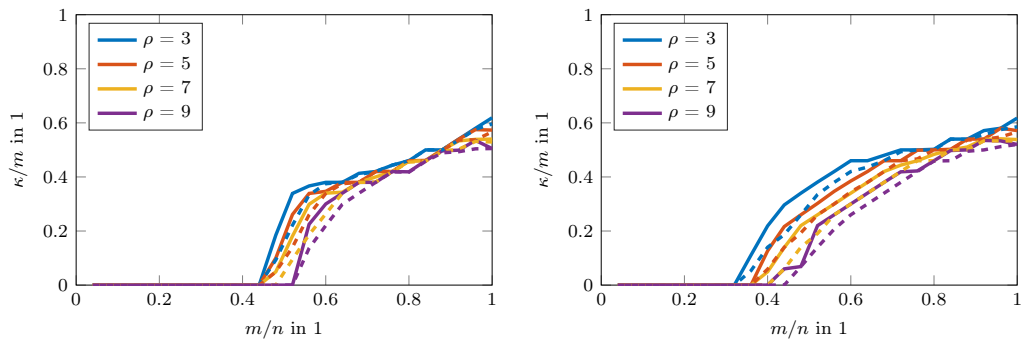
  $d_{l\text{o}} \leftarrow \left\|\boldsymbol{L}_j - \widehat{\boldsymbol{L}}\right\|_{\text{F}}/\left\|\widehat{\boldsymbol{L}}\right\|_{\text{F}}$

  $\widehat{\boldsymbol{S}} \leftarrow \boldsymbol{S}_j$, $\widehat{\boldsymbol{L}} \leftarrow \boldsymbol{L}_j$

  $\gamma_s \leftarrow c\gamma_s$, $\gamma_l \leftarrow c\gamma_l$

end while

Output: $\widehat{\boldsymbol{S}}$,$\widehat{\boldsymbol{L}}$



(**a**) (**b**)

Figure 16. *Cont.*

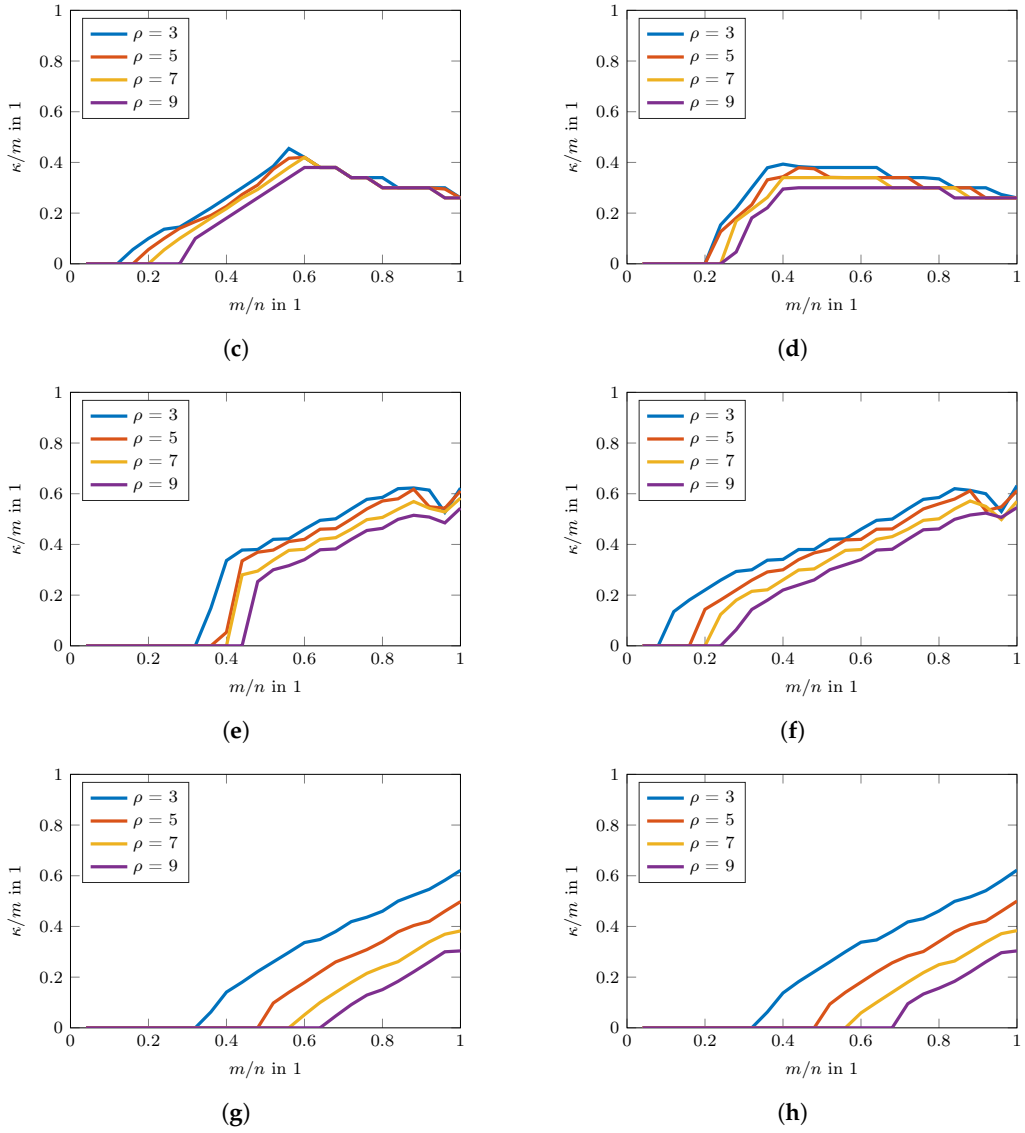
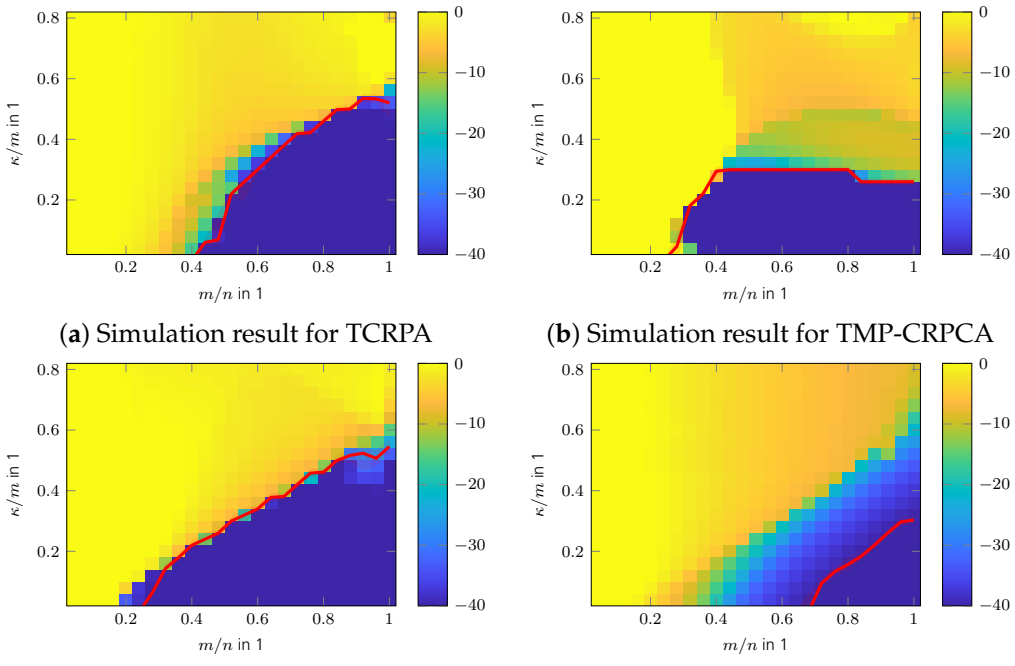


