Tensor completion and low-n-rank tensor recovery via convex optimization

Silvia Gandy[1], **Benjamin Recht**[2] **and Isao Yamada**[1]
24 January 2011

[1] Tokyo Institute of Technology
Department of Communications and Integrated Systems
Meguro-ku, Ookayama 2-12-1-S3-60
152-8550 Tokyo, Japan

[2] University of Wisconsin
Computer Sciences Department
1210 W Dayton St Madison, WI 53706

E-mail: gandy@sp.ss.titech.ac.jp, brecht@cs.wisc.edu, isao@sp.ss.titech.ac.jp

Abstract. In this paper we consider sparsity on a tensor level, as given by the n-rank of a tensor. In the important sparse-vector approximation problem (compressed sensing) and the low-rank matrix recovery problem, using a convex relaxation technique proved to be a valuable solution strategy. Here, we will adapt these techniques to the tensor setting. We use the n-rank of a tensor as sparsity measure and consider the low-n-rank tensor recovery problem, i.e., the problem of finding the tensor of lowest n-rank that fulfills some linear constraints. We introduce a tractable convex relaxation of the n-rank and propose efficient algorithms to solve the low-n-rank tensor recovery problem numerically. The algorithms are based on the Douglas-Rachford splitting technique and its dual variant, the alternating direction method of multipliers (ADM).

Keywords: tensor completion, n-rank, nuclear norm, sparsity, Douglas-Rachford splitting, alternating direction method of multipliers (ADM)

1. Introduction

Tensors are the higher-order generalization of vectors and matrices. They have many applications in the physical, imaging and information sciences, and an in depth survey can be found in [1]. Tensor decompositions give a concise representation of the underlying structure of the tensor, revealing when the tensor-data can be modeled as lying close to a low-dimensional subspace. Tensor decompositions serve as useful tools for data summarization in numerous applications, including chemometrics, psychometrics and higher order statistics.

In this work we do not focus on finding the decomposition of a given tensor. We will try to recover a tensor by assuming that it is 'sparse' in some sense and minimize a sparsity-measure over all tensors that fit the given data. As sparsity-measure we consider the n-rank of the tensor, i.e., the ranks of the unfoldings of the tensor. The unfoldings of the tensor are conversions of the tensor into a matrix. The information we have about the tensor is modeled as the image of the underlying tensor under a known linear mapping. One example of such a map is the sampling of a subset of the entries of the tensor. This problem is called the tensor completion problem; it is a missing value estimation problem. In computer graphics missing value estimations problem are known as in-painting problems and appear for images, videos, etc.

The approaches for completing the tensor can coarsely be divided into local and global approaches. A local approach looks at neighboring pixels or voxels of a missing element and locally estimate the unknown values on basis of some difference measure between the adjacent entries. In contrast, a global approach takes advantage of a global property of the data, and is the path that we are going to take here.

For matrix-valued data, the rank of a matrix is a good notion of sparsity. As it is a non-convex function, matrix rank is difficult to minimize in general. Recently, the nuclear norm was advocated to be used as convex surrogate function for the rank function [2, 3, 4, 5]. Generalizing this program, we will study a convex surrogate for the tensor rank applied to the unfoldings of the unknown tensor. A related approach, which penalizes the unfoldings of the solution tensor to have low nuclear norm, was already presented in [6] for the special case of tensor completion. In this paper, we consider the more general low-n-rank tensor recovery setting and explicitly derive convergence guarantees for our proposed algorithms.

The paper is organized as follows. We will first fix our notation and state some basic properties of tensors in section 2. Then we will formally introduce the problem of recovering a low-n-rank tensor from partial information together with its convex relaxation. In the subsequent sections, we will derive algorithms for the tensor recovery problem. Section 4 introduces an algorithm based on the Douglas-Rachford splitting technique, whereas section 5 focuses on the application of the classical alternating direction method of multipliers (ADM), the dual version of the Douglas-Rachford algorithm. We will demonstrate the effectiveness of the algorithms in section 6. We will test the algorithms on different settings, varying from randomly generated problem instances to applications to medical images and hyperspectral data.

2. Notations & Basics on tensors

We adopt the nomenclature of Kolda and Bader's review on tensor decompositions [1].

