USC-SIPI Report #443

LORAKS Software Version 2.0: Faster Implementation and Enhanced Capabilities

By

Tae Hyung Kim and Justin P. Haldar

May 2018

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
USC Viterbi School of Engineering
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Suite 400
Los Angeles, CA 90089-2564 U.S.A.

LORAKS Software Version 2.0: Faster Implementation and Enhanced Capabilities

Tae Hyung Kim and Justin P. Haldar*

Contents

*The authors are with the Signal and Image Processing Institute and the Ming Hsieh Department of Electrical Engineering at the University of Southern California, Los Angeles, CA, 90089, USA.

1 Overview

Over the past several years, our research group has been developing a novel structured low-rank matrix modeling framework for magnetic resonance (MR) image reconstruction that we call LORAKS (LOw-RAnk modeling of local K-Space neighborhoods) [1–12].[1] In the spirit of reproducible research [18], we had previously released a public open-source software implementation of LORAKS-based image reconstruction in 2014 [3]. In the present technical report (and supplementary material available for download at `http://mr.usc.edu/download/LORAKS2/`), we describe an updated public open-source software release that provides access to many of the new developments we've made since 2014, including substantially-faster algorithms and a variety of new formulations of the inverse problem [4–12].

The LORAKS framework is based on the assumption that ideal uniformly-sampled Fourier data will often possess a multitude of distinct shift-invariant linear predictability relationships. Linear predictability is a powerful constraint, because if a sample can be predicted accurately from its neighbors, then there's no need to actually measure that sample. Linear predictability also arises naturally as a consequence of several different kinds of classical and widely-used image reconstruction constraints. For example, shift-invariant linear predictability can be proven in scenarios for which the original image has limited support [19], slowly-varying phase [2, 20], inter-channel correlations in a multi-channel acquisition [21, 22], and/or sparsity in an appropriate transform domain [13].

An important feature of linear predictability is that high-dimensional convolution-structured matrices (e.g., Hankel and/or Toeplitz matrices) formed from linearly-predictable data will often possess low-rank structure [13], which in turn implies that modern low-rank matrix recovery methods [23] can be used to improve the recovery of linearly-predictable signals from noisy and/or incomplete data. When applied to MRI data, this type of structured low-rank matrix recovery approach has enabled impressive performance across a range of challenging scenarios [1–12, 15–17].

Relative to traditional constrained reconstruction, a unique advantage of structured low-rank matrix recovery approaches is that they are very flexible and adaptable to different situations. Specifically, classical constrained MR approaches often require (i) a substantial amount of prior information and (ii) a reconstruction procedure that is specially adapted to enforce the existence a specific kind of image structure [24]. In contrast, LORAKS can be used successfully even when it's unclear whether a specific constraint will be applicable to a given dataset. Rather than assuming that certain constraints are applicable in advance, LORAKS uses the subspace structure of the raw data to automatically identify and enforce any and all linear prediction relationships that may exist in the data. This approach is agnostic to the original source of the linear prediction relationship, and there is no need for the user to know in advance what kind of constraint may be applicable to a given image. This flexibility and generality allows LORAKS to transition seamlessly between different k-space sampling patterns (i.e., both calibrationless and calibration-based sampling; random, variable density, partial Fourier, and highly-unconventional sampling patterns), imaging geometries (i.e., both single-channel and multi-channel), and application contexts.

2 Mathematical Description of Reconstruction Approaches Provided by the Software

2.1 Problem Formulations

2.1.1 Original Single-Channel Formulation

Our original description and software implementation of LORAKS [2, 3] gave users the ability to solve regularized single-channel MRI reconstruction problems using up to three different LORAKS matrix constructions according to:

$$\hat{\mathbf{f}} = \arg\min_{\mathbf{f}} \|\mathbf{A}\mathbf{f} - \mathbf{d}\|_{\ell_2}^2 + \lambda_{\mathbf{C}} J_{r_{\mathbf{C}}}(\mathcal{P}_{\mathbf{C}}(\mathbf{f})) + \lambda_{\mathbf{G}} J_{r_{\mathbf{G}}}(\mathcal{P}_{\mathbf{G}}(\mathbf{f})) + \lambda_{\mathbf{S}} J_{r_{\mathbf{S}}}(\mathcal{P}_{\mathbf{S}}(\mathbf{f})). \tag{1}$$

In this expression, $\mathbf{f}$ is a vector of uniformly-sampled Cartesian k-space data to be estimated, $\mathbf{d}$ is the vector of measured data, and the matrix $\mathbf{A}$ is a sampling operator that models the conversion from full k-space data to subsampled

[1] Related methods have also been explored by other groups, e.g., [13–17]. However, for the sake of brevity, we will restrict our attention to the perspectives and terminology from our previous LORAKS work.

k-space data. In addition, $\mathcal{P}_{\mathbf{C}}(\cdot)$, $\mathcal{P}_{\mathbf{G}}(\cdot)$, and $\mathcal{P}_{\mathbf{S}}(\cdot)$ are operators that construct high-dimensional structured LORAKS matrices (respectively called the $\mathbf{C}$, $\mathbf{G}$, and $\mathbf{S}$ matrices); $\lambda_{\mathbf{C}}$, $\lambda_{\mathbf{G}}$, and $\lambda_{\mathbf{S}}$ are user-selected regularization parameters used to adjust the strength of the regularization penalty applied to each matrix; $r_{\mathbf{C}}$, $r_{\mathbf{G}}$, and $r_{\mathbf{S}}$ are user-selected rank estimates for the $\mathbf{C}$, $\mathbf{G}$, and $\mathbf{S}$ matrices, respectively; and $J_r(\cdot)$ is a nonconvex regularization penalty function that encourages its matrix argument to have rank less than or equal to r. Specifically, this function is equal to the Frobenius norm of the residual obtained after an optimal rank-r approximation of its matrix argument. In particular, assuming that $\mathbf{X} \in \mathbb{C}^{P \times Q}$, we have that

$$J_r(\mathbf{X}) = \sum_{k>r} \sigma_k^2 \tag{2}$$

$$= \min_{\mathbf{T} \in \mathbb{C}^{P \times Q}} \|\mathbf{X} - \mathbf{T}\|_F^2 \ \text{ s.t. } \ \text{rank}(\mathbf{T}) \leq r \tag{3}$$

$$= \min_{\mathbf{V} \in \mathbb{C}^{Q \times (Q-r)}} \|\mathbf{X}\mathbf{V}\|_F^2 \ \text{ s.t. } \ \mathbf{V}^H\mathbf{V} = \mathbf{I}_{(Q-r)}, \tag{4}$$

where σ_k is the kth singular value of the matrix $\mathbf{X}$, and $\mathbf{I}_{(Q-r)}$ is the $(Q-r) \times (Q-r)$ identity matrix.

Theoretically, the $\mathbf{C}$ matrix is expected to have low-rank whenever the image obeys a support constraint, the $\mathbf{G}$ matrix is expected to have low-rank whenever the image obeys a support constraint and possesses smoothly-varying phase, and the $\mathbf{S}$ matrix is expected to have low-rank whenever the image obeys a support constraint and/or possesses smoothly-varying phase. See Ref. [2] for further details. Additional LORAKS matrix constructors, which form a structured LORAKS matrix after applying a linear transformation to $\mathbf{f}$ were introduced in [6], and will have low-rank whenever the image is sparse in an appropriate transform domain.

The present software implementation still provides the ability to solve Eq. (1), but additionally includes a matrix constructor $\mathcal{P}_{\mathbf{W}}$ that generates what we call the $\mathbf{W}$-matrix [9]. The $\mathbf{W}$ matrix will have low rank if the spatial derivatives of the original image are sparse [6, 9, 13, 16, 17].

Note that the present software implementation no longer supports the $\mathbf{G}$ matrix, since previous literature has consistently demonstrated (e.g., [2]) that this LORAKS matrix is less powerful than the alternatives. In addition, we only provide support to use one matrix at a time rather than using multiple matrices simultaneously – it is possible to get small performance improvements by using multiple matrices simultaneously, although this comes at the expense of computational complexity and the need to tune a larger number of reconstruction parameters. Specifically, for the single-channel case, the present software provides capabilities to solve

$$\hat{\mathbf{f}} = \arg\min_{\mathbf{f}} \|\mathbf{A}\mathbf{f} - \mathbf{d}\|_{\ell_2}^2 + \lambda J_r(\mathcal{P}_{\mathbf{X}}(\mathbf{f})), \tag{5}$$

with $\mathbf{X} \in \{\mathbf{C}, \mathbf{S}, \mathbf{W}\}$.

Our software assumes a certain kind of Cartesian sampling by default (meaning that $\mathbf{A}$ is formed by taking a subset of rows from the identity matrix, which implies that the samples observed in $\mathbf{d}$ are obtained at a subset of the same grid of k-space locations to be reconstructed in $\mathbf{f}$) , similar to our previous software release [3]. This assumption implies that $\mathbf{A}^H\mathbf{A}$ has simple structure (i.e., it is a diagonal matrix) that enables substantial computational simplifications, as will be described later. Since our code is open-source, it would be straightforward to modify the code to use non-Cartesian/off-grid sampling if so desired. This would amount to modeling $\mathbf{A}$ as a gridding operator that interpolates from the original Cartesian grid onto the off-grid sample locations [25], and then solving the resulting optimization steps (to be described in the sequel) directly without assuming that $\mathbf{A}^H\mathbf{A}$ is diagonal.

In our current implementation, we also now support the use of virtual conjugate coils [26, 27], which enables the use of smoothly varying phase constraints with the $\mathbf{C}$ matrix [9, 12].

This software is currently implemented for 2D data reconstruction. However, extension to 3D reconstruction is straightforward [12].

2.1.2 Enforcing Exact Data Consistency

A potential disadvantage of Eq. (5) is the need for the user to choose the regularization parameter λ. The regularization parameter represents a trade-off between reliance on the measured data and reliance on the constraint, and selection of regularization parameters is an omnipresent problem whenever dealing with regularized reconstruction problems.

However, in many circumstances with high signal-to-noise ratio, a user may be interested in choosing λ such that the reconstruction is as consistent with the measured data as possible, so that the LORAKS constraints are just used to interpolate missing data without modifying the measured data samples. In the context of Eq. (5), this corresponds to choosing very small values of λ [2]. However, due to finite precision numerical effects, it is not possible to choose λ values that are arbitrarily small. A way of bypassing this issue is to instead solve the following data-consistency-constrained optimization problem [4], which no longer involves a regularization parameter:

$$\hat{\mathbf{f}} = \arg\min_{\mathbf{f}} J_r(\mathcal{P}_{\mathbf{X}}(\mathbf{f})) \;\; \text{s.t.} \;\; \mathbf{A}\mathbf{f} = \mathbf{d}. \tag{6}$$

The constraint in this problem is easy to enforce when the k-space sampling locations used for $\mathbf{d}$ are a subset of the k-space sampling locations used in $\mathbf{f}$ (i.e., the on-grid Cartesian sampling scenario described in Sec. 2.1.1). In particular, we can enforce the constraint in this case by only solving for the entries from $\mathbf{f}$ that are not present in the measured data $\mathbf{d}$, and otherwise directly setting the entries of $\mathbf{f}$ equal to the measured data. This can be achieved by solving [4]

$$\hat{\mathbf{f}} = \mathcal{A}(\mathbf{d}) + \mathbf{M}\hat{\mathbf{z}}, \tag{7}$$

with

$$\hat{\mathbf{z}} = \arg\min_{\mathbf{z}} J_r(\mathcal{P}_{\mathbf{X}}(\mathbf{M}\mathbf{z}) - (-\mathcal{P}_{\mathbf{X}}(\mathcal{A}(\mathbf{d})))), \tag{8}$$

where $\mathcal{A}(\mathbf{d})$ is the zero-filled vector that is the same size as $\mathbf{f}$ and contains the samples from $\mathbf{d}$ in the appropriate k-space locations, $\mathbf{z}$ is the vector of unmeasured data samples, and the matrix $\mathbf{M}$ places the unmeasured data samples in their appropriate locations while zero-filling the locations corresponding to acquired data samples.