Figure 16. Comparison of reconstruction performance. Dashed lines illustrate results of Part 1 of the TCRPCA algorithm given by Algorithm 5 and solid lines the refined results achieved by Part 2 the TCRPCA algorithm given by Algorithm 6. (**a**) TCRPCA for noiselet operator. (**b**) TCRPCA for DFT operator. (**c**) TMP-CRPCA for noiselet operator with $K = \kappa$ and $R = \rho$. (**d**) TMP-CRPCA for DFT operator with $K = \kappa$ and $R = \rho$. (**e**) NFL for noiselet operator with $K = \kappa$ and $R = \rho$. (**f**) NFL for DFT operator with $K = \kappa$ and $R = \rho$. (**g**) SpaRCS for noiselet operator with $K = \kappa$ and $R = \rho$. (**h**) SpaRCS for DFT operator with $K = \kappa$ and $R = \rho$.

As can be seen in Figure 16a,b, the reconstruction performance of TCRPCA gracefully degrades with increasing rank ρ. For the generation of these phase transition plots, no sparsity ratio operator $\mathcal{Q}_{\kappa_s}$ and infinity norm operator $\mathcal{P}_{\varphi_l}$ are applied, since these have turned out to be unnecessary for $\kappa > 0$, $\rho > 0$, and SNR ≥ 10 dB. The special cases of $\kappa = 0$ or $\rho = 0$ are treated further below. The performance gain of the refinement step achieved by Part 2 of the TCRPCA algorithm appears moderate compared with the gains achieved in the pure CS and ARM applications. The reason therefore is that Part 1 of the TCRPCA algorithm shows a rather sharp transition in SRE, and thus Part 2 of TCRPCA lacks a sufficiently good initial solution to improve upon (not shown here). Nevertheless, the refinement step constitutes a computationally efficient and fast procedure offering increased reconstruction performance. In comparison, TMP-CRPCA and NFL shown in Figure 16c–f offer higher reconstruction performance for lower measurement rates m/n,

with NFL offering the highest. The results for SpaRCS illustrated in Figure 16g,h show a comparable performance in the case the rank of the low-rank matrix is as low as $\rho = 3$. Its performance, however, rapidly degrades with increasing rank ρ. The reason therefore is that SpaRCS, in contrast to the remaining algorithms, lacks a sharp phase transition as illustrated in Figure 17. In this simulation, TMP-CRPCA, NFL, and SpaRCS are equipped with the true sparsity and rank parameters of the unknown matrices to reconstruct, namely $K = \kappa$ and $R = \rho$. In the case the parameters are set to $K = 2\kappa$ and $R = 2\rho$, all algorithms completely fail and, as such, are not shown here. A comparison of the convergence speed is shown in Figure 18, which shows the intermediate SREs. In this comparison, only Part 1 of the TCRPCA algorithm is shown. As can be seen, TMP-CRPCA, NFL, and SpaRCS show altogether a superior convergence rate compared with Part 1 of TCRPCA. This is in stark contrast to the pure CS and ARM counterparts shown in Sections 2.1 and 3.1. Further evaluations revealed that Part 1 of TCRPCA requires $\sim 1/4$ of its iterations to correctly identify κ and ρ, and the remaining iterations are used to minimize the reconstruction error. A comparison of the computation time is shown in Figure 19 for a region where all algorithms perform equally well. The system specifications are the same as given in Section 2. The overall fastest algorithm is TMP-CRPCA. In cases where the sensing operator can be implemented in an efficient manner, as is the case for the DFT sensing operator, the Part 1 TCRPCA algorithm offers the slowest computation time. In cases where the sensing operator is computationally more expansive to evaluate, Part 1 TCRPCA becomes more efficient and is faster than SpaRCS. The NFL algorithm is overall faster than in Part 1 TCRPCA. It should be noted that Part 1 TCRPCA can be aborted and Part 2 TCRPCA started sooner in order to reduce its overall computation time. This, however, is not done here for sake of clear evaluation. Finally, the SRE for various SNRs is shown in Figure 20 for all tested algorithms. For Part 1 of TCRPCA, the performance is shown with and without the spikiness operator $\mathcal{P}_\varphi(\boldsymbol{L})$ defined in (65) for $c_\varphi = 1/4$. As can be seen, for SNR $= 0$ dB, the spikiness operator is required to support the reconstruction, especially for higher sparsity ratios κ/m. In the case of higher SNRs, no supporting operator is required anymore; in fact, it is obstructive at high SNR. For these simulations, no sparsity ratio operator $\mathcal{Q}_{\kappa_s}$ is required. The best SRE offer TMP-CRPCA and NFL provided $K = \kappa$ and $R = \rho$.



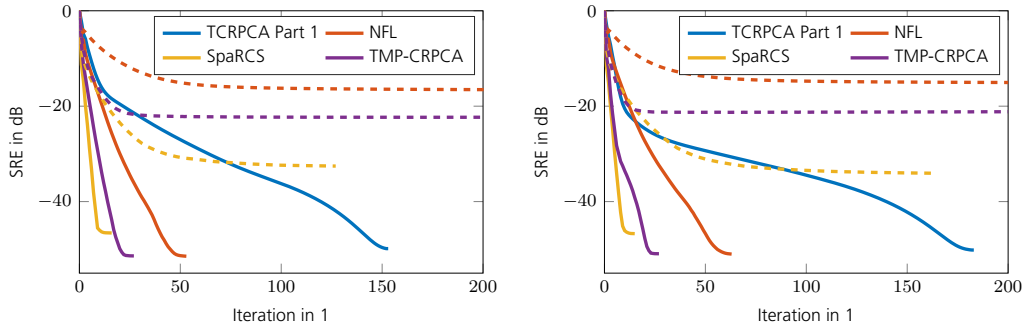
(**a**) Simulation result for TCRPA

(**b**) Simulation result for TMP-CRPCA

(**c**) Simulation result for NFL

(**d**) Simulation result for SpaRCS

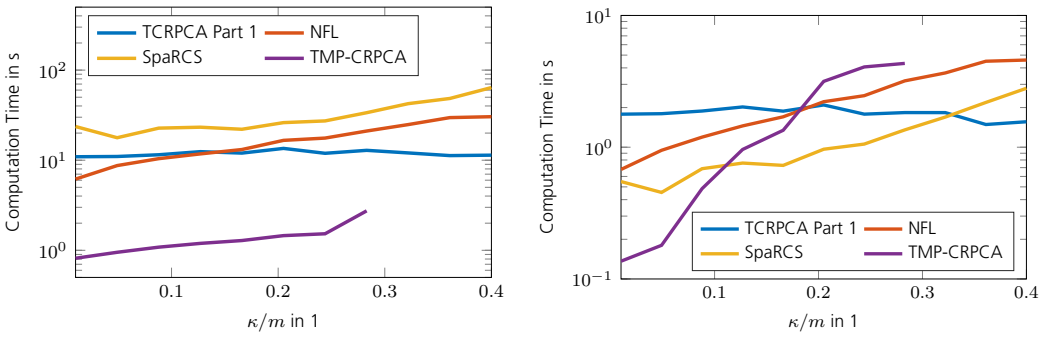
Figure 17. In detail reconstruction performance comparison for a DFT sensing operator and $\rho = 9$. The red line indicates the success boundary or the phase transitions shown in Figure 16.