A tensor is the generalization of a matrix to higher dimensions, i.e., $\mathbf{X} \in \mathbb{R}^{n_1 \times \ldots \times n_N}$. Let us denote the vector space of these tensors by $\mathcal{T}$, i.e., $\mathcal{T} := \mathbb{R}^{n_1 \times \ldots \times n_N}$.

The *order* N of a tensor is the number of dimensions, also known as ways or modes. A second-order tensor is a matrix and a first-order tensor is a vector. We will denote higher-order tensors by boldface letters, e.g., $\mathbf{X}$. Matrices are denoted by non-bold uppercase letters, e.g., X.

Fibers are the higher-order analogue of matrix rows and columns. A fiber is defined by fixing every index but one. The mode-n fibers are all vectors $x_{i_1 \ldots i_{n-1} : i_{n+1} \ldots i_N}$ that are obtained by fixing the values of $\{i_1, \ldots, i_N\} \setminus i_n$.

The mode-n unfolding (also called matricization or flattening) of a tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_N}$ is denoted by $X_{(n)}$ and arranges the mode-n fibers to be the columns of the resulting matrix. The tensor element $(i_1, i_2, \ldots, i_N)$ is mapped to the matrix element (i_n, j), where

$$j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} (i_k - 1) J_k \quad \text{with} \quad J_k = \prod_{\substack{m=1 \\ m \neq n}}^{k-1} n_m$$

Therefore, $X_{(n)} \in \mathbb{R}^{n_n \times I_n}$, where $I_n = \prod_{\substack{k=1 \\ k \neq n}}^{N} n_k$.

The n-rank of a N-dimensional tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_N}$ is the tuple of the ranks of the mode-n unfoldings.

$$\text{n-rank}(\mathbf{X}) = \left(\operatorname{rank} X_{(1)}, \operatorname{rank} X_{(2)}, \ldots, \operatorname{rank} X_{(N)} \right).$$

The inner product of two same-sized tensors $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_N}$ is the sum of the products of their entries, i.e.,

$$\langle \mathbf{X}, \mathbf{Y} \rangle = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_N=1}^{n_N} x_{i_1 i_2 \ldots i_N} y_{i_1 i_2 \ldots i_N}.$$

The corresponding (Frobenius-) norm is $\|\mathbf{X}\|_F = \sqrt{\langle \mathbf{X}, \mathbf{X} \rangle}$.

The *n-mode (matrix) product* of a tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times \ldots \times n_N}$ with a matrix $\Psi \in \mathbb{R}^{J \times n_n}$ is denoted by $\mathbf{X} \times_n \Psi$ and is of size $n_1 \times \ldots \times n_{n-1} \times J \times n_{n+1} \times \ldots \times n_N$. Each mode-$n$ fiber is multiplied by the matrix Ψ. This idea is compactly expressed in terms of unfolded tensors:

$$\mathbf{Y} = \mathbf{X} \times_n \Psi \quad \Leftrightarrow \quad Y_{(n)} = \Psi X_{(n)}$$

3. Recovery of low-n-rank tensors from partial information

In this section we consider the following problem: Given a linear map $\mathcal{A} : \mathbb{R}^{n_1 \times \ldots \times n_N} \to \mathbb{R}^p$ with $p \leq \prod_{i=1}^{N} n_i$ and given $b \in \mathbb{R}^p$, find the tensor $\mathbf{X}$ that fulfills the linear measurements $\mathcal{A}(\mathbf{X}) = b$ and minimizes a function of the n-rank of the tensor.

The n-rank is only one notion of tensor rank [1]. A second notion of rank is $\operatorname{rank}_{\text{CP}}(\mathbf{X})$ which defines the rank as the minimal number of rank-1 tensors that is necessary to represent the tensor. A rank-1 tensor is a tensor that is the outer product of N vectors ('$\circ$' represents the vector outer product) and

$$\operatorname{rank}_{\text{CP}}(\mathbf{X}) := \min_{r \in \mathbb{N}} \left\{ \exists v_k^{(i)}, \gamma_i \text{ s. t. } \mathbf{X} = \sum_{i=1}^{r} \gamma_i v_1^{(i)} \circ v_2^{(i)} \circ \ldots \circ v_N^{(i)} \right\}.$$

In a recent work, Lim and Common [7] discuss the uniqueness of the best approximation of a third-order tensor with a tensor of fixed value of $\operatorname{rank}_{\text{CP}} = r$.