2.1.3 Multi-Channel Formulations: P-LORAKS and SENSE-LORAKS

Beyond the single-channel case, subsequent work has developed multi-channel parallel imaging versions of the $\mathbf{C}$, $\mathbf{G}$, and $\mathbf{S}$ matrices, which are obtained by concatenating together the corresponding LORAKS matrices constructed from different channels [7, 8]. These concatenated matrices will have low-rank in the same scenarios as for the single-channel case (i.e., in the presence of support, phase constraints, and/or sparsity constraints), but will have additional low-rank structure due to the correlations that will exist between channels in a multi-channel imaging experiment [7, 14, 15, 17].

One of these multi-channel LORAKS approaches, called P-LORAKS [7], simply solves Eq. (5) using this generalized concatenation-based definition of the LORAKS matrices. Due to the fact that the P-LORAKS matrices reduce to the single-channel LORAKS matrices in the special case of single-channel data, we do not distinguish notation between the single-channel LORAKS and the P-LORAKS formulations of this problem, and use Eq. (5) to represent both variations. Note that using Eq. (5) with the multichannel $\mathbf{C}$ matrix is very similar to SAKE [15] – in this case, there are only minor differences between SAKE and P-LORAKS related to the shape of the LORAKS neighborhood system and the choice of cost functional. Note that it is also possible to use the data-consistency constraint as described in Sec. 2.1.2 with both single-channel and multi-channel data.

The other approach, called SENSE-LORAKS [8], combines P-LORAKS with SENSE [28,29]. In this case, instead of solving for the fully-sampled k-space data of all the coils $\mathbf{f}$, we instead use prior knowledge of the coil sensitivity profiles solve for a single image vector $\boldsymbol{\rho}$ according to

$$\hat{\boldsymbol{\rho}} = \arg\min_{\boldsymbol{\rho}} \|\mathbf{E}\boldsymbol{\rho} - \mathbf{d}\|_{\ell_2}^2 + \lambda J_r(\mathcal{P}_{\mathbf{X}}(\mathbf{F}\boldsymbol{\rho})), \tag{9}$$

with $\mathbf{X} \in \{\mathbf{C}, \mathbf{S}, \mathbf{W}\}$, where the matrix $\mathbf{E}$ is the SENSE model for how the measured data is related to the desired image (which, for each channel, includes weighting of the image by the sensitivity profile of the coil, followed by Fourier transformation and sampling at the k-space locations corresponding to those in $\mathbf{d}$ [29]), and the matrix $\mathbf{F}$ converts the image into fully-sampled k-space data for each coil (which, for each channel, includes weighting of the image by the sensitivity profile of the coil, followed by Fourier transformation and sampling at the k-space locations corresponding to those in $\mathbf{f}$) as needed for constructing the multi-channel LORAKS matrices.

2.1.4 Autocalibrated LORAKS

The problem formulations listed above are all nonconvex and can be relatively computionally-demanding to solve. The Autocalibrated LORAKS (AC-LORAKS) approach [4] recognized that substantial improvements in computational efficiency may be possible by using the approximation

$$J_r(\mathbf{X}) \approx \|\mathbf{X}\hat{\mathbf{V}}\|_F^2 \tag{10}$$

for an appropriate choice of the matrix $\hat{\mathbf{V}}$. This approximation is based on the representation of $J_r(\cdot)$ shown in Eq. (4), and can be used to convert all of the previously mentioned nonconvex formulations described by Eqs. (5), (6), and (9) into simple convex linear least-squares problems that can be solved quickly and efficiently.

In particular, the solution to the AC-LORAKS version of Eq. (5) is given by

$$\begin{aligned}\hat{\mathbf{f}} &= \arg\min_{\mathbf{f}} \|\mathbf{A}\mathbf{f} - \mathbf{d}\|_{\ell_2}^2 + \lambda\|\mathcal{P}_{\mathbf{X}}(\mathbf{f})\hat{\mathbf{V}}\|_F^2 \\ &= \mathcal{M}_1^{-1}\mathbf{A}^H\mathbf{d},\end{aligned} \tag{11}$$

where

$$\mathcal{M}_1(\mathbf{f}) \triangleq \mathbf{A}^H\mathbf{A}\mathbf{f} + \lambda\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathbf{f})\hat{\mathbf{V}}\hat{\mathbf{V}}^H) \tag{12}$$

and * is used to denote the adjoint; the solution to the AC-LORAKS version of Eq. (6) is given by

$$\begin{aligned}\hat{\mathbf{f}} &= \mathcal{A}(\mathbf{d}) + \mathbf{M}\arg\min_{\mathbf{z}} \|\mathcal{P}_{\mathbf{X}}(\mathbf{M}\mathbf{z})\hat{\mathbf{V}} - (-\mathcal{P}_{\mathbf{X}}(\mathcal{A}(\mathbf{d}))\hat{\mathbf{V}})\|_F^2 \\ &= \mathcal{A}(\mathbf{d}) - \mathbf{M}\mathcal{M}_2^{-1}\mathbf{M}^H\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathcal{A}(\mathbf{d}))\hat{\mathbf{V}}\hat{\mathbf{V}}^H),\end{aligned} \tag{13}$$

where

$$\mathcal{M}_2(\mathbf{z}) \triangleq \mathbf{M}^H\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathbf{M}\mathbf{z})\hat{\mathbf{V}}\hat{\mathbf{V}}^H); \tag{14}$$

and the solution to the AC-LORAKS version of Eq. (9) is given by

$$\begin{aligned}\hat{\boldsymbol{\rho}} &= \arg\min_{\boldsymbol{\rho}} \|\mathbf{E}\boldsymbol{\rho} - \mathbf{d}\|_{\ell_2}^2 + \lambda\|\mathcal{P}_{\mathbf{X}}(\mathbf{F}\boldsymbol{\rho})\hat{\mathbf{V}}\|_F^2 \\ &= \mathcal{M}_3^{-1}\mathbf{E}^H\mathbf{d},\end{aligned} \tag{15}$$

where

$$\mathcal{M}_3(\boldsymbol{\rho}) \triangleq \mathbf{E}^H\mathbf{E}\boldsymbol{\rho} + \lambda\mathbf{F}^H\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathbf{F}\boldsymbol{\rho})\hat{\mathbf{V}}\hat{\mathbf{V}}^H). \tag{16}$$

All three of these solutions can be obtained easily using standard iterative linear least-squares solvers like the conjugate gradient method or LSQR [30, 31]. As will be discussed in Sec. 2.2.3, some of the LORAKS-related computations in these expressions can be substantially accelerated using FFTs.

The AC-LORAKS approach can be viewed as a generalization of PRUNO [14], and is especially similar to PRUNO when using the multichannel $\mathbf{C}$ matrix, in which case there are only minor differences between PRUNO and AC-LORAKS related to the shape of the LORAKS neighborhood. Like PRUNO, we estimate the matrix $\hat{\mathbf{V}}$ using autocalibration (ACS) data. Specifically, if a fully-sampled region of k-space is available (commonly known in the MR literature as ACS data), then a LORAKS matrix formed from zero-filled data will contain a submatrix with fully-populated rows. Since the nullspace of the full LORAKS matrix should be included in the nullspace of this submatrix, we choose $\hat{\mathbf{V}}$ to be a basis for the nullspace of the submatrix.

2.2 Algorithm Choices

2.2.1 Original Additive Half-Quadratic Majorize-Minimize Approach

Our previous algorithm implementation [3] was based on applying a majorize-minimize (MM) algorithm [32] to Eq. (1), as originally described in Ref. [2]. Specifically, it is easy to use the representation of $J_r(\cdot)$ from Eq. (3) to show that, for a generic matrix $\mathbf{X}$ constructed by operator $\mathcal{P}_{\mathbf{X}}(\cdot)$, the following function

$$g_{\mathbf{X}}(\mathbf{f}, \hat{\mathbf{f}}^{(i-1)}) \triangleq \|\mathcal{P}_{\mathbf{X}}(\mathbf{f}) - \mathcal{L}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))\|_F^2 \tag{17}$$

is a majorant of the function $J_r(\mathcal{P}_{\mathbf{X}}(\mathbf{f}))$ at the point $\hat{\mathbf{f}}^{(i-1)}$, where the operator $\mathcal{L}_r(\cdot)$ computes the optimal rank-r approximation of its matrix argument (e.g., which is easily done using the singular value decomposition). We call this an additive half-quadratic majorizer because the majorant has structural resemblence to previous additive half-quadratic methods [33].

This majorization relationship means that we can monotonically decrease the cost function value from Eq. (5) using an additive half-quadratic MM approach in which we iteratively solve the following very simple linear least-squares problem from some initialization $\hat{\mathbf{f}}^{(0)}$:

$$\begin{aligned}\hat{\mathbf{f}}^{(i)} &= \arg\min_{\mathbf{f}} \|\mathbf{A}\mathbf{f} - \mathbf{d}\|_{\ell_2}^2 + \lambda g_{\mathbf{X}}(\mathbf{f}, \hat{\mathbf{f}}^{(i-1)}) \\ &= \mathcal{M}_4^{-1}\left(\mathbf{A}^H\mathbf{d} + \lambda\mathcal{P}_{\mathbf{X}}^*(\mathcal{L}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)})))\right),\end{aligned} \tag{18}$$

where

$$\mathcal{M}_4(\mathbf{f}) \triangleq \mathbf{A}^H\mathbf{A}\mathbf{f} + \lambda\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathbf{f})). \tag{19}$$

In the case of the right kind of Cartesian data (as defined in Sec. 2.1.1, and as assumed by our software), this specific problem can even be solved analytically because the operator $\mathcal{M}_4(\cdot)$ can be represented as a diagonal matrix, due to the special structure of the $\mathcal{P}_{\mathbf{X}}(\cdot)$ operators and the special structure of the $\mathbf{A}$ matrix. For users interested in non-Cartesian/off-grid k-space sampling, this would be possible by finding the line of code in our software that analytically computes $\hat{\mathbf{f}}^{(i)}$ based on diagonal matrix structure, and replacing it with an algorithm for solving Eq. (18) that uses standard iterative linear least-squares solvers like the conjugate gradient method or LSQR [30, 31].

Similar additive half-quadratic MM algorithms are possible for the other formulations that involve $J_r(\cdot)$, i.e., LORAKS/P-LORAKS with exact data consistency constraints as defined in Eq. (6), and SENSE-LORAKS as defined in Eq. (9). Specifically, the corresponding additive half-quadratic MM algorithm for Eq. (6) is given by the iteration

$$\begin{aligned}\hat{\mathbf{f}}^{(i)} &= \mathcal{A}(\mathbf{d}) + \mathbf{M}\arg\min_{\mathbf{z}} g_{\mathbf{X}}(\mathcal{A}(\mathbf{d}) + \mathbf{M}\mathbf{z}, \hat{\mathbf{f}}^{(i-1)}) \\ &= \mathcal{A}(\mathbf{d}) + \mathbf{M}\mathcal{M}_5^{-1}\mathbf{M}^H\left(\mathcal{P}_{\mathbf{X}}^*(\mathcal{L}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))) - \mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathcal{A}(\mathbf{d})))\right),\end{aligned} \tag{20}$$

where

$$\mathcal{M}_5(\mathbf{z}) = \mathbf{M}^H\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathbf{M}\mathbf{z})). \tag{21}$$

In this case, the matrix inverse can also be solved analytically because the operator $\mathcal{M}_5(\cdot)$ can also be represented as a diagonal matrix.