(**a**) Noiselet sensing operator. (**b**) DFT sensing operator.

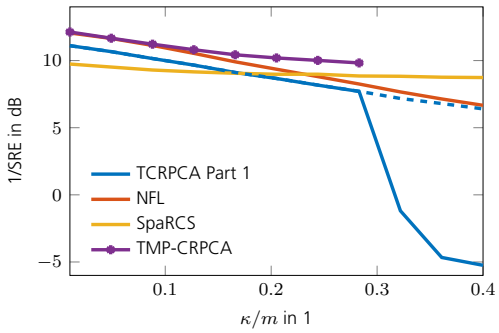
Figure 18. Comparison of convergence speed for $\rho = 3$, $\kappa/m = 0.2$ and $m/n = 0.8$. The solid lines show NFL, SpaRCS, and TMP-CRPCA for $K = \kappa$ and $R = \rho$ and the dashed lines for $K = 2\kappa$ and $R = 2\rho$.

For CRPCA, $\widetilde{\boldsymbol{S}} = \mathbf{0}$ and $\widetilde{\boldsymbol{L}} = \mathbf{0}$ represent special cases. In order to allow for successful reconstructions in the case of $\widetilde{\boldsymbol{S}} = \mathbf{0}$, it is found from simulations that the sparsity ratio operator $\mathcal{Q}_{\kappa_s}$ is required for all iterations with a setting of $\kappa_s \leq 0.25$ to allow for a successful identification of $\widehat{\boldsymbol{S}} = \mathbf{0}$. Likewise, the infinity norm operator $\mathcal{P}_{\varphi_l}$ with a setting of $c_\varphi \leq 0.5$ is found to be required for all iterations to successfully identify the special case of $\widehat{\boldsymbol{L}} = \mathbf{0}$. The use of the sparsity ratio and infinity norm operators incur restrictions regarding the possible reconstruction performance of the TCRPCA algorithm. Obviously, the use of $\mathcal{Q}_{\kappa_s}$ with $\kappa_s = 0.25$ prohibits successful reconstructions in the case of $\kappa/\min(N_1, N_2) > 0.25$. In a similar manner, the use of $\mathcal{P}_{\varphi_l}$ with $c_\varphi = 0.5$ prohibits successful reconstructions in the case $\rho/\min(N_1, N_2) > 0.25$. How to treat the special cases of either $\widetilde{\boldsymbol{S}} = \mathbf{0}$ or $\widetilde{\boldsymbol{L}} = \mathbf{0}$ in a satisfactory manner, i.e., how to relax κ_s and c_φ conveniently, is an open question and subject to further investigation. The obvious approach of conducting a CS, an ARM, and an CRPCA reconstruction separately and comparing the resulting residual errors to determine if either $\widetilde{\boldsymbol{S}} = \mathbf{0}$ or $\widetilde{\boldsymbol{L}} = \mathbf{0}$ does not work. Particularly in the case of low n/m, the CRPCA approach always yields the lowest residual error regardless if the scene is strictly sparse or low-rank due to its larger degree of freedom.

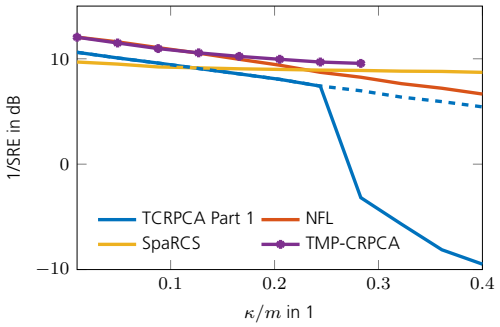


(**a**) (**b**)

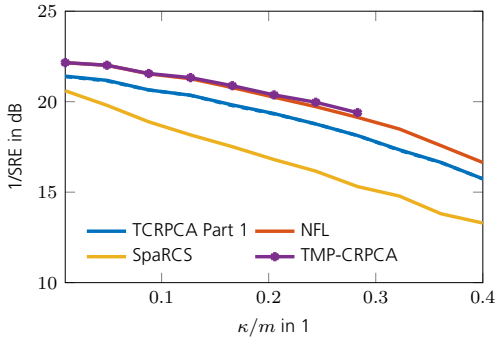
Figure 19. Comparison of computation time for $\rho = 3$ and $m/n = 0.8$, i.e., in an interval where all algorithms perform equally well. For NFL, SpaRCS, and TMP-CRPCA, the results are shown for $K = \kappa$ and $R = \rho$ only. For $\kappa/m > 0.3$, TMP-CRPCA did not converge and is thus not shown here. (**a**) Noislet sensing operators. (**b**) DFT sensing operators with random rows.



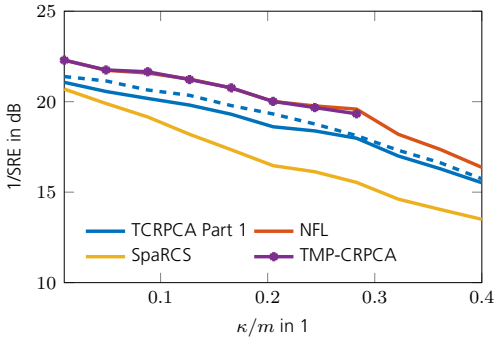
(**a**) Noislet sensing operator for SNR = 0 dB.



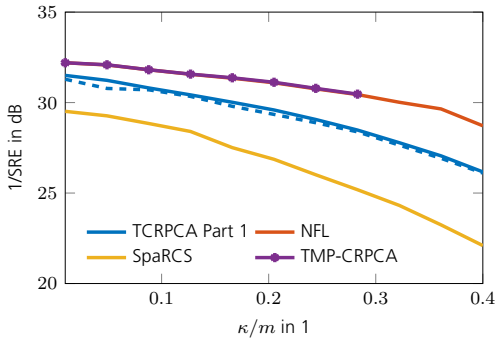
(**b**) DFT sensing operator for SNR = 0 dB.



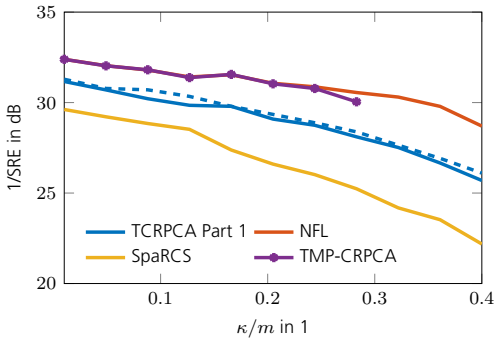
(**c**) Noislet sensing operator for SNR = 10 dB.



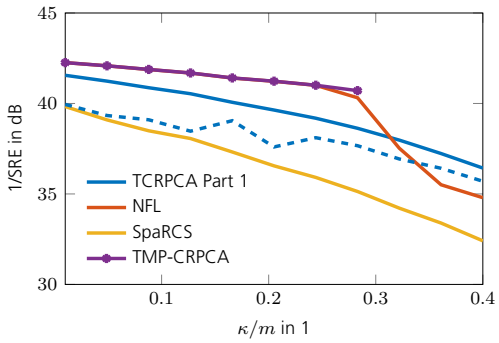
(**d**) DFT sensing operator for SNR = 10 dB.