In general such a best approximation might not exist, but by introducing conditions on the appearing rank-1 terms, uniqueness can be enforced [7].

This tensor rank is difficult to handle, as there is no straightforward algorithm to determine rank_{CP} of a specific given tensor; in fact, the problem is NP-hard [1, 8]. The n-rank on the other hand is easy to compute. Therefore, we only focus on the n-rank in this work and consider the minimization problem:

$$\underset{\mathbf{X}\in\mathcal{T}}{\textbf{minimize}} \quad f\left(\text{n-rank}(\mathbf{X})\right) \quad \textbf{s. t.} \quad \mathcal{A}\left(\mathbf{X}\right) = b,$$

where $f\left(\text{n-rank}(\mathbf{X})\right) = f\left(\text{rank}\, X_{(1)}, \text{rank}\, X_{(2)}, \ldots, \text{rank}\, X_{(N)}\right)$. In order to keep things simple, we will use the sum of the ranks of the different unfoldings in the place of the function f, i.e., $f(\text{n-rank}(\mathbf{X})) = ||\text{n-rank}(\mathbf{X})||_1 = \sum_{i=1}^{N} \text{rank}\left(X_{(i)}\right)$. This is one choice, but is also possible to incorporate a weighting, e.g., $f(\text{n-rank}(X)) = \sum_{i=1}^{N} \gamma_i \,\text{rank}\, X_{(i)}$. Thus, our minimization problem of interest becomes:

Problem setting 3.1 (**Low-n-rank tensor recovery**).

$$\underset{\mathbf{X}\in\mathcal{T}}{\textbf{minimize}} \quad \sum_{i=1}^{N} \text{rank}\left(X_{(i)}\right) \quad \textbf{s. t.} \quad \mathcal{A}\left(\mathbf{X}\right) = b \tag{1}$$

The exact form of the linear constraints on the tensor $\mathbf{X}$ is governed by the linear operator $\mathcal{A}$ which can take many possible forms in general. We would like to point out one special case, the tensor completion case. In tensor completion, a subset of the entries of the tensor $\mathbf{X}$ is given, and under the low-n-rank assumption, the unknown entries are to be deduced. Denoting the set of revealed entries by Ω, the corresponding operator $\mathcal{A}$ retrieves the values of these locations. The vector b will then contain the given entries of the tensor $\mathbf{X}$. Equivalently, we can compactly write the constraint as $\mathbf{X}_\Omega = \mathbf{T}_\Omega$, where $\mathbf{X}_\Omega$ denotes the restriction of the tensor on the entries given by Ω, and $\mathbf{T}_\Omega$ contains the values of those entries of $\mathbf{X}$.

Problem setting 3.2 (**Tensor completion**).

$$\underset{\mathbf{X}\in\mathcal{T}}{\textbf{minimize}} \quad \sum_{i=1}^{N} \text{rank}\left(X_{(i)}\right) \quad \textbf{s. t.} \quad \mathbf{X}_\Omega = \mathbf{T}_\Omega \tag{2}$$

The low-n-rank tensor recovery problem (1) (also its variant (2)) is a difficult non-convex problem. Therefore we will relax it to the following convex problem:

Problem setting 3.3 (**Low-n-rank tensor pursuit**).

$$\underset{\mathbf{X}\in\mathcal{T}}{\textbf{minimize}} \quad \sum_{i=1}^{N} ||X_{(i)}||_* \quad \textbf{s. t.} \quad \mathcal{A}(\mathbf{X}) = b. \tag{3}$$

Here, $||Z||_* = \sum_{i=1}^{n} \sigma_i(Z)$ denotes the nuclear norm, the sum of the singular values of a matrix Z. It is well known that the nuclear norm is the greatest convex minorant of the rank function [2] and therefore we chose to replace each rank term by a nuclear-norm term.