The corresponding additive half-quadratic MM algorithm for Eq. (9) is given by

$$\begin{aligned}\hat{\boldsymbol{\rho}}^{(i)} &= \arg\min_{\boldsymbol{\rho}} \|\mathbf{E}\boldsymbol{\rho} - \mathbf{d}\|_{\ell_2}^2 + \lambda g_{\mathbf{x}}(\mathbf{F}\boldsymbol{\rho}, \mathbf{F}\hat{\boldsymbol{\rho}}^{(i-1)}) \\ &= \mathcal{M}_6^{-1}\left(\mathbf{E}^H\mathbf{d} + \lambda\mathbf{F}^H\mathcal{P}_{\mathbf{X}}^*(\mathcal{L}_r(\mathcal{P}_{\mathbf{X}}(\mathbf{F}\hat{\boldsymbol{\rho}}^{(i-1)})))\right),\end{aligned} \tag{22}$$

where

$$\mathcal{M}_6(\boldsymbol{\rho}) \triangleq \mathbf{E}^H\mathbf{E}\boldsymbol{\rho} + \lambda\mathbf{F}^H\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathbf{F}\boldsymbol{\rho})). \tag{23}$$

In this case, the matrix inverse cannot be solved analytically, although fast computations are possible using algorithms like the conjugate gradient method or LSQR [30,31] because the operator $\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\cdot))$ can be represented as a diagonal matrix, and multiplications with the $\mathbf{F}$ and $\mathbf{E}$ matrices can be computed efficiently using FFTs [29].

2.2.2 Multiplicative Half-Quadratic Majorize-Minimize Approach

A different majorizer for $J_r(\cdot)$ can also be easily derived from the representation of $J_r(\cdot)$ in Eq. (4). Specifically, it is easy to show that the following function

$$h_{\mathbf{X}}(\mathbf{f}, \hat{\mathbf{f}}^{(i-1)}) \triangleq \|\mathcal{P}_{\mathbf{X}}(\mathbf{f})\mathcal{N}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))\|_F^2 \tag{24}$$

$$= \|\mathcal{P}_{\mathbf{X}}(\mathbf{f}) - \mathcal{P}_{\mathbf{X}}(\mathbf{f})\mathcal{R}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))\mathcal{R}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))^H\|_F^2, \tag{25}$$

is also a majorizer of the function $J_r(\mathcal{P}_{\mathbf{X}}(\mathbf{f}))$ at the point $\hat{\mathbf{f}}^{(i-1)}$. If we assume that $\mathbf{X} \in \mathbb{C}^{P\times Q}$, then we can define $\mathcal{N}_r(\mathbf{X})$ as the operator that constructs a $Q \times (Q-r)$ matrix whose columns are equal to the right singular vectors associated with the $(Q-r)$ smallest and/or zero singular values in the extended singular value decomposition of $\mathbf{X}$. Under the same assumptions on $\mathbf{X}$, we can define $\mathcal{R}_r(\mathbf{X})$ as the operator that constructs a $Q \times r$ matrix whose columns are equal to the right singular vectors associated with the r largest singular values in the extended singular value decomposition of $\mathbf{X}$. In other words, the columns of $\mathcal{N}_r(\mathbf{X})$ form an orthonormal basis for the $(Q-r)$-dimensional approximate nullspace of $\mathbf{X}$, while the columns of $\mathcal{R}_r(\mathbf{X})$ form an orthonormal basis for the r-dimensional approximate rowspace of $\mathbf{X}$. We call Eqs. (24) and (25) multiplicative half-quadratic majorizers because the majorant has structural resemblence to previous multiplicative half-quadratic methods [33].

Both Eqs. (24) and (25) enable the following multiplicative half-quadratic MM algorithm for solving Eq. (5) that consists of iteratively solving simple least-squares problems:

$$\begin{aligned}\hat{\mathbf{f}}^{(i)} &= \arg\min_{\mathbf{f}} \|\mathbf{A}\mathbf{f}-\mathbf{d}\|_{\ell_2}^2 + \lambda h_{\mathbf{X}}(\mathbf{f}, \hat{\mathbf{f}}^{(i-1)}) \\ &= \mathcal{M}_7^{-1}\mathbf{A}^H\mathbf{d}\end{aligned} \tag{26}$$

where

$$\begin{aligned}\mathcal{M}_7(\mathbf{f}) &\triangleq \mathbf{A}^H\mathbf{A}\mathbf{f} + \lambda\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathbf{f})\mathcal{N}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))\mathcal{N}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))^H). \\ &= \mathbf{A}^H\mathbf{A}\mathbf{f} + \lambda\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathbf{f})) - \lambda\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathbf{f})\mathcal{R}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))\mathcal{R}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))^H).\end{aligned} \tag{27}$$

Note that Eq. (24) is extremely similar to the cost function used in AC-LORAKS [4] from Eq. (10), and can be optimized in exactly the same way. We should also note the MM algorithm described in Eq. (26) using the majorant from Eq. (24) has strong similarities to an algorithm described in Ref. [34] for a slightly different cost function.

Similar to the case for the additive half-qudaratic MM algorithm, this multiplicative half-quadratic approach is easy to generalize to the other formulations that involve $J_r(\cdot)$, i.e., LORAKS/P-LORAKS with exact data consistency constraints as defined in Eq. (6), and SENSE-LORAKS as defined in Eq. (9). Specifically, the multiplicative half-quadratic MM algorithm corresponding to Eq. (6) is given by the iteration

$$\begin{aligned}\hat{\mathbf{f}}^{(i)} &= \mathcal{A}(\mathbf{d}) + \mathbf{M}\arg\min_{\mathbf{z}} h_{\mathbf{X}}(\mathcal{A}(\mathbf{d}) + \mathbf{M}\mathbf{z}, \hat{\mathbf{f}}^{(i-1)}) \\ &= \mathcal{A}(\mathbf{d}) - \mathbf{M}\mathcal{M}_8^{-1}\mathbf{M}^H\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathcal{A}(\mathbf{d}))\mathcal{N}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))\mathcal{N}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))^H) \\ &= \mathcal{A}(\mathbf{d}) + \mathbf{M}\mathcal{M}_8^{-1}\mathbf{M}^H\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathcal{A}(\mathbf{d}))\mathcal{R}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))\mathcal{R}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))^H),\end{aligned} \tag{28}$$

where

$$\begin{aligned}\mathcal{M}_8(\mathbf{z}) &\triangleq \mathbf{M}^H\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathbf{M}\mathbf{z})\mathcal{N}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))\mathcal{N}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))^H) \\ &= \mathbf{M}^H\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathbf{M}\mathbf{z})) - \mathbf{M}^H\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathbf{M}\mathbf{z})\mathcal{R}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))\mathcal{R}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))^H),\end{aligned} \tag{29}$$

and the corresponding multiplicative half-quadratic MM algorithm for Eq. (9) is given by

$$\begin{aligned}\hat{\boldsymbol{\rho}}^{(i)} &= \arg\min_{\boldsymbol{\rho}} \|\mathbf{E}\boldsymbol{\rho}-\mathbf{d}\|_{\ell_2}^2 + \lambda h_{\mathbf{x}}(\mathbf{F}\boldsymbol{\rho}, \mathbf{F}\hat{\boldsymbol{\rho}}^{(i-1)}) \\ &= \mathcal{M}_9^{-1}\mathbf{E}^H\mathbf{d},\end{aligned} \tag{30}$$

where

$$\begin{aligned}\mathcal{M}_9(\boldsymbol{\rho}) &\triangleq \mathbf{E}^H\mathbf{E}\boldsymbol{\rho} + \lambda\mathbf{F}^H\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathbf{F}\boldsymbol{\rho})\mathcal{N}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))\mathcal{N}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))^H) \\ &= \mathbf{E}^H\mathbf{E}\boldsymbol{\rho} + \lambda\mathbf{F}^H\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathbf{F}\boldsymbol{\rho})) - \lambda\mathbf{F}^H\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathbf{F}\boldsymbol{\rho})\mathcal{R}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))\mathcal{R}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))^H).\end{aligned} \tag{31}$$

Our implementations of Eqs. (26), (28), and (30) frequently use the majorant form based on $\mathcal{R}_r(\cdot)$ from Eq. (25) (with one exception as described in the next subsection). This choice generally reduces memory requirements and computational complexity, since $\mathcal{R}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))$ will be a smaller matrix than $\mathcal{N}_r(\mathcal{P}_{\mathbf{X}}(\hat{\mathbf{f}}^{(i-1)}))$ whenever $r < (Q-r)$ (which is true for most multi-channel reconstruction scenarios.).

2.2.3 FFT-Based Computations

All of the algorithms described above depend on being able to evaluate the LORAKS matrix constructor $\mathcal{P}_{\mathbf{X}}(\mathbf{f})$ repeatedly for different choices of $\mathbf{f}$. However, it's important to note that constructing the LORAKS matrix can be computationally expensive, especially because the LORAKS matrix often has a substantially higher dimension than the original k-space data. In scenarios where the data is large (e.g., high-resolution acquisitions, 3D or higher-dimensional acquisitions, or parallel imaging with a large receiver array), the LORAKS matrices will often occupy a substantial amount of memory if they are calculated explicitly and stored in their entirety [12].

A key recent observation [34] is that, because the LORAKS matrices are associated with shift-invariant convolution operations, it is possible to use FFT-based implementations of fast convolution to rapidly compute matrix-vector multiplications of the form $\mathcal{P}_{\mathbf{X}}(\mathbf{f})\mathbf{n}$ for arbitrary vectors $\mathbf{f}$ and $\mathbf{n}$, without the need to explicitly calculate the matrix $\mathcal{P}_{\mathbf{X}}(\mathbf{f})$. This observation is useful for accelerating computations associated with AC-LORAKS (i.e., Eqs. (11), (13), and (15)) and the multiplicative half-quadratic MM algorithms (i.e., Eqs. (26), (28), (30)), which don't possess the simple analytic inversion formulae associated with the additive half-quadratic algorithm, and for which this type of multiplication appears as a component of the operators that need to be inverted.

We illustrate this FFT-based approach for the case of the single-channel $\mathbf{C}$ matrix. Specifically, consider the computation of the matrix-matrix product $\mathcal{P}_{\mathbf{C}}(\mathbf{f})\hat{\mathbf{V}}$, which is a subcomponent of the $\mathcal{M}_1$ operator from Eq. (12). It is easy to see that the ith column of $\mathcal{P}_{\mathbf{C}}(\mathbf{f})\hat{\mathbf{V}}$ can be computed by the operator

$$\mathcal{L}_i(\mathbf{f}) \triangleq \mathcal{P}_{\mathbf{C}}(\mathbf{f})\hat{\mathbf{v}}_i, \tag{32}$$

where $\hat{\mathbf{v}}_i$ is ith column of $\hat{\mathbf{V}}$. Because of the convolutional structure of the $\mathbf{C}$ matrix, this operator can be computed with FFTs, leveraging the fact that standard convolution can be implemented using zero-padded circular convolution, and circular convolution can be implemented efficiently using FFTs. Specifically, Eq. (32) is equivalent to

$$\mathcal{L}_i(\mathbf{f}) = \mathbf{T}\mathcal{F}^{-1}(\mathcal{F}(\mathbf{Z}_2\hat{\mathbf{v}}_i) \odot \mathcal{F}(\mathbf{Z}_1\mathbf{f})), \tag{33}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are FFT and inverse FFT operators; $\mathbf{Z}_1$ and $\mathbf{Z}_2$ are zero-padding operators, $\odot$ represents the Hadamard product operation (i.e., element-wise multiplication), and $\mathbf{T}$ extracts the relevant samples (i.e., the samples corresponding to the LORAKS neighborhood centers) from the convolution output.

The adjoint of $\mathcal{L}_i$ is also easy to define using FFTs:

$$\mathcal{L}_i^*(\mathbf{y}) = \mathbf{Z}_1^H\mathcal{F}^{-1}(\overline{\mathcal{F}(\mathbf{Z}_2\hat{\mathbf{v}}_i)} \odot \mathcal{F}(\mathbf{T}^H\mathbf{y})), \tag{34}$$

where $\overline{\mathbf{x}}$ is used to denote the complex conjugate of $\mathbf{x}$.