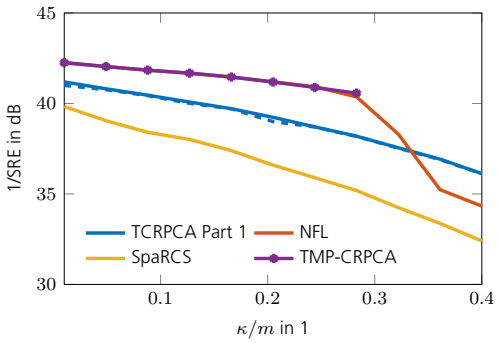
(**e**) Noislet sensing operator for SNR = 20 dB.



(**f**) DFT sensing operator for SNR = 20 dB.



(**g**) Noislet sensing operator for SNR = 30 dB.



(**h**) DFT sensing operator for SNR = 30 dB.

Figure 20. Comparison of the low-rank SRE vs. SNR for $m/n = 0.8$. The solid line for TCRPCA shows the performance without and the dashed line with the spikiness operator $\mathcal{P}_\varphi(\boldsymbol{L})$ defined in (64), respectively.

In summary, TCRPCA offers comparable reconstruction performance to its greedy and HT counterparts despite its unawareness of the true sparsity and rank values. It is very easy to implement and has low computational complexity, as there are closed form solutions available for all required parameters and subsequent optimization problems. In the case of $\widetilde{\boldsymbol{S}} \neq \boldsymbol{0}$ and $\widetilde{\boldsymbol{L}} \neq \boldsymbol{0}$, no generally unknown parameters are required for successful reconstruction. Hence, TCRPCA is well suited for practical applications. In the special cases $\widetilde{\boldsymbol{S}} = \boldsymbol{0}$ or $\widetilde{\boldsymbol{L}} = \boldsymbol{0}$, TCRPCA is capable of a successful reconstruction if $\kappa / \min(N_1, N_2) \leq 0.25$ or $\rho / \min(N_1, N_2) \leq 0.25$, respectively.

5. Conclusions

In this paper, fast, efficient, and viable CS, ARM, and CRPCA algorithms suitable for radar signal processing are proposed. They are designed such that no parameters unknown in practice, e.g., the number of sparse entries or the rank of the unknown low-rank matrix, are required. The only parameter that is needed to be known is the noise power, which in the field of radar signal processing is usually available. For all remaining parameters, either suitable heuristic formulas or closed form solutions are given. The general reconstruction scheme comprises two steps: First, a convex solution is calculated for which the turbo-message-passing framework is utilized. This initial solution is in a second step refined by use of smoothed ℓ_0-refinements. The proposed algorithms for CS are termed TST and CSCSA, for ARM problems TSVT and CSRA, and TCRPCA for the combined CS and ARM problems. All algorithms show state-of-the-art reconstruction performance and are of high computational efficiency, as closed form solutions are available for subsequent optimization tasks.

Funding: This research was funded by Hensoldt Sensor GmbH.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflicts of interest. In addition, the funders had no role in the writing of the manuscript.

Appendix A. Divergence of the Complex Soft-Thresholding Operator

In this section, the weak divergence of the complex soft thresholding operator is derived in closed form, which is defined as [69]

$$\mathcal{S}_{\mathrm{S}}^{(a)}(z) = \operatorname{sgn}(z) \max(0, |z| - a), \tag{A1}$$

where $z \in \mathbb{C}$, $a \in \mathbb{R}_+$, and

$$\operatorname{sgn}(z) = \begin{cases} 0 & \text{if } z = 0 \\ \frac{z}{|z|} & \text{else} \end{cases} \tag{A2}$$

is the complex sign function. A useful alternative formulation of the second term in (A1) is

$$\max(0, |z| - a) = \begin{cases} 0 & \text{if } |z| \leq a \\ |z| - a & \text{if } |z| > a \end{cases}. \tag{A3}$$

Furthermore, for some matrix $\boldsymbol{Z} \in \mathbb{C}^{N_1 \times N_2}$, the complex soft thresholding operator is defined as an element-wise operation as

$$\left[\mathcal{S}_{\mathrm{S}}^{(a)}(\boldsymbol{Z})\right]_{ij} = \mathcal{S}_{\mathrm{S}}^{(a)}(z_{ij}) \tag{A4}$$

with $i = 1, 2, \ldots, N_1$ and $j = 1, 2, \ldots, N_2$. Finally, the definition of the divergence for a scalar complex function $f : \mathbb{C} \to \mathbb{C}$ is

$$\operatorname{div} f = 2 \operatorname{Re}\left(\frac{\partial f(z)}{\partial z}\right)$$

and for a multidimensional function $\boldsymbol{F} : \mathbb{C}^N \to \mathbb{C}^N$ [70]

$$\operatorname{div} \boldsymbol{F} = \sum_{i=1}^{N} 2 \operatorname{Re}\left(\frac{\partial F_i(\boldsymbol{Z})}{\partial z_i} \right). \tag{A5}$$

The mapping f or $\boldsymbol{F}$ may not be differentiable everywhere. Fortunately, in the case of weak divergence, only weak differentiability is required. For such, sets of Lebesgue measure zero can be discarded [67]. Combining (A4) and (A5) yields the desired divergence

$$\operatorname{div}\left(\mathcal{S}_{\mathrm{S}}^{(a)}(\boldsymbol{Z}) \right) = \sum_{i,j=1}^{N_1,N_2} 2 \operatorname{Re}\left(\frac{\partial \left[\mathcal{S}_{\mathrm{S}}^{(a)}(\boldsymbol{Z}) \right]_{ij}}{\partial z_{ij}} \right). \tag{A6}$$

Using (A3), the required derivative is

$$\begin{aligned} \frac{\partial \left[\mathcal{S}_{\mathrm{S}}^{(a)}(\boldsymbol{Z}) \right]_{ij}}{\partial z_{ij}} = & \underbrace{\frac{\partial \operatorname{sgn}(z_{ij})}{\partial z_{ij}} \max(0, |z_{ij}| - a)}_{T_1} \\ & + \underbrace{\operatorname{sgn}(z) \frac{\partial \max(0, |z_{ij}| - a)}{\partial z_{ij}}}_{T_2}. \end{aligned}$$

After a few steps, the derivative in T_1 results in

$$\frac{\partial \operatorname{sgn}(z_{ij})}{\partial z_{ij}} = \begin{cases} 0 & \text{if } z_{ij} = 0 \\ \frac{1}{2|z_{ij}|} & \text{else} \end{cases}, \tag{A7}$$

where the discontinuity at $z_{ij} = 0$ can be discarded since $\mathcal{S}_{\mathrm{S}}^{(a)}(z) = 0$ for $|z| < a$. The derivative in T_2 yields

$$\frac{\partial \max(0, |z_{ij}| - a)}{\partial z_{ij}} = \begin{cases} 0 & \text{if } |z_{ij}| < a \\ \text{not differentiable} & \text{if } |z_{ij}| = a \\ \frac{z_{ij}^*}{2|z_{ij}|} & \text{if } |z_{ij}| > a. \end{cases}$$

The set $\{z : |z| = a\}$ has Lebesgue measure zero and can be discarded due to weak differentiability. Combining (A2), (A3), and (A7) yields after a few steps

$$T_1 = \begin{cases} 0 & \text{if } |z_{ij}| \leq a \\ \frac{|z_{ij}| - a}{2|z_{ij}|} & \text{if } |z_{ij}| > a \end{cases} \tag{A8}$$

and

$$T_2 = \begin{cases} 0 & \text{if } |z_{ij}| \leq a \\ \frac{1}{2} & \text{if } |z_{ij}| > a \end{cases}. \tag{A9}$$

Finally, combining (A6), (A8), and (A9) yields for the divergence

$$\operatorname{div}\left(\mathcal{S}_{\mathrm{S}}^{(a)}(\boldsymbol{Z}) \right) = \sum_{i,j=1}^{N_1,N_2} \left(2 - \frac{a}{|z_{ij}|} \right) \mathbb{I}(|z_{ij}| > a),$$

where $\mathbb{I}(\cdot)$ denotes the indicator function.