Problem (3) is equivalent to a semidefinite program. For this class of problems there exist off-the-shelf solvers that apply interior-point methods and have very good convergence guarantees. Unfortunately, the calculation of the search direction uses second-order information and scales very badly in the problem dimensions. Therefore, these solvers can only be used for very small-sized problems. As large-size problems appear naturally in tensor problems, the present work will propose first-order algorithms that are able to cope with these larger-sized problem instances.

The same objective function as in (3) was also considered in [9], but with a different focus. Their minimization problem assumes the knowledge of complete (noisy) information of a tensor and finds the best low-n-rank-approximation of this tensor. They do not consider the presence of a linear map $\mathcal{A}$. The work of [6] is also related. They use a similar formulation that uses the sum of the nuclear norms of the unfoldings as criterion and derived a specialized algorithm for the tensor completion problem. We will compare to their algorithm, namely LRTC, in section 6.

In the presence of noise, we can substitute the equality constraint by a norm-bound on the deviation from equality:

$$\underset{\mathbf{X}\in\mathcal{T}}{\textbf{minimize}} \quad \sum_{i=1}^{N} ||X_{(i)}||_* \quad \textbf{s. t.} \quad ||\mathcal{A}(\mathbf{X}) - b||_2 \leq \epsilon$$

Introducing a quadratic penalty term [10], we obtain a corresponding unconstrained formulation:

$$\underset{\mathbf{X}\in\mathcal{T}}{\textbf{minimize}} \quad \sum_{i=1}^{N} ||X_{(i)}||_* + \frac{\lambda}{2}||\mathcal{A}(\mathbf{X}) - b||_2^2 \,. \tag{4}$$

The Lagrange multiplier λ controls the fit to the constraint $\mathcal{A}(\mathbf{X}) - b$. We will be interested in ensuring the equality $\mathcal{A}(\mathbf{X}) = b$, so we will apply a continuation technique, i.e., solve (4) for increasing values of λ. In order to achieve $\|\mathcal{A}(\mathbf{X}) - b\|_2 \leq \delta$, some heuristic as used in [11], i.e., $\lambda_{\text{new}} = \lambda_{\text{old}}\delta/\|\mathcal{A}(\mathbf{X}^{(k)}) - b\|_2$, can be used.

4. Minimization via the Douglas-Rachford splitting technique

In this section we will derive an algorithm based on the Douglas-Rachford splitting technique to solve (4) efficiently. We first need to introduce some notation. Let $\Gamma_0(\mathcal{H})$ denote the class of all lower semicontinuous convex functions from a real Hilbert space $\mathcal{H}$ to $(-\infty, \infty]$ which are not identically equal to $+\infty$; let $\|.\|_{\mathcal{H}}$ denote the norm on $\mathcal{H}$.

The Douglas-Rachford splitting technique has a long history [12, 13]; it addresses the minimization of the sum of two functions $(f + g)(x)$, where f and g are assumed to be elements of $\Gamma_0(\mathcal{H})$. It was recently extended in [14] for the minimization of a sum over multiple functions in $\Gamma_0(\mathcal{H})$, based on a product space formulation. The Douglas-Rachford splitting was also identified as an instance of a Mann iteration [15].

The Douglas-Rachford splitting algorithm approximates a minimizer of $(f+g)(x)$ with the help of the following sequence $(x_n)_{n\geq 0}$:

$$x_{n+1} := x_n + t_n\big\{\text{prox}_{\gamma f}\big[2\text{prox}_{\gamma g}(x_n) - x_n\big] - \text{prox}_{\gamma g}(x_n)\big\}, \tag{5}$$

where $(t_n)_{n\geq 0} \subset [0, 2]$ satisfies $\sum_{n\geq 0} t_n(2 - t_n) = \infty$.
In [13, 14], more involved versions of this process are studied in view of unavoidable numerical errors during the calculation of the iterates.
Under certain conditions (see Theorem 4.1 for details), the iteration process (5) converges (weakly) to a point $\widetilde{x}$, which has the property that $\text{prox}_{\gamma g}(\widetilde{x})$ is a minimizer of $(f+g)(x)$. In [16], this algorithm was successfully applied to the so-called principal component pursuit problem, i.e., the problem of splitting a matrix into the sum of a low-rank matrix and a sparse matrix.
Recall, that the proximal map, $\text{prox}_{\gamma f}$, of index $\gamma \in (0, \infty)$ of $f \in \Gamma_0(\mathcal{H})$ [17, 18], is defined as

$$\text{prox}_{\gamma f} : \mathcal{H} \to \mathcal{H}, x \mapsto \underset{y\in\mathcal{H}}{\arg\min} \left\{ f(y) + \frac{1}{2\gamma}\|x - y\|_{\mathcal{H}}^2 \right\}, \tag{6}$$

where the existence and the uniqueness of the minimizer are guaranteed by the coercivity and the strict convexity of $f(.) + \frac{1}{2\gamma}\|x - .\|_{\mathcal{H}}^2$ respectively.