Using these FFT-based definitions, the computation of $\mathcal{M}_1$ from Eq. (12) can then be calculated using

$$\begin{aligned}
\mathcal{M}_1(\mathbf{f}) &\triangleq \mathbf{A}^H\mathbf{A}\mathbf{f} + \lambda\mathcal{P}_{\mathbf{X}}^*(\mathcal{P}_{\mathbf{X}}(\mathbf{f})\hat{\mathbf{V}}\hat{\mathbf{V}}^H) \\
&= \mathbf{A}^H\mathbf{A}\mathbf{f} + \lambda\sum_i \mathcal{L}_i^*(\mathcal{L}_i(\mathbf{f})) \\
&= \mathbf{A}^H\mathbf{A}\mathbf{f} + \lambda\sum_i \mathbf{Z}_1^H\mathcal{F}^{-1}(\overline{\mathcal{F}(\mathbf{Z}_2\hat{\mathbf{v}}_i)} \odot \mathcal{F}(\mathbf{T}^H\mathbf{T}\mathcal{F}^{-1}(\mathcal{F}(\mathbf{Z}_2\hat{\mathbf{v}}_i) \odot \mathcal{F}(\mathbf{Z}_1\mathbf{f})))).
\end{aligned} \tag{35}$$

Further computational accelerations are possible if we approximate the binary diagonal matrix $\mathbf{T}^H\mathbf{T}$ with an identity matrix [34]. This approximation only effects the behavior at the edges of k-space and has relatively small impact on the reconstruction result [34]. With this approximation, an approximation of $\mathbf{M}_1$ can be implemented using

$$\begin{aligned}
\mathcal{M}_1(\mathbf{f}) &\approx \mathbf{A}^H\mathbf{A}\mathbf{f} + \lambda\sum_i \mathbf{Z}_1^H\mathcal{F}^{-1}(\overline{\mathcal{F}(\mathbf{Z}_2\hat{\mathbf{v}}_i)} \odot \mathcal{F}(\mathbf{Z}_2\hat{\mathbf{v}}_i) \odot \mathcal{F}(\mathbf{Z}_1\mathbf{f})) \\
&= \mathbf{A}^H\mathbf{A}\mathbf{f} + \lambda\mathbf{Z}_1^H\mathcal{F}^{-1}\left(\left(\sum_i |\mathcal{F}(\mathbf{Z}_2\hat{\mathbf{v}}_i)|^2\right) \odot \mathcal{F}(\mathbf{Z}_1\mathbf{f})\right) \\
&= \mathbf{A}^H\mathbf{A}\mathbf{f} + \lambda\mathbf{Z}_1^H\mathcal{F}^{-1}(\mathbf{N} \odot \mathcal{F}(\mathbf{Z}_1\mathbf{f})),
\end{aligned} \tag{36}$$

where $\mathbf{N} = \sum_i |\mathcal{F}(\mathbf{Z}_2 \hat{\mathbf{v}}_i)|^2$ is a diagonal matrix (in the single-channel case). Note that when we approximate $\mathbf{T}^H\mathbf{T}$ with an identity matrix, we observe that using the nullspace-based majorant from Eq. (24) is preferred over the majorant from (25) based on the signal subspace. As a result, our software implementation for this case uses the majorant from Eq. (24).

Although we have only showed $\mathcal{M}_1$ computation for the case of the single-channel $\mathbf{C}$ matrix, we can apply similar approaches for the other operators $\mathcal{M}_i$, the other single-channel LORAKS matrices $\mathbf{S}$ and $\mathbf{W}$, and to the multi-channel LORAKS matrices (note that in the multi-channel case, $\mathbf{N}$ is no longer block diagonal, but instead has the form of a Hermitian-symmetric block matrix, where each block is diagonal).

3 Software

The supplementary MATLAB code contains three main LORAKS-related functions, `P_LORAKS.m`, `AC_LORAKS.m`, and `SENSE_LORAKS.m` (which are described below), as well as several example demonstration scripts (which are described in Sec. 4).

3.1 P_LORAKS.m

The `P_LORAKS.m` function provides capabilities to solve the optimization problems from Eqs. (5) or (6), and provides a variety of options as described in the MATLAB `help` documentation (as reproduced below):

```
function [recon] = P_LORAKS(kData, kMask, rank, R, LORAKS_type, lambda, alg, tol, max_iter, VCC)
% This function provides capabilities to solve single-channel and multi-channel
% LORAKS reconstruction problems using one of the formulations from either
% Eq. (5) (which uses LORAKS as regularization and does not require strict data-
% consistency) or Eq. (6) (which enforces strict data-consistency) from the
% technical report:
%
% [1] T. H. Kim, J. P. Haldar.  LORAKS Software Version 2.0:
%     Faster Implementation and Enhanced Capabilities.  University of Southern
%     California, Los Angeles, CA, Technical Report USC-SIPI-443, May 2018.
%
% The problem formulations implemented by this function were originally reported
% in:
%
% [2] J. P. Haldar. Low-Rank Modeling of Local k-Space Neighborhoods (LORAKS)
%     for Constrained MRI. IEEE Transactions on Medical Imaging 33:668-681,
%     2014.
%
% [3] J. P. Haldar, J. Zhuo. P-LORAKS: Low-Rank Modeling of Local k-Space
%     Neighborhoods with Parallel Imaging Data. Magnetic Resonance in Medicine
%     75:1499-1514, 2016.
%
% ********************
%   Input Parameters:
% ********************
%
%    kData: A 3D (size N1 x N2 x Nc) array of measured k-space data to be
%           reconstructed.  The first two dimensions correspond to k-space
%           positions, while the third dimension corresponds to the channel
%           dimension for parallel imaging.  Unsampled data samples should be
%           zero-filled.  The software will use the multi-channel formulation if
%           Nc > 1, and will otherwise use the single-channel formulation.
%
%    kMask: A binary mask of size N1 x N2 that corresponds to the same k-space
%           sampling grid used in kData.   Each entry has value 1 if the
%           corresponding k-space location was sampled and has value 0 if that
%           k-space location was not measured.
%
```

```
%    rank:  The matrix rank value used to define the non-convex regularization
%           penalty from Eq. (2) of Ref. [1].
%
%    R: The k-space radius used to construct LORAKS neighborhoods. If not
%       specified, the software will use R=3 by default.
%
%    LORAKS_type: A string that specifies the type of LORAKS matrix that will
%                 be used in reconstruction.  Possible options are: 'C', 'S',
%                 and 'W'.  If not specified, the software will use
%                 LORAKS_type='S' by default.
%
%    lambda: The regularization parameter from Eq. (5).  If lambda=0, the
%            software will use the data-consistency constrained formulation from
%            Eq. (6) instead.  If not specified, the software will use lambda=0
%            by default.
%
%    alg: A parameter that specifies which algorithm the software should use for
%         computation. There are four different options:
%            -alg=1: This choice will use the additive half-quadratic algorithm,
%                    as described in Eq. (18) or (20) of Ref. [1].
%            -alg=2: This choice will use the multiplicative half-quadratic
%                    algorithm, as described in Eq. (26) or (28) of Ref. [1].
%                    This version does NOT use FFTs.
%            -alg=3: This choice will use the multiplicative half-quadratic
%                    algorithm, as described in Eq. (26) or (28) of Ref. [1].
%                    This version uses FFTs without approximation, as in
%                    Eq. (35) of Ref. [1].
%            -alg=4: This choice will use the multiplicative half-quadratic
%                    algorithm, as described in Eq. (26) or (28) of Ref. [1].
%                    This version uses FFTs with approximation, as in Eq. (36)
%                    of Ref. [1].
%         If not specified, the software will use alg=4 by default.
%
%    tol: A convergence tolerance.  The computation will halt if the relative
%         change (measured in the Euclidean norm) between two successive
%         iterates is small than tol.  If not specified, the software will use
%         tol=1e-3 by default.
%
%    max_iter: The computation will halt if the number of iterations exceeds
%              max_iter.  If not specified, the software will default to using
%              max_iter=1000 for the additive half-quadratic algorithm (alg=1),
%              and will use max_iter=50 for the multiplicative half-quadratic
%              algorithms (alg=2,3, or 4).
%
%    VCC: The software will use virtual conjugate coils if VCC=1, and otherwise
%         will not.  If not specified, the software will use VCC=0 by default.
%
% **********************
%   Output Parameters:
% **********************
%
%    recon: The array (size N1 x N2 x Nc) of reconstructed k-space data.
```

3.2 AC_LORAKS.m

The `AC_LORAKS.m` function provides capabilities to solve the optimization problems from Eqs. (11) or (13), and provides a variety of options as described in the MATLAB `help` documentation (as reproduced below):

```
function [recon] = AC_LORAKS(kData, kMask, rank, R, LORAKS_type, lambda, alg, tol, max_iter, VCC)
% This function provides capabilities to solve single-channel and multi-channel
% AC-LORAKS reconstruction problems using one of the formulations from either
% Eq. (11) (which uses LORAKS as regularization and does not require strict data-
```

```
% consistency) or Eq. (13) (which enforces strict data-consistency) from the
% technical report:
%
% [1] T. H. Kim, J. P. Haldar.  LORAKS Software Version 2.0:
%     Faster Implementation and Enhanced Capabilities.  University of Southern
%     California, Los Angeles, CA, Technical Report USC-SIPI-443, May 2018.
%
% The problem formulations implemented by this function were originally reported
% in:
%
% [2] J. P. Haldar. Autocalibrated LORAKS for Fast Constrained MRI
%     Reconstruction. IEEE International Symposium on Biomedical Imaging: From
%     Nano to Macro, New York City, 2015, pp. 910-913.
%
% *********************
%   Input Parameters:
% *********************
%
%    kData: A 3D (size N1 x N2 x Nc) array of measured k-space data to be
%           reconstructed.  The first two dimensions correspond to k-space
%           positions, while the third dimension corresponds to the channel
%           dimension for parallel imaging.  Unsampled data samples should be
%           zero-filled.  The software will use the multi-channel formulation if
%           Nc > 1, and will otherwise use the single-channel formulation.
%
%    kMask: A binary mask of size N1 x N2 that corresponds to the same k-space
%           sampling grid used in kData.   Each entry has value 1 if the
%           corresponding k-space location was sampled and has value 0 if that
%           k-space location was not measured. It is assumed that kMask will
%           contain a fully-sampled autocalibration region that is of
%           sufficiently-large size that it is possible to estimate the
%           nullspace of the LORAKS matrix by looking at the nullspace of a
%           fully-sampled submatrix.  An error will occur if the software cannot
%           find such an autocalibration region.
%
%    rank:  The matrix rank value used to define the dimension of the V matrix
%           in Eq. (10) of Ref. [1].
%
%    R: The k-space radius used to construct LORAKS neighborhoods. If not
%       specified, the software will use R=3 by default.
%
%    LORAKS_type: A string that specifies the type of LORAKS matrix that will
%                 be used in reconstruction.  Possible options are: 'C', 'S',
%                 and 'W'.  If not specified, the software will use
%                 LORAKS_type='S' by default.
%
%    lambda: The regularization parameter from Eq. (11).  If lambda=0, the
%            software will use the data-consistency constrained formulation from
%            Eq. (13) instead.  If not specified, the software will use lambda=0
%            by default.
%
%    alg: A parameter that specifies which algorithm the software should use for
%         computation. There are three different options:
%            -alg=2: This choice will use the multiplicative half-quadratic
%                    algorithm, as described in Eq. (11) or (13) of Ref. [1].
%                    This version does NOT use FFTs.
%            -alg=3: This choice will use the multiplicative half-quadratic
%                    algorithm, as described in Eq. (11) or (13) of Ref. [1].
%                    This version uses FFTs without approximation, as in
%                    Eq. (35) of Ref. [1].
%            -alg=4: This choice will use the multiplicative half-quadratic
%                    algorithm, as described in Eq. (11) or (13) of Ref. [1].
%                    This version uses FFTs with approximation, as in Eq. (36)
%                    of Ref. [1].
%         If not specified, the software will use alg=4 by default.
```

```
%
%    tol: A convergence tolerance.  The computation will halt if the relative
%         change (measured in the Euclidean norm) between two successive
%         iterates is small than tol.  If not specified, the software will use
%         tol=1e-3 by default.
%
%    max_iter: The computation will halt if the number of iterations exceeds
%              max_iter.  If not specified, the software will default to using
%               max_iter=50.
%
%    VCC: The software will use virtual conjugate coils if VCC=1, and otherwise
%         will not.  If not specified, the software will use VCC=0 by default.
%
% **********************
%   Output Parameters:
% **********************
%
%    recon: The array (size N1 x N2 x Nc) of reconstructed k-space data.
```