Appendix B. Complex Smoothed Rank Approximation

In this section, a minimization procedure to acquire a solution to the regularized smoothed rank Problem (61)

$$\min_{\boldsymbol{L}} \lambda_\gamma F_\gamma(\boldsymbol{L}) + h(\boldsymbol{L}) \tag{A10}$$

is derived. The smoothed rank function was defined in (59) as

$$F_\gamma(\boldsymbol{L}) = F'_\gamma(\boldsymbol{\sigma}(\boldsymbol{L})) = \sum_{i=1}^{n_{\min}} f_\gamma(\sigma_i(\boldsymbol{L})) \approx \operatorname{rank}(\boldsymbol{L}), \tag{A11}$$

where $f_\gamma(x)$ is given by (58). In program (A10), $h(\boldsymbol{L})$ is convex and differentiable with Lipschitz continuous gradient whereas $F_\gamma(\boldsymbol{L})$ is neither concave nor convex (since $f_\gamma(x)$ is also defined for negative numbers due to evidential requirements) and not differentiable at the origin. Nevertheless, the IT method can be utilized to conduct the desired minimization: For a fixed γ, a solution to program (A10) can be obtained by iteratively solving

$$\boldsymbol{L}_{j+1} = \arg\min_{\boldsymbol{L}} \left\{ \frac{1}{2\mu} \|\boldsymbol{L} - \boldsymbol{L}_{0j}\|_{\mathrm{F}}^2 + \lambda_\gamma F_\gamma(\boldsymbol{L}) \right\}, \tag{A12}$$

until convergence, where $\mu > 0$ is some step size and

$$\boldsymbol{L}_{0j} = \boldsymbol{L}_j - \mu \nabla h(\boldsymbol{L}_j) \tag{A13}$$

is the result of a gradient update step [63]. In order to minimize (A12), the following two theorems are useful.

Theorem A1. *The function $F(\boldsymbol{Z})$ is unitarily invariant if $F(\boldsymbol{Z}) = F'(\boldsymbol{\sigma}(\boldsymbol{Z})) = F' \circ \boldsymbol{\sigma}(\boldsymbol{Z})$ provided $F'(\boldsymbol{z})$ is absolutely symmetric, i.e., $F'(\boldsymbol{z})$ is invariant under arbitrary permutations and sign changes of the elements of $\boldsymbol{z}$ [60].*

This property applies to the rank approximation function $F'_\gamma(\boldsymbol{x})$ defined in (A11).

Theorem A2. *For unitarily invariant functions $F(\boldsymbol{Z}) = F' \circ \boldsymbol{\sigma}(\boldsymbol{Z})$ the optimal solution to the problem*

$$\min_{\boldsymbol{Z}} F(\boldsymbol{Z}) + c\|\boldsymbol{Z} - \boldsymbol{A}\|_{\mathrm{F}}^2$$

is

$$\widehat{\boldsymbol{Z}} = \boldsymbol{U}\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{Z}}\boldsymbol{V}^{\mathrm{H}},$$

where $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}_{\boldsymbol{A}}\boldsymbol{V}^{\mathrm{H}}$ is the SVD decomposition of $\boldsymbol{A}$ and $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{Z}} = \operatorname{diag}(\widehat{\boldsymbol{\sigma}})$ is obtained by solving the separable minimization problem

$$\widehat{\boldsymbol{\sigma}} = \arg\min_{\boldsymbol{\sigma}} \left\{ F'(\boldsymbol{\sigma}) + c\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_{\boldsymbol{A}}\|_2^2 \right\},$$

where $\boldsymbol{\sigma}_{\boldsymbol{A}} = \boldsymbol{\sigma}(\boldsymbol{A})$ [60].

It should be noted that Theorem A2 also works if $F(\boldsymbol{Z})$ would not be a unitarily invariant function, provided $F(\boldsymbol{Z}) = F(\boldsymbol{\Sigma}_{\boldsymbol{Z}})$, where $\boldsymbol{\Sigma}_{\boldsymbol{Z}} = \operatorname{diag}(\boldsymbol{\sigma}(\boldsymbol{Z}))$, which is inherently fulfilled since $\boldsymbol{\Sigma}_{\boldsymbol{Z}}$ is a real positive diagonal matrix. By use of Theorem A2, a solution to (A12) is obtained as

$$\boldsymbol{L}_{j+1} = \boldsymbol{U}_{0j}\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{L}}\boldsymbol{V}_{0j}^{\mathrm{H}}, \tag{A14}$$

where

$$\boldsymbol{L}_{0j} = \boldsymbol{U}_{0j}\boldsymbol{\Sigma}_{0j}\boldsymbol{V}_{0j}^{\mathrm{H}} \tag{A15}$$

is the SVD of $\boldsymbol{L}_{0j}$, $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{L}} = \operatorname{diag}(\widehat{\boldsymbol{\sigma}})$, and

$$\widehat{\boldsymbol{\sigma}} = \underset{\boldsymbol{\sigma} \geq \mathbf{0}}{\arg\min} \left\{ \lambda_\gamma F'_\gamma(\boldsymbol{\sigma}) + \frac{1}{2\mu} \left\| \boldsymbol{\sigma} - \boldsymbol{\sigma}_{0j} \right\|_2^2 \right\}, \tag{A16}$$

where $\boldsymbol{\sigma}_{0j} = \boldsymbol{\sigma}(\boldsymbol{L}_{0j})$. The objective function (A16) is the sum of a concave and convex function (since $\boldsymbol{\sigma} \geq \mathbf{0}$). To the contrary of [64], we do not solve (A16) by applying a D.C. optimization strategy which would require multiple iterations. An alternative approach, first shown in [63], is to utilize the Lambert W function

$$W(z)\mathrm{e}^{W(z)} = z, \tag{A17}$$

which allows for a closed form solution of (A16). We start by noticing that the minimization in (A16) is separable and as such can be conducted element wise as

$$\widehat{\sigma}_i = \underset{\sigma \geq 0}{\arg\min} \left\{ \lambda_\gamma f_\gamma(\sigma) + \frac{1}{2\mu} \left(\sigma - \sigma_{0j,i} \right)^2 \right\}. \tag{A18}$$

Defining the argument of (A18) as

$$L\left(\sigma, \sigma_{0j,i}\right) = \lambda_\gamma f_\gamma(\sigma) + \frac{1}{2\mu} \left(\sigma - \sigma_{0j,i} \right)^2, \tag{A19}$$

taking its derivative with respect to σ and setting it to zero yields after short manipulation

$$\frac{\sigma - \sigma_{0j,i}}{\mu} = -\frac{\lambda_\gamma}{\gamma} \frac{\sigma}{|\sigma|} \exp\left(-\frac{\sigma}{\gamma} \right) = -\frac{\lambda_\gamma}{\gamma} \exp\left(-\frac{\sigma}{\gamma} \right), \tag{A20}$$

where we used the fact that $\sigma \geq 0$. To apply the Lambert W function we modify (A20) to

$$\frac{\sigma - \sigma_{0j,i}}{\gamma} \exp\left(\frac{\sigma - \sigma_{0j,i}}{\gamma} \right) = -\frac{\lambda_\gamma \mu}{\gamma^2} \exp\left(-\frac{\sigma_{0j,i}}{\gamma} \right). \tag{A21}$$