We will consider minimization in the Hilbert space $\mathcal{H}_0$, denoting the $(N+1)$-fold Cartesian product of $\mathcal{T}$, $\mathcal{H}_0 := \underbrace{\mathcal{T} \times \mathcal{T} \times \ldots \times \mathcal{T}}_{N+1 \text{ terms}}$, with the inner product $\langle \mathcal{X}, \mathcal{Y} \rangle_{\mathcal{H}_0} := \frac{1}{N+1}\sum_{i=0}^{N} \langle \mathbf{X}_i, \mathbf{Y}_i \rangle$ and induced norm $||\mathcal{X}||_{\mathcal{H}_0} := \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle_{\mathcal{H}_0}}$.

4.1. Douglas-Rachford splitting for n-rank-minimization

We can recast problem (4) into the unconstrained minimization of $(f+g)(x)$ as follows:

$$\underset{\mathcal{Z} \in \mathcal{H}_0}{\textbf{minimize}} \quad f(\mathcal{Z}) + g(\mathcal{Z}),$$

where $\mathcal{Z} = (\mathbf{Z}_0, \mathbf{Z}_1, \ldots, \mathbf{Z}_N)$, $D = \left\{ \mathcal{Z} \in \mathcal{H}_0 \;\middle|\; \mathbf{Z}_0 = \mathbf{Z}_1 = \ldots = \mathbf{Z}_N \right\}$, and

$$\begin{cases} f(\mathcal{Z}) := \displaystyle\sum_{i=0}^{N} f_i(\mathbf{Z}_i) = \frac{\lambda}{2}||\mathcal{A}(\mathbf{Z}_0) - b||_2^2 + \sum_{i=1}^{N} ||Z_{i,(i)}||_* \\ g(\mathcal{Z}) := i_D(\mathcal{Z}) = \begin{cases} 0, & \text{if } \mathcal{Z} \in D \\ +\infty, & \text{otherwise} \end{cases} \end{cases} \tag{7}$$

This problem formulation is equivalent to (4), so we only need to identify the proximal maps of f and g to apply algorithm (5). The proximal map of f is given by

$$\begin{aligned} \text{prox}_{\gamma f}\mathcal{X} &= \underset{\mathcal{Y} \in \mathcal{H}_0}{\arg\min} \left\{ \sum_{i=0}^{N} f_i(\mathbf{Y}_i) + \frac{1}{2\gamma}||\mathcal{Y} - \mathcal{X}||_{\mathcal{H}_0}^2 \right\} \\ &= \underset{\mathcal{Y} \in \mathcal{H}_0}{\arg\min} \left\{ \sum_{i=0}^{N} f_i(\mathbf{Y}_i) + \frac{1}{N+1}\sum_{i=0}^{N} \left(\frac{1}{2\gamma}||\mathbf{Y}_i - \mathbf{X}_i||_F^2 \right) \right\} \\ &= \left(\text{prox}_{(N+1)\gamma f_0}\mathbf{X}_0, \ldots, \text{prox}_{(N+1)\gamma f_N}\mathbf{X}_N \right) \end{aligned}$$

where we used the definition of $||.||_{\mathcal{H}_0}$ and (6).