3.3 SENSE_LORAKS.m

The `SENSE_LORAKS.m` function provides capabilities to solve the optimization problem from Eq. (9), and provides a variety of options as described in the MATLAB `help` documentation (as reproduced below):

```
function [recon] = SENSE_LORAKS(kData, kMask, coil_sens, rank, lambda, R, LORAKS_type,  alg, ...
    tol, max_iter)
% This function provides capabilities to solve multi-channel SENSE-LORAKS
% reconstruction problems using the formulation from Eq. (9) from the
% technical report:
%
% [1] T. H. Kim, J. P. Haldar.  LORAKS Software Version 2.0:
%     Faster Implementation and Enhanced Capabilities.  University of Southern
%     California, Los Angeles, CA, Technical Report USC-SIPI-443, May 2018.
%
% The problem formulation implemented by this function was originally reported
% in:
%
% [2] T. H. Kim, J. P. Haldar. LORAKS makes better SENSE: Phase?constrained
%     partial fourier SENSE reconstruction without phase calibration. Magnetic
%     Resonance in Medicine 77:1021-1035, 2017.
%
% **********************
%   Input Parameters:
% **********************
%
%    kData: A 3D (size N1 x N2 x Nc) array of measured k-space data to be
%           reconstructed.  The first two dimensions correspond to k-space
%           positions, while the third dimension corresponds to the channel
%           dimension for parallel imaging.  Unsampled data samples should be
%           zero-filled.
%
%    kMask: A binary mask of size N1 x N2 that corresponds to the same k-space
%           sampling grid used in kData.   Each entry has value 1 if the
%           corresponding k-space location was sampled and has value 0 if that
%           k-space location was not measured.
%
%    coil_sens: A 3D (size N1 x N2 x Nc) array of estimated coil sensitivity
%               profiles.
%
%    rank:  The matrix rank value used to define the non-convex regularization
%           penalty from Eq. (2) of Ref. [1].
%
```

```
%    lambda: The regularization parameter from Eq. (9).
%
%    R: The k-space radius used to construct LORAKS neighborhoods. If not
%       specified, the software will use R=3 by default.
%
%    LORAKS_type: A string that specifies the type of LORAKS matrix that will
%                 be used in reconstruction.  Possible options are: 'C', 'S',
%                 and 'W'.  If not specified, the software will use
%                 LORAKS_type='S' by default.
%
%    alg: A parameter that specifies which algorithm the software should use for
%         computation. There are four different options:
%            -alg=1: This choice will use the additive half-quadratic algorithm,
%                    as described in Eq. (22) of Ref. [1].
%            -alg=2: This choice will use the multiplicative half-quadratic
%                    algorithm, as described in Eq. (30) of Ref. [1].
%                    This version does NOT use FFTs.
%            -alg=3: This choice will use the multiplicative half-quadratic
%                    algorithm, as described in Eq. (30) of Ref. [1].
%                    This version uses FFTs without approximation, as in
%                    Eq. (35) of Ref. [1].
%            -alg=4: This choice will use the multiplicative half-quadratic
%                    algorithm, as described in Eq. (30) of Ref. [1].
%                    This version uses FFTs with approximation, as in Eq. (36)
%                    of Ref. [1].
%         If not specified, the software will use alg=4 by default.
%
%    tol: A convergence tolerance.  The computation will halt if the relative
%         change (measured in the Euclidean norm) between two successive
%         iterates is small than tol.  If not specified, the software will use
%         tol=1e-3 by default.
%
%    max_iter: The computation will halt if the number of iterations exceeds
%              max_iter.  If not specified, the software will default to using
%              max_iter=1000 for the additive half-quadratic algorithm (alg=1),
%              and will use max_iter=50 for the multiplicative half-quadratic
%              algorithms (alg=2,3, or 4).
%
% **********************
%   Output Parameters:
% **********************
%
%    recon: The N1 x N2 reconstructed image.
```

4 Examples and Usage Recommendations

Our software provides a lot of different options, and selecting between different alternative approaches may be a little daunting for users who are new to LORAKS. The following illustrative examples are provided in the `examples` subfolder of our software distribution, and are designed to provide users with some high-level guidance.

4.1 Single-channel reconstruction

Our first example, provided in `ex1.m`, demonstrates the use of LORAKS and AC-LORAKS with single-channel data. Gold standard T2-weighted data was fully sampled on a 256×340 k-space grid with a multi-channel receiver array coil, and was coil compressed down to a single channel. The magnitude and phase of this gold standard data is shown in Fig. 1.

To demonstrate the flexibility and characteristics of LORAKS and AC-LORAKS, this dataset is retrospectively undersampled using the four different sampling patterns shown in Fig. 2. All four sampling patterns have an acceleration factor of 2 (i.e., retaining only 50% of the original k-space data), which is somewhat aggressive for single-channel

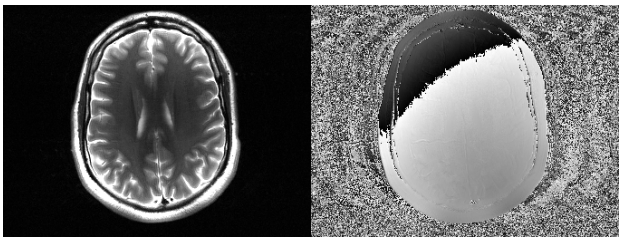
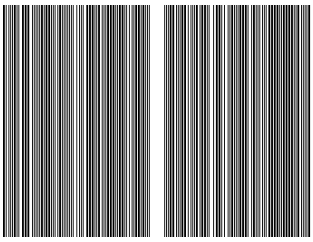
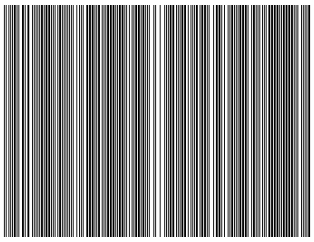


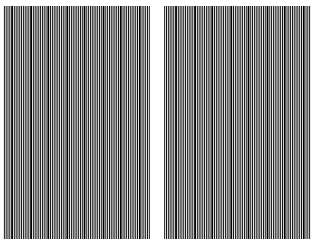
Figure 1: Gold standard magnitude (left) and phase (right) images for the single-channel dataset.



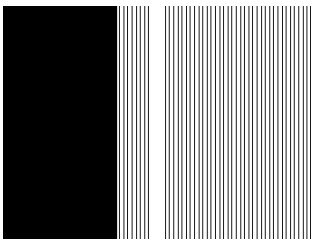
(a) Random Sampling with Calibration Region



(b) Calibrationless Random Sampling

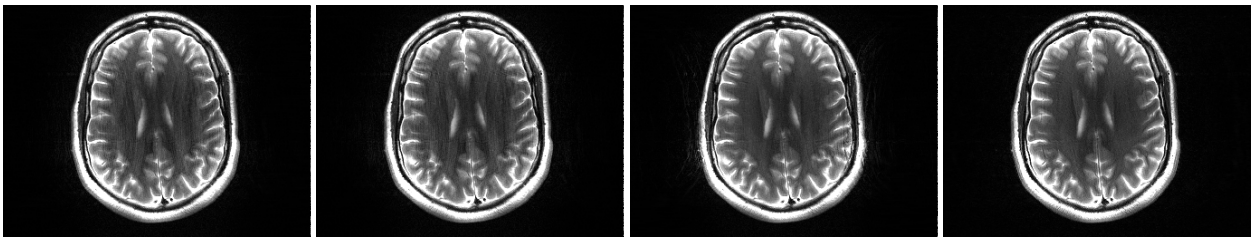


(c) Uniform Sampling with Calibration Region



(d) Partial Fourier Sampling with Calibration Region

Figure 2: Sampling patterns used with single-channel data.

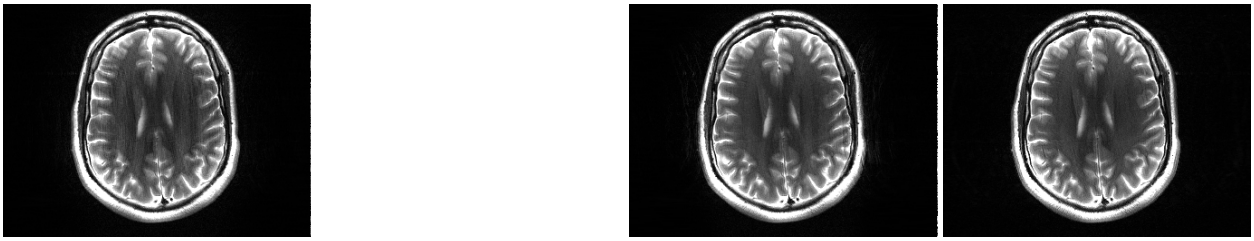


(a) Sampling using Fig. 2(a), NRMSE = 0.088, time = 18.5 sec.

(b) Sampling using Fig. 2(b), NRMSE = 0.087, time = 21.1 sec.

(c) Sampling using Fig. 2(c), NRMSE = 0.102, time = 15.6 sec.

(d) Sampling using Fig. 2(d), NRMSE = 0.083, time = 6.6 sec.



(e) Sampling using Fig. 2(a), NRMSE = 0.082, time = 3.8 sec.

(f) Sampling using Fig. 2(c), NRMSE = 0.089, time = 3.9 sec.

(g) Sampling using Fig. 2(d), NRMSE = 0.084, time = 3.6 sec.

Figure 3: Reconstruction results from `ex1.m`. The top row shows results obtained with LORAKS (using `P_LORAKS.m`), while the bottom row shows results obtained with AC-LORAKS (using `AC_LORAKS.m`).

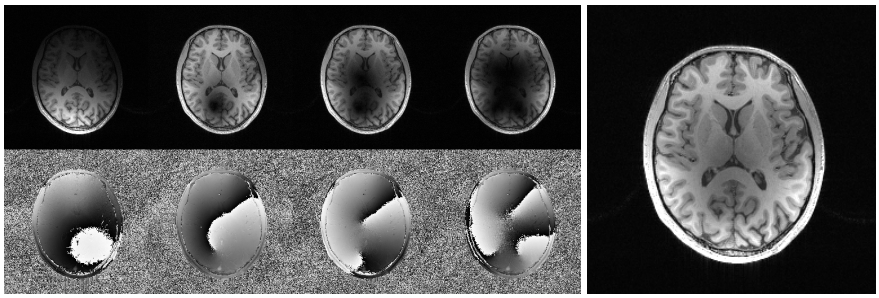
data with one-dimensional undersampling (i.e., acceleration only along the phase encoding dimension). The retrospectively undersampled data is reconstructed using `P_LORAKS.m` and `AC_LORAKS.m` using the default reconstruction parameters (i.e., using the $\mathbf{S}$ matrix with a LORAKS neighborhood radius of 3; using exact data consistency as in Eqs. (6) and (13); and using the FFT-based multiplicative half-quadratic algorithm with approximation of $\mathbf{T}^H\mathbf{T}$ as an identity matrix, as described in Sec. 2.2.3).

Reconstruction results are shown in Fig. 3, and we report the normalized mean-squared error (NRMSE) and reconstruction time (measured on one of our standard desktop computers) in addition to showing qualitative results. As can be seen, all of these reconstruction results are reasonably successful, despite the variety of different sampling patterns (including random sampling, calibrationless sampling, uniform sampling, and partial Fourier sampling) and despite the relatively aggressive acceleration factor. Classical image reconstruction techniques will not be as successful across such a wide range of different settings. Although both approaches generally lead to similar image quality and NRMSE, LORAKS reconstruction using Eq. (6) tends to be much slower than AC-LORAKS reconstruction using Eq. (13). On the other hand, the LORAKS implementation is slightly more generally applicable than the AC-LORAKS implementation, since it accommodates calibrationless sampling (e.g., Fig. 2(b)). While AC-LORAKS can be used with external calibration data [10], our current software implementation only supports autocalibration. It would be straightforward to modify `AC_LORAKS.m` to allow external calibration data if so desired, and the use of external calibration preserves the significant speed advantages of AC-LORAKS while also enabling substantial improvements in image quality [10].