Applying the Lambert W function on both sides of (A21) gives two solutions

$$\sigma_1 = \gamma W_0(z) + \sigma_{0j,i} \tag{A22}$$

$$\sigma_2 = \gamma W_{-1}(z) + \sigma_{0j,i}, \tag{A23}$$

where

$$z = -\frac{\lambda_\gamma \mu}{\gamma^2} \exp\left(-\frac{\sigma_{0j,i}}{\gamma} \right) \tag{A24}$$

and $W_0(\cdot)$ denotes the upper branch and $W_{-1}(\cdot)$ the lower branch of the multi-valued Lambert W function. As shown in [63], it can be proven that σ_2 from (A23) cannot be the minimizer of (A18). The rest of the derivation, which establishes conditions under which σ_1 is the true minimizer of (A18), is left to look up in [63]. In consequence, the solution to (A18) is the shrinkage operator

$$\mathcal{T}^{(\gamma)}_{\mu\lambda_\gamma}(x) = \begin{cases} 0 & x < \gamma\left(1 + \ln\left(\frac{\mu\lambda_\gamma}{\gamma^2}\right)\right) \\ 0 & L(0,x) < L(\sigma_1, x) \\ \sigma_1 & \text{otherwise} \end{cases}, \tag{A25}$$

where σ_1 is defined in (A22) and $L(\cdot,\cdot)$ in (A19). The solution to (A12) is therefore

$$\boldsymbol{L}_{j+1} = \boldsymbol{U}_{0j} \operatorname{diag}\left(\mathcal{T}^{(\gamma)}_{\mu\lambda_\gamma}(\boldsymbol{\sigma}_{0j}) \right) \boldsymbol{V}_{0j}^{\mathrm{H}},$$

where $\mathcal{T}^{(\gamma)}_{\mu\lambda_\gamma}(\sigma_{0j}) = \left[\mathcal{T}^{(\gamma)}_{\mu\lambda_\gamma}(\sigma_{0j,1}) \quad \cdots \quad \mathcal{T}^{(\gamma)}_{\mu\lambda_\gamma}(\sigma_{0j,n_{\min}})\right]^{\mathrm{T}}$ is the vector operator version of (A25), $\boldsymbol{\sigma}_{0j} = \boldsymbol{\sigma}(\boldsymbol{L}_{0j})$, $\boldsymbol{L}_{0j}$ is (A13), and $\boldsymbol{U}_{0j}$ and $\boldsymbol{V}^{\mathrm{H}}_{0j}$ are defined in (A15).

The structure of the CSRA algorithm is similar to the SCSA algorithm presented in [63], which was designed for real valued CS problems. Hence, the convergence proof given in [63] can readily be adapted to the CSRA algorithm and is therefore not recapitulated here. Only the following theorem stating the convergence shall be given, in which h denotes the data fidelity term defined in (5).

Theorem A3. *Let $M > 0$ denote the smallest Lipschitz constant of ∇h and M' the smallest constant such that for all $\boldsymbol{X} \in \mathbb{C}^{N_1 \times N_2}$ and $\boldsymbol{Y} \in \mathbb{C}^{N_1 \times N_2}$*

$$\begin{aligned}\lambda F_\gamma(\boldsymbol{X}) \leq \lambda F_\gamma(\boldsymbol{Y}) + \mathrm{Re}\left\{\langle \boldsymbol{X}-\boldsymbol{Y}, \nabla \lambda F_\gamma(\boldsymbol{X})\rangle_{\mathrm{F}}\right\} \\ + \frac{M'}{2}\|\boldsymbol{X}-\boldsymbol{Y}\|^2_{\mathrm{F}}.\end{aligned}$$

For any step size $\mu \in (0, 1/(M+M'))$, the sequence $\{\boldsymbol{L}_j\}$ generated by (A12) converges to a stationary point of (A10).

The proof of Theorem A3 is equivalent to the proof given in [63] when replacing all ℓ_2 norms and inner products with the Frobenius norm and Frobenius product. The Lipschitz constant of ∇h is given by the squared operator norm $M = \|\mathcal{A}\|^2_2$. However, a formal proof of the existence of M', which would need to fulfill

$$\|\nabla \lambda F_\gamma(\boldsymbol{X}) - \nabla \lambda F_\gamma(\boldsymbol{Y})\|_{\mathrm{F}} \leq M' \|\boldsymbol{X}-\boldsymbol{Y}\|_{\mathrm{F}},$$

is open. Nevertheless, a step size of $\mu \in (0, 1/M)$ resulted in converging behavior in all conducted simulations. An alternative proof of convergence is given in [64], however for a decreasing step size.

Finally, a few words on the initialization of γ are in order. Let $\widehat{\boldsymbol{L}}$ be the unique solution to

$$\min_{\boldsymbol{L}} \|\boldsymbol{L}\|_* \text{ subject to } \mathcal{A}(\boldsymbol{L}) = \boldsymbol{y}, \tag{A26}$$

which is the equivalent nuclear norm minimization (NNM) noiseless optimization problem to (17). In [62] it was shown, that for $\gamma \to \infty$, the following statement holds

$$\begin{aligned}\lim_{\gamma\to\infty} \arg\min_{\boldsymbol{L}} \{ F_\gamma(\boldsymbol{L}) | \mathcal{A}(\boldsymbol{L}) = \boldsymbol{y}\} = \\ = \arg\min_{\boldsymbol{L}} \{ \|\boldsymbol{L}\|_* | \mathcal{A}(\boldsymbol{L}) = \boldsymbol{y}\} = \widehat{\boldsymbol{L}},\end{aligned} \tag{A27}$$

provided that (A26) has a unique solution. Therefore, (60), for $\gamma \to \infty$, can be optimized by solving (17) for which SVT or TSVT may be used. According to [62], $\gamma_0 = 8\|\boldsymbol{L}_0\|_2$ is a reasonable choice such that (A27) approximately holds. For CSRA, $\gamma_0 = \|\boldsymbol{L}_0\|_2/10$ was found to work well in every case.

References

1. Ender, J.H. On compressive sensing applied to radar. *Signal Process.* **2010**, *90*, 1402–1414. [CrossRef]
2. Weng, Z.; Wang, X. Low-rank matrix completion for array signal processing. In Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 25–30 March 2012; pp. 2697–2700. [CrossRef]
3. Ender, J. A brief review of compressive sensing applied to radar. In Proceedings of the 2013 14th International Radar Symposium (IRS), Dresden, Germany, 19–21 June 2013; Volume 1, pp. 3–16.
4. de Lamare, R.C. Low-Rank Signal Processing: Design, Algorithms for Dimensionality Reduction and Applications. *arXiv* **2015**, arxiv:1508.00636.
5. Sun, S.; Mishra, K.V.; Petropulu, A.P. Target Estimation by Exploiting Low Rank Structure in Widely Separated MIMO Radar. In Proceedings of the 2019 IEEE Radar Conference (RadarConf), Boston, MA, USA, 22–26 April 2019; pp. 1–6. [CrossRef]
6. Xiang, Y.; Xi, F.; Chen, S. LiQuiD-MIMO Radar: Distributed MIMO Radar with Low-Bit Quantization. *arXiv* **2023**.