For $i = 1 \ldots N$, the proximal map of f_i is essentially the shrinkage operator shrink(.), as it is the proximal map associated to the nuclear norm function [19]:

$$\text{prox}_{\tau||.||_*}T = \underset{Z}{\arg\min} \left\{ ||Z||_* + \frac{1}{2\tau}||Z - T||_F^2 \right\} = \text{shrink}\left(T, \tau\right), \tag{8}$$

where $\text{shrink}(T, \tau)$ denotes the shrinkage operator which applies the soft-thresholding operator to the singular values of T. The shrinkage operator acts as follows on a matrix T. Let $T = U_t \Sigma_t V_t^*$ be the singular value decomposition of T. Then, $\Sigma_t = \text{diag}(\sigma_1(T), \ldots, \sigma_r(T))$ is a diagonal matrix containing the singular values $\sigma_k(T)$ on the diagonal. Define $\widetilde{\Sigma}_t$ as the diagonal matrix that contains the singular values shrunk by τ, i.e., $\widetilde{\Sigma}_t := \text{diag}(\max\{\sigma_1(T) - \tau, 0\}, \ldots, \max\{\sigma_r(T) - \tau, 0\})$. Then, the singular value shrinkage operator is given by $\text{shrink}(T, \tau) := U_t \widetilde{\Sigma}_t V_t^*$ and we can compute

$$\begin{aligned} \text{prox}_{(N+1)\gamma f_i}\mathbf{X}_i &= \underset{\mathbf{X} \in \mathcal{T}}{\arg\min} \left\{ ||X_{(i)}||_* + \frac{1}{2(N+1)\gamma}||\mathbf{X}_i - \mathbf{X}||_F^2 \right\} \\ &= \text{refold}\left(\underset{X_{(i)}}{\arg\min} \left\{ ||X_{(i)}||_* + \frac{1}{2(N+1)\gamma}||X_{(i)} - X_{i,(i)}||_F^2 \right\} \right) \end{aligned}$$

$$= \text{refold}\left(\text{prox}_{(N+1)\gamma\|.\|_*} X_{i,(i)}\right)$$
$$= \text{refold}\left(\text{shrink}(X_{i,(i)},\ (N+1)\gamma)\right)$$

Here, refold(.), denotes the refolding of the matrix (= unfolded tensor) into a tensor. For $i = 0$, the proximal map of f_0 is given by ($\mathcal{A}^*$ denotes the adjoint operator of $\mathcal{A}$ and $\mathcal{I}$ denotes the identity operator on $\mathcal{T}$):

$$\begin{aligned}\text{prox}_{\tau f_0}\mathbf{X}_0 &= \underset{Y_0\in\mathcal{T}}{\arg\min}\left\{\frac{\lambda}{2}||\mathcal{A}(\mathbf{Y}_0)-b||_2^2+\frac{1}{2\tau}||\mathbf{Y}_0-\mathbf{X}_0||_F^2\right\}\\ &= \left(\lambda\mathcal{A}^*\mathcal{A}+\frac{1}{\tau}\mathcal{I}\right)^{-1}\left(\lambda\mathcal{A}^*b+\frac{1}{\tau}\mathbf{X}_0\right) \qquad (9)\end{aligned}$$

In the case of tensor completion, where $\mathcal{A} = \mathcal{A}_{\text{TC}}$ is a sampling operator which extracts the entries at positions given by the set Ω, the inversion in (9) reduces to an easy computation. Let $\xi = (i_1, \ldots, i_N)$, then $\mathcal{A}^*_{\text{TC}}\mathcal{A}_{\text{TC}}$ takes the form:

$$\left(\mathcal{A}^*_{\text{TC}}\mathcal{A}_{\text{TC}}(\mathbf{Z})\right)(\xi) = \begin{cases}\mathbf{Z}(\xi), & \text{if } \xi\in\Omega\\ 0, & \text{otherwise}\end{cases} \qquad (10)$$

Therefore, (9) reduces to:

$$\begin{aligned}\left(\text{prox}_{\tau f_0^{\text{TC}}}\mathbf{Z}\right)(\xi) &= \left[\left(\lambda\mathcal{A}^*_{\text{TC}}\mathcal{A}_{\text{TC}}+\frac{1}{\tau}\mathcal{I}\right)^{-1}\left(\lambda\mathcal{A}^*_{\text{TC}}b+\frac{1}{\tau}\mathbf{Z}\right)\right](\xi)\\ &= \begin{cases}\left(\lambda+\frac{1}{\tau}\right)^{-1}\left(\lambda\mathcal{A}^*_{\text{TC}}b+\frac{1}{\tau}\mathbf{Z}\right)(\xi), & \text{if } \xi\in\Omega\\ \tau\frac{1}{\tau}\mathbf{Z}(\xi), & \text{otherwise}\end{cases}\\ &= \begin{cases}\frac{\tau}{\lambda\tau+1}\left(\lambda\mathcal{A}^*_{\text{TC}}b+\frac{1}{\tau}\mathbf{Z}\right)(\xi), & \text{if } \xi\in\Omega\\ \mathbf{Z}(\xi), & \text{otherwise}\end{cases}\end{aligned}$$