We should note that we have not shown results for uniform undersampling without a calibration region. As described in [10], getting good LORAKS results with calibrationless uniform undersampling (e.g., as in standard echo-planar imaging) requires additional prior information and/or external calibration data.

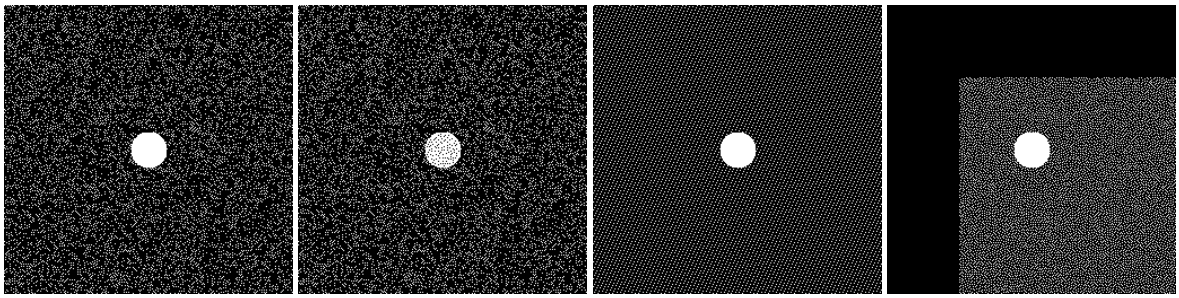
4.2 Multi-channel reconstruction

Our second example, provided in `ex2.m`, demonstrates the use of P-LORAKS, AC-LORAKS, and SENSE-LORAKS with multi-channel data. Gold standard MPRAGE data was fully sampled on a 256×256 k-space grid with a multi-channel receiver array coil, and was coil compressed down to four channels. The magnitude and phase and the root-sum-of-squares coil combination results for this gold standard data is shown in Fig. 4.



(a) (b)

Figure 4: (a) Gold standard magnitude (top) and phase (bottom) images for each channel of the multi-channel dataset. (b) The gold standard root-sum-of-squares combination of the channels.



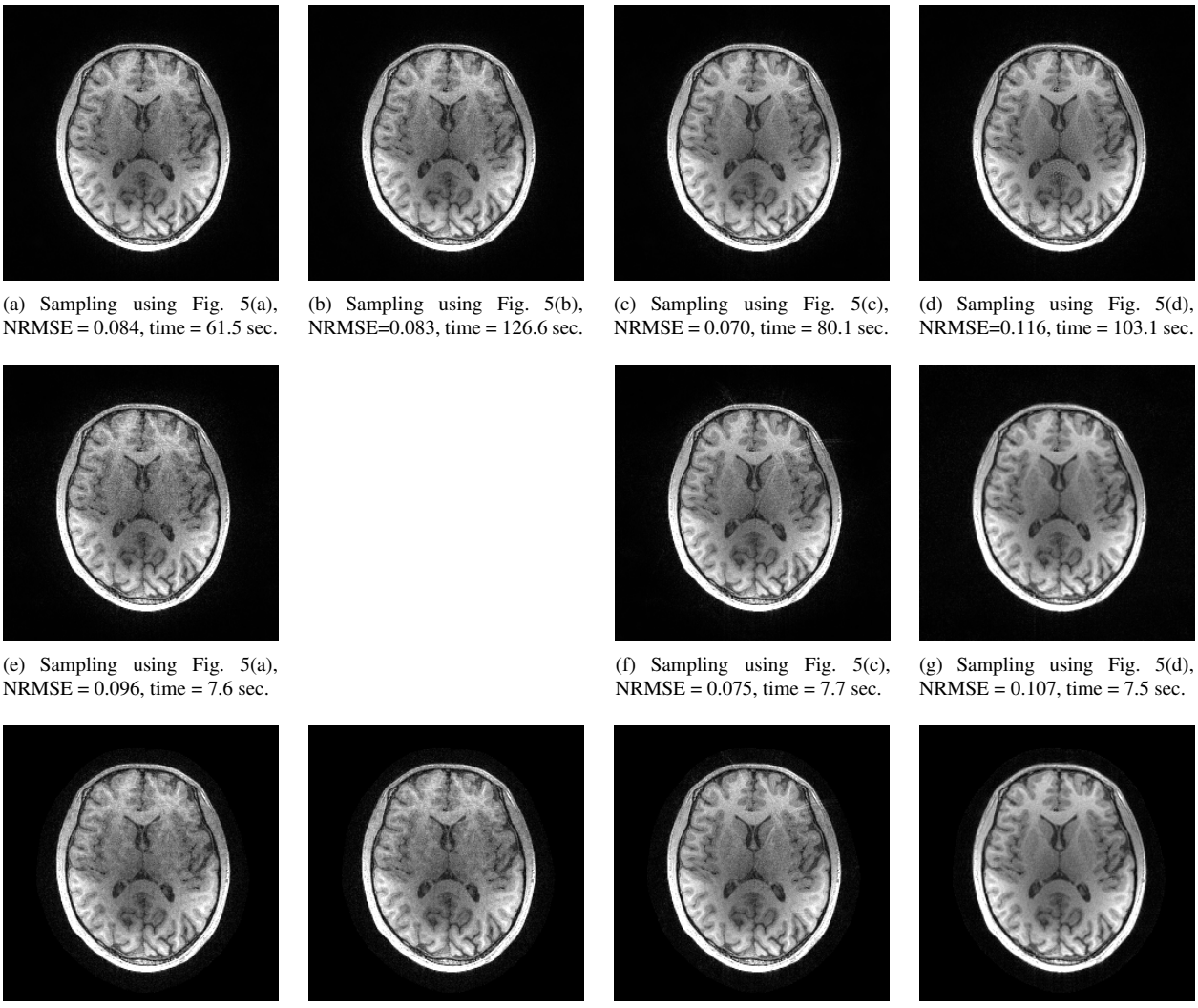
(a) Random Sampling with Calibration Region (b) Calibrationless Random Sampling (c) Uniform Sampling with Calibration Region (d) Partial Fourier Sampling with Calibration Region

Figure 5: Sampling patterns used with multi-channel data.

To demonstrate the flexibility and characteristics of P-LORAKS, AC-LORAKS, and SENSE-LORAKS, this dataset is retrospectively undersampled using the four different sampling patterns shown in Fig. 5. All four sampling patterns have an acceleration factor of 7 (i.e., retaining only $\approx$14% of the original k-space data), which is very aggressive for four-channel data. Reconstruction results at this acceleration factor are not necessarily of diagnostic quality, but we've chosen an aggressive acceleration strategy to better highlight the differences between different reconstruction approaches. The retrospectively undersampled data is reconstructed using `P_LORAKS.m`, `AC_LORAKS.m`, and `SENSE_LORAKS.m` using the default reconstruction parameters (i.e., using the $\mathbf{S}$ matrix with a LORAKS neighborhood radius of 3; using exact data consistency as in Eqs. (6) and (13) for P-LORAKS and AC-LORAKS, while using regularization as in Eq. (9) for SENSE-LORAKS; and using the FFT-based multiplicative half-quadratic algorithm with approximation of $\mathbf{T}^H\mathbf{T}$ as an identity matrix, as described in Sec. 2.2.3).

Reconstruction results are shown in Fig. 6. Similar to the single-channel case, all of these reconstruction results are reasonably successful, despite the variety of different sampling patterns (including random sampling, calibrationless sampling, uniform sampling, and partial Fourier sampling) and despite the relatively aggressive acceleration factor. Although all three approaches generally lead to roughly similar image quality and NRMSE, LORAKS reconstruction using Eq. (6) tends to be much slower than SENSE-LORAKS reconstruction using Eq. (9), which is in turn much slower than AC-LORAKS reconstruction using Eq. (13). The SENSE-LORAKS reconstruction is the most general in some ways, because can be used with arbitrary sampling patterns (including calibrationless uniform undersampling [8]), though requires slightly more prior information (in the form of coil sensitivity maps) than the other two approaches. Our implementation of SENSE-LORAKS does not include the additional Tikhonov regularization term described in [8] for simplicity, although the code is easily modified to include this additional regularization term, and the additional regularization would be beneficial for reducing noise amplification and improving image quality.

Similar to the single-channel case, the LORAKS implementation is slightly more generally applicable than the



(a) Sampling using Fig. 5(a), NRMSE = 0.084, time = 61.5 sec.

(b) Sampling using Fig. 5(b), NRMSE=0.083, time = 126.6 sec.

(c) Sampling using Fig. 5(c), NRMSE = 0.070, time = 80.1 sec.

(d) Sampling using Fig. 5(d), NRMSE=0.116, time = 103.1 sec.

(e) Sampling using Fig. 5(a), NRMSE = 0.096, time = 7.6 sec.

(f) Sampling using Fig. 5(c), NRMSE = 0.075, time = 7.7 sec.

(g) Sampling using Fig. 5(d), NRMSE = 0.107, time = 7.5 sec.

(h) Sampling using Fig. 5(a), NRMSE = 0.103, time = 20.2 sec.

(i) Sampling using Fig. 5(b), NRMSE = 0.102, time = 23.8 sec.

(j) Sampling using Fig. 5(c), NRMSE = 0.085, time = 16.7 sec.

(k) Sampling using Fig. 5(d), NRMSE = 0.115, time = 12.9 sec.

Figure 6: Reconstruction results from `ex1.m`. The top row shows results obtained with P-LORAKS (using `P_LORAKS.m`), while the middle row shows results obtained with AC-LORAKS (using `AC_LORAKS.m`), and the bottom row shows results obtained with SENSE-LORAKS (using `SENSE_LORAKS.m`).

AC-LORAKS implementation, since it accommodates calibrationless sampling (e.g., Fig. 5(b)). As before, it would be straightforward to modify `AC_LORAKS.m` to allow external calibration data if so desired, and the AC-LORAKS approach has substantial speed advantages.

4.3 Choice of LORAKS matrix and neighborhood radius

All of the previous results used the $\mathbf{S}$ matrix with a LORAKS neighborhood radius of 3, and without using virtual conjugate coils. Our third example, provided in `ex3.m`, demonstrates the behavior when these parameter settings are changed, in the context of the multi-channel data from Sec. 4.2 and the random sampling pattern from Fig. 5(a). Results without and with virtual conjugate coils are respectively shown in Figs. 7 and 8. The results are consistent with our past experience [9], and show that the $\mathbf{S}$ matrix without virtual coils, the $\mathbf{S}$ matrix with virtual coils, or the $\mathbf{C}$ matrix with virtual coils are generally the top-performing LORAKS matrices, and are relatively similar to each other in reconstructed image quality. On the other hand, the $\mathbf{W}$ matrices tend to have the worst performance. Among the top-performing matrices, we often recommend using the $\mathbf{S}$ matrix without virtual conjugate coils, because using virtual coils increases the memory requirements and computational complexity without a noticeable improvement in quality for the $\mathbf{S}$ matrix. The use of virtual coils can often lead to substantial quality improvements for the $\mathbf{C}$ matrix and sometimes (but not in this specific example) for the $\mathbf{W}$ matrix. Virtual conjugate coils are generally not as useful for the $\mathbf{S}$ matrix because their main purpose is to introduce phase constraints, while the $\mathbf{S}$ matrix already exploits such constraints.

The choice of the neighborhood radius represents a classical trade-off in constrained reconstruction. Larger neighborhood radii allows the LORAKS model to be more flexible and adaptable, but also more sensitive to noise. Using larger neighborhood radii is also generally associated with increased memory requirements, increased computational complexity per iteration, and a larger number of iterations to reach convergence.