7. Rangaswamy, M.; Lin, F. Radar applications of low rank signal processing methods. In Proceedings of the Thirty-Sixth Southeastern Symposium on System Theory, Atlanta, GA, USA, 16 March 2004; pp. 107–111. [CrossRef]
8. Prünte, L. GMTI on short sequences of pulses with compressed sensing. In Proceedings of the 2015 3rd International Workshop on Compressed Sensing Theory and Its Applications to Radar, Sonar and Remote Sensing (CoSeRa), Pisa, Italy, 17–19 June 2015; pp. 66–70. [CrossRef]
9. Sen, S. Low-Rank Matrix Decomposition and Spatio-Temporal Sparse Recovery for STAP Radar. *IEEE J. Sel. Top. Signal Process.* **2015**, *9*, 1510–1523. [CrossRef]
10. Prünte, L. Compressed sensing for the detection of moving targets from short sequences of pulses: Special section "sparse reconstruction in remote sensing". In Proceedings of the 2016 4th International Workshop on Compressed Sensing Theory and Its Applications to Radar, Sonar and Remote Sensing (CoSeRa), Aachen, Germany, 19–22 September 2016; pp. 85–89. [CrossRef]
11. Prünte, L. Detection of Moving Targets Using Off-Grid Compressed Sensing. In Proceedings of the 2018 19th International Radar Symposium (IRS), Bonn, Germany, 20–22 June 2018; pp. 1–10. [CrossRef]
12. Dao, M.; Nguyen, L.; Tran, T.D. Temporal rate up-conversion of synthetic aperture radar via low-rank matrix recovery. In Proceedings of the 2013 IEEE International Conference on Image Processing, Melbourne, VIC, Australia, 15–18 September 2013; pp. 2358–2362. [CrossRef]
13. Cerutti-Maori, D.; Prünte, L.; Sikaneta, I.; Ender, J. High-resolution wide-swath SAR processing with compressed sensing. In Proceedings of the 2014 IEEE Geoscience and Remote Sensing Symposium, Quebec City, QC, Canada, 13–18 July 2014; pp. 3830–3833. [CrossRef]
14. Mason, E.; Son, I.-Y.; Yazici, B. Passive synthetic aperture radar imaging based on low-rank matrix recovery. In Proceedings of the 2015 IEEE Radar Conference (RadarCon), Arlington, VA, USA, 10–15 May 2015; pp. 1559–1563.
15. Kang, J.; Wang, Y.; Schmitt, M.; Zhu, X.X. Object-Based Multipass InSAR via Robust Low-Rank Tensor Decomposition. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 3062–3077. [CrossRef]
16. Hamad, A.; Ender, J. Three Dimensional ISAR Autofocus based on Sparsity Driven Motion Estimation. In Proceedings of the 2020 21st International Radar Symposium (IRS), Warsaw, Poland, 5–8 October 2020; pp. 51–56. [CrossRef]
17. Qiu, W.; Zhou, J.; Fu, Q. Jointly Using Low-Rank and Sparsity Priors for Sparse Inverse Synthetic Aperture Radar Imaging. *IEEE Trans. Image Process.* **2020**, *29*, 100–115. [CrossRef]
18. Wagner, S.; Ender, J. Scattering Identification in ISAR Images via Sparse Decomposition. In Proceedings of the 2022 IEEE Radar Conference (RadarConf22), New York, NY, USA, 21–25 March 2022; pp. 1–6. [CrossRef]
19. Tang, V.H.; Bouzerdoum, A.; Phung, S.L.; Tivive, F.H.C. Radar imaging of stationary indoor targets using joint low-rank and sparsity constraints. In Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, 20–25 March 2016; pp. 1412–1416. [CrossRef]
20. Sun, Y.; Breloy, A.; Babu, P.; Palomar, D.P.; Pascal, F.; Ginolhac, G. Low-Complexity Algorithms for Low Rank Clutter Parameters Estimation in Radar Systems. *IEEE Trans. Signal Process.* **2016**, *64*, 1986–1998. [CrossRef]
21. Wang, J.; Ding, M.; Yarovoy, A. Interference Mitigation for FMCW Radar with Sparse and Low-Rank Hankel Matrix Decomposition. *IEEE Trans. Signal Process.* **2022**, *70*, 822–834. [CrossRef]
22. Brehier, H.; Breloy, A.; Ren, C.; Hinostroza, I.; Ginolhac, G. Robust PCA for Through-the-Wall Radar Imaging. In Proceedings of the 2022 30th European Signal Processing Conference (EUSIPCO), Belgrade, Serbia, 29 August–2 September 2022; pp. 2246–2250. [CrossRef]
23. Yang, D.; Yang, X.; Liao, G.; Zhu, S. Strong Clutter Suppression via RPCA in Multichannel SAR/GMTI System. *IEEE Geosci. Remote Sens. Lett.* **2015**, *12*, 2237–2241. [CrossRef]
24. Guo, Y.; Liao, G.; Li, J.; Chen, X. A Novel Moving Target Detection Method Based on RPCA for SAR Systems. *IEEE Trans. Geosci. Remote Sens.* **2020**, *58*, 6677–6690. [CrossRef]
25. A Clutter Suppression Method Based on NSS-RPCA in Heterogeneous Environments for SAR-GMTI. *IEEE Trans. Geosci. Remote Sens.* **2020**, *58*, 5880–5891. [CrossRef]
26. Yang, J.; Jin, T.; Xiao, C.; Huang, X. Compressed Sensing Radar Imaging: Fundamentals, Challenges, and Advances. *Sensors* **2019**, *19*, 3100. [CrossRef]
27. Zuo, L.; Wang, J.; Zhao, T.; Cheng, Z. A Joint Low-Rank and Sparse Method for Reference Signal Purification in DTMB-Based Passive Bistatic Radar. *Sensors* **2021**, *21*, 3607. [CrossRef] [PubMed]
28. De Maio, A.; Eldar, Y.; Haimovich, A. *Compressed Sensing in Radar Signal Processing*; Cambridge University Press: Cambridge, UK, 2019.
29. Amin, M. *Compressive Sensing for Urban Radar*; CRC Press: Boca Raton, FL, USA, 2017.
30. Manchanda, R.; Sharma, K. A Review of Reconstruction Algorithms in Compressive Sensing. In Proceedings of the 2020 International Conference on Advances in Computing, Communication Materials (ICACCM), Dehradun, India, 21–22 August 2020; pp. 322–325. [CrossRef]
31. Cai, J.F.; Candès, E.J.; Shen, Z. A Singular Value Thresholding Algorithm for Matrix Completion. *arXiv* **2008**.
32. Candès, E.J.; Li, X.; Ma, Y.; Wright, J. Robust Principal Component Analysis? *CoRR* **2009**, abs/0912.3599.
33. Eldar, Y.; Kutyniok, G. *Compressed Sensing: Theory and Applications*; Cambridge University Press: Cambridge, UK, 2012.
34. Pilastri, A.; Tavares, J. Reconstruction Algorithms in Compressive Sensing: An Overview. In Proceedings of the FAUP-11th edition of the Doctoral Symposium in Informatics Engineering, Porto, Portugal, 3 February 2016.

35. Park, D.; Kyrillidis, A.; Caramanis, C.; Sanghavi, S. Finding Low-Rank Solutions via Non-Convex Matrix Factorization, Efficiently and Provably. *arXiv* **2016**, arXiv:1606.03168.
36. Chandrasekaran, V.; Sanghavi, S.; Parrilo, P.A.; Willsky, A.S. Rank-Sparsity Incoherence for Matrix Decomposition. *SIAM J. Optim.* **2011**, *21*, 572–596. [CrossRef]
37. Tropp, J.A.; Gilbert, A.C. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans. Inf. Theory* **2007**, *53*, 4655–4666. [CrossRef]
38. Donoho, D.L.; Tsaig, Y.; Drori, I.; Starck, J.L. Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit. *IEEE Trans. Inf. Theory* **2012**, *58*, 1094–1121. [CrossRef]
39. Needell, D.; Vershynin, R. Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit. *IEEE J. Sel. Top. Signal Process.* **2010**, *4*, 310–316. [CrossRef]
40. Needell, D.; Tropp, J.A. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Appl. Comput. Harmon. Anal.* **2009**, *26*, 301–321. [CrossRef]
41. Dai, W.; Milenkovic, O. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Trans. Inf. Theory* **2009**, *55*, 2230–2249. [CrossRef]
42. Boche, H.; Calderbank, R.; Kutyniok, G.; Vybiral, J. *A Survey of Compressed Sensing*; Springer: Berlin/Heidelberg, Germany, 2014. [CrossRef]
43. Lee, K.; Bresler, Y. ADMiRA: Atomic Decomposition for Minimum Rank Approximation. *IEEE Trans. Inf. Theory* **2010**, *56*, 4402–4416. [CrossRef]
44. Waters, A.; Sankaranarayanan, A.; Baraniuk, R. SpaRCS: Recovering low-rank and sparse matrices from compressive measurements. In *Proceedings of the Advances in Neural Information Processing Systems*; Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2011; Volume 24.
45. Xiang, J.; Yue, H.; Xiangjun, Y.; Guoqing, R. A Reweighted Symmetric Smoothed Function Approximating L0-Norm Regularized Sparse Reconstruction Method. *Symmetry* **2018**, *10*, 583. [CrossRef]
46. Xiang, J.; Yue, H.; Xiangjun, Y.; Wang, L. A New Smoothed L0 Regularization Approach for Sparse Signal Recovery. *Math. Probl. Eng.* **2019**, *2019*, 1978154. [CrossRef]
47. Blumensath, T.; Davies, M.E. Normalized Iterative Hard Thresholding: Guaranteed Stability and Performance. *IEEE J. Sel. Top. Signal Process.* **2010**, *4*, 298–309. [CrossRef]
48. Meka, R.; Jain, P.; Dhillon, I.S. Guaranteed Rank Minimization via Singular Value Projection. *arXiv* **2009**.
49. Zhang, X.; Wang, L.; Gu, Q. A Unified Framework for Low-Rank plus Sparse Matrix Recovery. *arXiv* **2017**, arxiv:1702.06525.
50. Blanchard, J.D.; Tanner, J. Performance comparisons of greedy algorithms in compressed sensing. *Numer. Linear Algebra Appl.* **2015**, *22*, 254–282. [CrossRef]
51. Mansour, H. Beyond ℓ_1-norm minimization for sparse signal recovery. In Proceedings of the 2012 IEEE Statistical Signal Processing Workshop (SSP), Ann Arbor, MI, USA, 5–8 August 2012; pp. 337–340. [CrossRef]
52. Aravkin, A.; Becker, S.; Cevher, V.; Olsen, P. A variational approach to stable principal component pursuit. *arXiv* **2014**, arxiv:1406.1089.
53. Beck, A.; Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.* **2009**, *2*, 183–202. [CrossRef]
54. Panhuber, R.; Prünte, L. Complex Successive Concave Sparsity Approximation. In Proceedings of the 2020 21st International Radar Symposium (IRS), Warsaw, Poland, 5–8 October 2020; pp. 67–72. [CrossRef]
55. Ma, J.; Yuan, X.; Ping, L. Turbo Compressed Sensing with Partial DFT Sensing Matrix. *IEEE Signal Process. Lett.* **2015**, *22*, 158–161. [CrossRef]
56. Xue, Z.; Ma, J.; Yuan, X. Denoising-Based Turbo Compressed Sensing. *IEEE Access* **2017**, *5*, 7193–7204. [CrossRef]
57. Xue, Z.; Yuan, X.; Ma, J.; Ma, Y. TARM: A Turbo-Type Algorithm for Affine Rank Minimization. *IEEE Trans. Signal Process.* **2019**, *67*, 5730–5745. [CrossRef]
58. Xue, Z.; Yuan, X.; Yang, Y. Turbo-Type Message Passing Algorithms for Compressed Robust Principal Component Analysis. *IEEE J. Sel. Top. Signal Process.* **2018**, *12*, 1182–1196. [CrossRef]
59. He, X.; Xue, Z.; Yuan, X. Learned Turbo Message Passing for Affine Rank Minimization and Compressed Robust Principal Component Analysis. *IEEE Access* **2019**, *7*, 140606–140617. [CrossRef]
60. Kang, Z.; Peng, C.; Cheng, J.; Cheng, Q. LogDet Rank Minimization with Application to Subspace Clustering. *Comput. Intell. Neurosci.* **2015**, *2015*, 824289. [CrossRef] [PubMed]
61. Malek-Mohammadi, M.; Babaie-Zadeh, M.; Amini, A.; Jutten, C. Recovery of Low-Rank Matrices Under Affine Constraints via a Smoothed Rank Function. *IEEE Trans. Signal Process.* **2014**, *62*, 981–992. [CrossRef]
62. Malek-Mohammadi, M.; Babaie-Zadeh, M.; Skoglund, M. Iterative Concave Rank Approximation for Recovering Low-Rank Matrices. *IEEE Trans. Signal Process.* **2014**, *62*, 5213–5226. [CrossRef]
63. Malek-Mohammadi, M.; Koochakzadeh, A.; Babaie-Zadeh, M.; Jansson, M.; Rojas, C. Successive Concave Sparsity Approximation for Compressed Sensing. *IEEE Trans. Signal Process.* **2016**, *64*, 5657–5671. [CrossRef]
64. Ye, H.; Li, H.; Yang, B.; Cao, F.; Tang, Y. A Novel Rank Approximation Method for Mixture Noise Removal of Hyperspectral Images. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 4457–4469. [CrossRef]
65. Bickel, P.J.; Ritov, Y.; Tsybakov, A.B. Simultaneous analysis of Lasso and Dantzig selector. *arXiv* **2008**, arXiv:0801.1095.

66. Donoho, D.; Tanner, J. Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing. *Philos. Trans. R. Soc. Math. Phys. Eng. Sci.* **2009**, *367*, 4273–4293. [CrossRef]
67. Candès, E.J.; Sing-Long, C.A.; Trzasko, J.D. Unbiased Risk Estimates for Singular Value Thresholding and Spectral Estimators. *IEEE Trans. Signal Process.* **2013**, *61*, 4643–4657. [CrossRef]
68. Bouwmans, T.; Sobral, A.; Javed, S.; Jung, S.K.; Zahzah, E.H. Decomposition into low-rank plus additive matrices for background/foreground separation: A review for a comparative evaluation with a large-scale dataset. *Comput. Sci. Rev.* **2017**, *23*, 1–71. [CrossRef]
69. Foucart, S.; Rauhut, H. *A Mathematical Introduction to Compressive Sensing*; Springer: Berlin/Heidelberg, Germany, 2013. [CrossRef]
70. Zill, D.; Wright, W. *Differential Equations with Boundary-Value Problems*; Cengage Learning: Boston, MA, USA, 2012.