4.4 Choice of algorithm

Our software provides access to four different algorithms: (alg 1) the original additive half-quadratic approach (Sec. 2.2.1), (alg 2) the multiplicative half-quadratic approach (Sec. 2.2.2), (alg 3) the multiplicative half-quadratic approach with FFT-based computations (Sec. 2.2.3), and (alg 4) the multiplicative half-quadratic approach with FFT-based computations and with approximation of $\mathbf{T}^H\mathbf{T}$ as an identity matrix (Sec. 2.2.3). These four different algorithms are compared in our fourth example, provided in `ex4.m`.

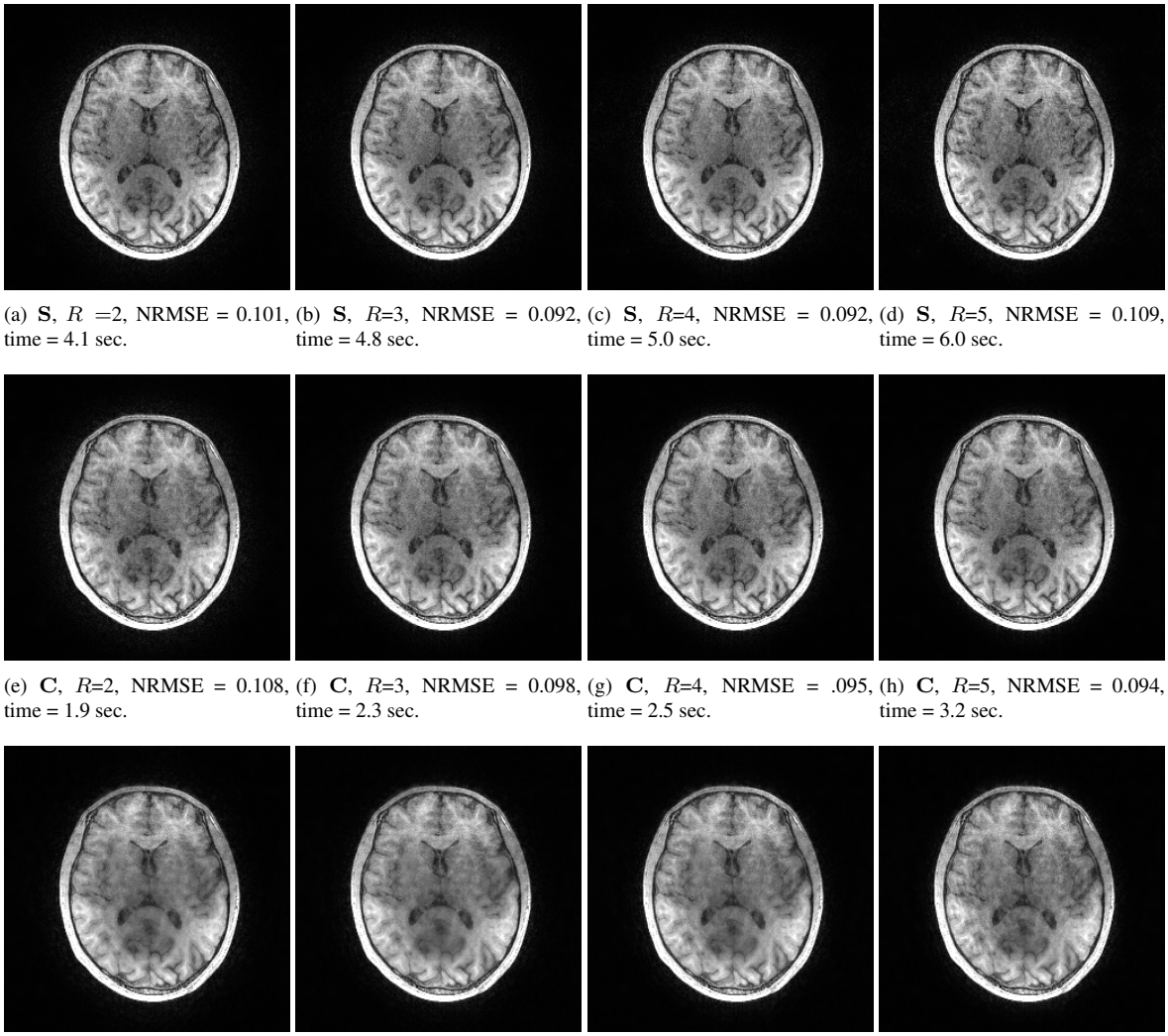
This example considers P-LORAKS reconstruction in the context of the multi-channel data from Sec. 4.2 and the random sampling pattern from Fig. 5(a). P-LORAKS reconstruction is performed using the software default settings, other than varying the choice of algorithm. NRMSE, iteration, and computation time results are shown below in Table 1.

	alg 1	**alg 2**	**alg 3**	**alg 4**
NRMSE	0.09538	0.08354	0.08354	0.08353
# of iterations	71	10	10	10
Time (sec)	141.8	1059.7	1253.5	61.4

Table 1: Algorithm comparison results from `ex4.m`.

As can be seen, the additive half-quadratic algorithm (alg 1) has different characteristics than the multiplicative half-quadratic algorithms (alg 2, alg 3, and alg 4). In particular, additive half-quadratic approach is characterized by having a small computational cost per iteration, though requires a large number of iterations to converge. The multiplicative half-quadratic algorithms converge much faster, though require more computational effort per iteration. These differences in convergence characteristics mean that the step size for each iteration is different between the additive and multiplicative approaches, such that using the same stopping criterion does not guarantee the same degree of convergence.

As expected, the two unapproximated multiplicative half-quadratic algorithms (alg 2 and alg 3) have identical results, aside from numerical finite precision effects. However, the original version (alg 2) is much faster than the

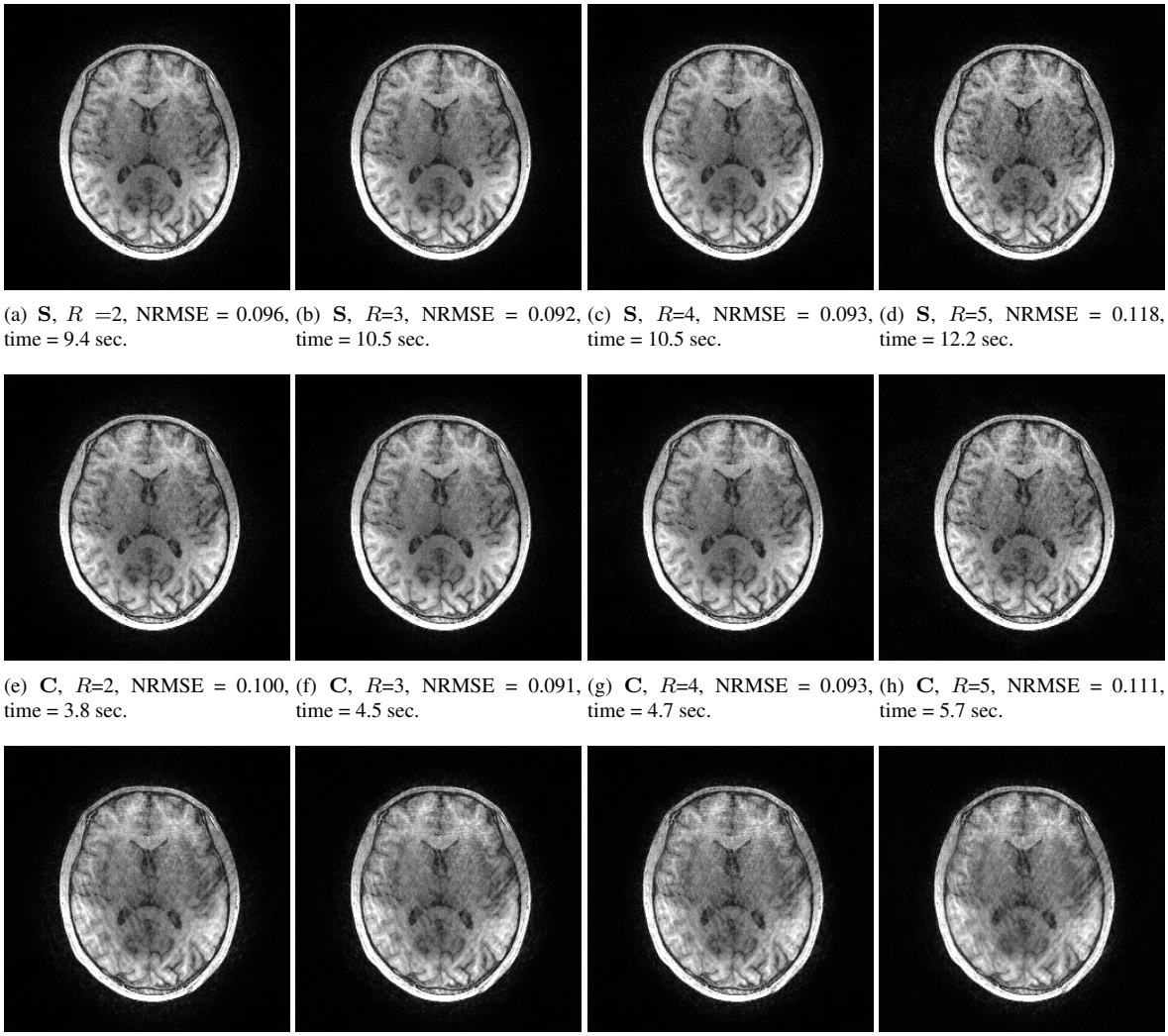


(a) $\mathbf{S}$, R =2, NRMSE = 0.101, time = 4.1 sec. (b) $\mathbf{S}$, R=3, NRMSE = 0.092, time = 4.8 sec. (c) $\mathbf{S}$, R=4, NRMSE = 0.092, time = 5.0 sec. (d) $\mathbf{S}$, R=5, NRMSE = 0.109, time = 6.0 sec.

(e) $\mathbf{C}$, R=2, NRMSE = 0.108, time = 1.9 sec. (f) $\mathbf{C}$, R=3, NRMSE = 0.098, time = 2.3 sec. (g) $\mathbf{C}$, R=4, NRMSE = .095, time = 2.5 sec. (h) $\mathbf{C}$, R=5, NRMSE = 0.094, time = 3.2 sec.

(i) $\mathbf{W}$, R=2, NRMSE = 0.111, time = 6.2 sec. (j) $\mathbf{W}$, R=3, NRMSE = 0.111, time = 7.3 sec. (k) $\mathbf{W}$, R=4, NRMSE = 0.110, time = 6.7 sec. (l) $\mathbf{W}$, R=5, NRMSE = 0.116, time = 8.1 sec.

Figure 7: The effects of different choices of the LORAKS matrix type ($\mathbf{C}$, $\mathbf{S}$, or $\mathbf{W}$), different choices of the LORAKS neighborhood radius R, and not using virtual conjugate coils.



(a) $\mathbf{S}$, R =2, NRMSE = 0.096, time = 9.4 sec. (b) $\mathbf{S}$, R=3, NRMSE = 0.092, time = 10.5 sec. (c) $\mathbf{S}$, R=4, NRMSE = 0.093, time = 10.5 sec. (d) $\mathbf{S}$, R=5, NRMSE = 0.118, time = 12.2 sec.

(e) $\mathbf{C}$, R=2, NRMSE = 0.100, time = 3.8 sec. (f) $\mathbf{C}$, R=3, NRMSE = 0.091, time = 4.5 sec. (g) $\mathbf{C}$, R=4, NRMSE = 0.093, time = 4.7 sec. (h) $\mathbf{C}$, R=5, NRMSE = 0.111, time = 5.7 sec.

(i) $\mathbf{W}$, R=2, NRMSE = 0.134, time = 17.5 sec. (j) $\mathbf{W}$, R=3, NRMSE = 0.135, time = 19.6 sec. (k) $\mathbf{W}$, R=4, NRMSE = 0.137, time = 18.3 sec. (l) $\mathbf{W}$, R=5, NRMSE = 0.146, time = 18.5 sec.

Figure 8: The effects of different choices of the LORAKS matrix type ($\mathbf{C}$, $\mathbf{S}$, or $\mathbf{W}$), different choices of the LORAKS neighborhood radius R, and using virtual conjugate coils.

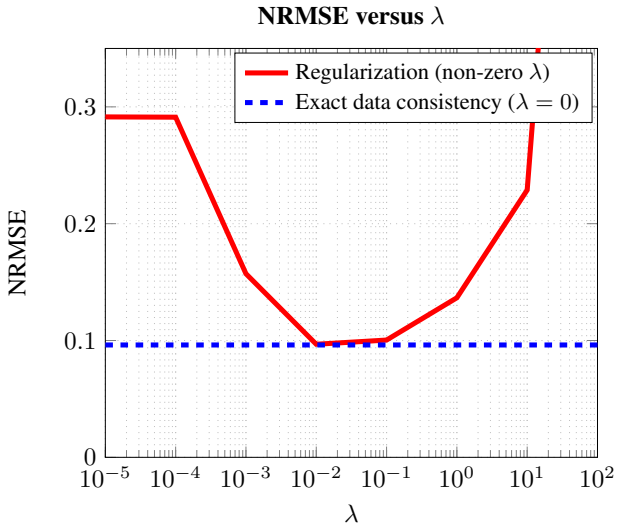


Figure 9: The effects of the regularization parameter λ on AC-LORAKS reconstruction in `ex5.m`.

version using FFTs (alg 3). The approximate multiplicative half-quadratic algorithm (alg 4) yields very similar results to the unapproximated versions, though is by far the fastest algorithm. We generally recommend the use of alg 4 due to its computational efficiency.

4.5 Choice of regularization parameter λ and the maximum number of iterations

Our final examples illustrate the effects of the regularization parameter λ and the maximum number of iterations. Both examples consider AC-LORAKS reconstruction in the context of the multi-channel data from Sec. 4.2 and the random sampling pattern from Fig. 5(a). These reconstructions use default settings, other than varying λ and the maximum number of iterations.

Different choices of λ are illustrated in `ex5.m`, and NRMSE values are shown as a function of λ in Fig. 9. As can be seen, the tuning of λ is nontrivial. When λ is too small, numerical effects dominate and LORAKS regularization doesn't have much effect. On the other hand, choosing λ too large can also lead to significant image reconstruction biases. However, enforcing exact data consistency and using LORAKS only to interpolate/extrapolate missing data (by setting $\lambda = 0$) works well and does not require parameter tuning. It should be noted that our implementation of SENSE-LORAKS does not currently allow setting $\lambda = 0$. (In the context of SENSE, there are several different ways of trying to incorporate exact data consistency constraints, and each of these would be straightforward to incorporate by modifying the code we've provided).

It is also worth mentioning that, like many iterative algorithms, the iterative algorithms we've implemented for LORAKS can demonstrate classical semiconvergence phenomenon, in which truncating the iterative procedure early can have a noise-suppressing regularization effect. This effect is illustrated in `ex6.m`, with results shown in Fig. 10. As a result, it is not always beneficial to iterate the LORAKS algorithms until convergence. On the other hand, rather than relying on semiconvergence (which can be difficult to characterize), it can also be beneficial to include additional regularization penalties into the LORAKS reconstruction to prevent excessive noise amplification [2, 7, 8, 12]. For simplicity, our current software does not provide access to additional regularization, although this would be relatively easy to incorporate through simple modifications of the provided code.

Acknowledgments

This work was supported in part by a USC Annenberg Fellowship, a Kwanjeong Educational Foundation Scholarship, NSF research award CCF-1350563, and NIH research grant R21-EB022951.

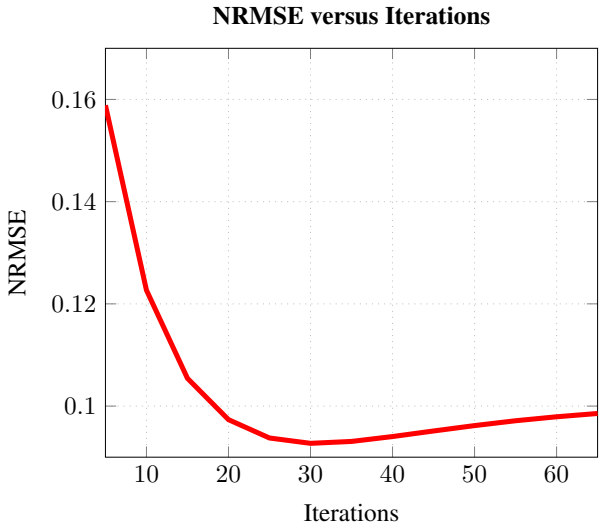


Figure 10: The effects of the number of iterations on AC-LORAKS reconstruction in `ex6.m`.

References

[1] J. P. Haldar, "Calibrationless partial Fourier reconstruction of MR images with slowly-varying phase: A rank-deficient matrix recovery approach," in *ISMRM Workshop on Data Sampling & Image Reconstruction*, Sedona, 2013.

[2] J. P. Haldar, "Low-rank modeling of local k-space neighborhoods (LORAKS) for constrained MRI," *IEEE Trans. Med. Imag.*, vol. 33, pp. 668–681, 2014.

[3] J. P. Haldar, "Low-rank modeling of local k-space neighborhoods (LORAKS): Implementation and examples for reproducible research," University of Southern California, Los Angeles, CA, Tech. Rep. USC-SIPI-414, April 2014.

[4] J. P. Haldar, "Autocalibrated LORAKS for fast constrained MRI reconstruction," in *Proc. IEEE Int. Symp. Biomed. Imag.*, 2015, pp. 910–913.

[5] T. H. Kim and J. P. Haldar, "SMS-LORAKS: Calibrationless simultaneous multislice MRI using low-rank matrix modeling," in *Proc. IEEE Int. Symp. Biomed. Imag.*, 2015, pp. 323–326.

[6] J. P. Haldar, "Low-rank modeling of local k-space neighborhoods: from phase and support constraints to structured sparsity," *Wavelets and Sparsity XVI, Proc. SPIE 9597*, p. 959710, 2015.

[7] J. P. Haldar and J. Zhuo, "P-LORAKS: Low-rank modeling of local k-space neighborhoods with parallel imaging data," *Magn. Reson. Med.*, vol. 75, pp. 1499–1514, 2016.

[8] T. H. Kim, K. Setsompop, and J. P. Haldar, "LORAKS makes better SENSE: Phase-constrained partial fourier SENSE reconstruction without phase calibration," *Magn. Reson. Med.*, vol. 77, pp. 1021–1035, 2017.

[9] J. P. Haldar and T. H. Kim, "Computational imaging with LORAKS: Reconstructing linearly predictable signals using low-rank matrix regularization." in *Proc. Asilomar Conf. Sig. Sys. Comp.*, 2017, pp. 1870–1874.

[10] R. A. Lobos, T. H. Kim, W. S. Hoge, and J. P. Haldar, "Navigator-free EPI ghost correction with structured low-rank matrix models: New theory and methods," *IEEE Trans. Med. Imag.*, 2018, Early View.

[11] B. Bilgic, T. H. Kim, C. Liao, M. K. Manhard, L. L. Wald, J. P. Haldar, and K. Setsompop, "Improving parallel imaging by jointly reconstructing multi-contrast data," *Magn. Reson. Med.*, vol. 80, pp. 619–632, 2018.

[12] T. H. Kim, B. Bilgic, D. Polak, K. Setsompop, and J. P. Haldar, "Wave-LORAKS: Combining wave encoding with structured low-rank matrix modeling for more highly accelerated 3D imaging," 2018, Submitted.

[13] Z. P. Liang, E. M. Haacke, and C. W. Thomas, "High-resolution inversion of finite Fourier transform data through a localised polynomial approximation," *Inverse Probl.*, vol. 5, pp. 831–847, 1989.

[14] J. Zhang, C. Liu, and M. E. Moseley, "Parallel reconstruction using null operations," *Magn. Reson. Med.*, vol. 66, pp. 1241–1253, 2011.

[15] P. J. Shin, P. E. Z. Larson, M. A. Ohliger, M. Elad, J. M. Pauly, D. B. Vigneron, and M. Lustig, "Calibrationless parallel imaging reconstruction based on structured low-rank matrix completion," *Magn. Reson. Med.*, vol. 72, pp. 959–970, 2014.

[16] G. Ongie and M. Jacob, "Off-the-grid recovery of piecewise constant images from few Fourier samples," *SIAM J. Imaging Sci.*, vol. 9, pp. 1004–1041, 2016.

[17] K. H. Jin, D. Lee, and J. C. Ye, "A general framework for compressed sensing and parallel MRI using annihilating filter based low-rank Hankel matrix," *IEEE Trans. Comput. Imaging*, vol. 2, pp. 480–495, 2016.

[18] D. L. Donoho, "An invitation to reproducible computational research," *Biostat.*, vol. 11, pp. 385–388, 2010.

[19] K. F. Cheung and R. J. Marks II, "Imaging sampling below the Nyquist density without aliasing," *J. Opt. Soc. Am. A*, vol. 7, pp. 92–105, 1990.

[20] F. Huang, W. Lin, and Y. Li, "Partial Fourier reconstruction through data fitting and convolution in *k*-space," *Magn. Reson. Med.*, vol. 62, pp. 1261–1269, 2009.

[21] D. K. Sodickson and W. J. Manning, "Simultaneous acquisition of spatial harmonics (SMASH): fast imaging with radiofrequency coil arrays," *Magn. Reson. Med.*, vol. 38, pp. 591–603, 1997.

[22] M. A. Griswold, P. M. Jakob, R. M. Heidemann, M. Nittka, V. Jellus, J. Wang, B. Kiefer, and A. Haase, "Generalized autocalibrating partially parallel acquisitions (GRAPPA)," *Magn. Reson. Med.*, vol. 47, pp. 1202–1210, 2002.

[23] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Rev.*, vol. 52, pp. 471–501, 2010.

[24] Z. P. Liang, F. Boada, T. Constable, E. M. Haacke, P. C. Lauterbur, and M. R. Smith, "Constrained reconstruction methods in MR imaging," *Magn. Reson. Med.*, vol. 4, pp. 67–185, 1992.

[25] M. Lustig, "Post-Cartesian calibrationless parallel imaging reconstruction by structured low-rank matrix completion," in *Proc. Int. Soc. Magn. Reson. Med.*, 2011, p. 483.

[26] M. Blaimer, M. Gutberlet, P. Kellman, F. A. Breuer, H. Kostler, and M. A. Griswold, "Virtual coil concept for improved parallel MRI employing conjugate symmetric signals," *Magn. Reson. Med.*, 2009.

[27] M. Uecker and M. Lustig, "Estimating absolute-phase maps using ESPIRiT and virtual conjugate coils," *Magn. Reson. Med.*, vol. 77, pp. 1201–1207, 2017.

[28] K. P. Pruessmann, M. Weiger, M. B. Scheidegger, and P. Boesiger, "SENSE: sensitivity encoding for fast MRI," *Magn. Reson. Med.*, vol. 42, pp. 952–962, 1999.

[29] K. P. Pruessmann, M. Weiger, P. Börnert, and P. Boesiger, "Advances in sensitivity encoding with arbitrary k-space trajectories," *Magn. Reson. Med.*, vol. 46, pp. 638–651, 2001.

[30] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Res. Natl. Bur. Stand.*, vol. 49, pp. 409–436, 1952.

[31] C. C. Paige and M. A. Saunders, "LSQR: An algorithm for sparse linear equations and sparse least squares," *ACM Trans. Math. Soft.*, vol. 8, pp. 43–71, 1982.

[32] D. R. Hunter and K. Lange, "A tutorial on MM algorithms," *Am. Stat.*, vol. 58, pp. 30–37, 2004.

[33] M. Nikolova and M. K. Ng, "Analysis of half-quadratic minimization methods for signal and image recovery," *SIAM J. Sci. Comput.*, vol. 27, pp. 937–966, 2005.

[34] G. Ongie and M. Jacob, "A fast algorithm for convolutional structured low-rank matrix recovery," *IEEE Trans. Comput. Imaging*, vol. 3, pp. 535–550, 2017.