where we used that $(\mathcal{A}^*_{\text{TC}}b)(\xi) = 0$ for all $\xi \notin \Omega$.

For general $\mathcal{A}$, it can be helpful to apply the so-called Sherman-Morrison-Woodbury formula [20] which expresses the inverse of a matrix M that underwent a rank-one correction vv^T as a rank-one correction of M^{-1}:

$$\left(M+vv^T\right)^{-1} = M^{-1} - M^{-1}v\left(1+v^TM^{-1}v\right)^{-1}v^TM^{-1},$$

The matrix $\lambda\mathcal{A}^*_{\text{TC}}\mathcal{A}_{\text{TC}} + \frac{1}{\tau}\mathcal{I}$ can be written as p rank-one corrections of $\frac{1}{\tau}\mathcal{I}$:

$$\lambda\mathcal{A}^*_{\text{TC}}\mathcal{A}_{\text{TC}}+\frac{1}{\tau}\mathcal{I} = \frac{1}{\tau}\mathcal{I}+\sum_{i=1}^{p}(A^{(i)})^TA^{(i)}, \qquad (11)$$

where $A^{(i)}$ denotes the i-th row vector of the matrix representation of $\mathcal{A}$. Thus the inversion can be performed as a series of matrix multiplications.

Another approach is to use some inner structure, e.g., block structure, of $\mathcal{A}$ if applicable. If for example $\mathcal{A}(\mathbf{X})-b$ can be expressed as $\mathcal{A}(\mathbf{X})-b = \sum_{i\in J}(\mathcal{A}_i(\mathbf{X})-b_i)$, then the term $\|\mathcal{A}(\mathbf{X})-b\|_2^2$ can be expressed as sum of $|J|$ terms depending on $\mathcal{A}_i$ and b_i. When applying the variable splitting as in (7), the function f then has $N+|J|$ terms. The inversion problem with respect to $\mathcal{A}$ as in (9) then results in $|J|$ separate inversion problems with respect to $\mathcal{A}_i$, which can be beneficial if the operators $\mathcal{A}_i$ have nice properties.

Finally, the proximal map of the indicator function i_D is the orthogonal projection $\mathcal{P}_D$ onto the set D. It is given by:

$$\mathcal{P}_D(\mathcal{Z}) = [\mathcal{I}_\mathcal{T}, \ldots, \mathcal{I}_\mathcal{T}]\cdot\frac{1}{N+1}\sum_{i=0}^{N}\mathbf{Z}_i = [\mathcal{I}_\mathcal{T}, \ldots, \mathcal{I}_\mathcal{T}]\cdot\text{mean}(\mathcal{Z}),$$

where $\mathcal{I}_\mathcal{T}$ is the identity operator on $\mathcal{T}$ and we define $\text{mean}(\mathcal{Z}) := \frac{1}{N+1}\sum_{i=0}^{N}(\mathbf{Z}_i)$.

Having identified all ingredients, we can now specialize the Douglas-Rachford splitting algorithm of (5) to this choice of f and g, (7). Here, c_λ controls the increase of the Lagrange multiplier λ:

(DR-TR): Douglas-Rachford splitting for Tensor Recovery
input: $\mathcal{A}$, b, t_k, λ, c_λ,γ initialization: $\mathcal{X}^{(0)} = (0,\ldots,0), k = 0, \gamma' = (N+1)\gamma$
repeat until convergence: if subproblem solved: $\lambda = c_\lambda\lambda$ $\widehat{\mathbf{X}} = \text{mean}(\mathcal{X}^{(k)}) = \text{mean}\left((\mathbf{X}_0^{(k)}, \mathbf{X}_1^{(k)}, \ldots, \mathbf{X}_N^{(k)})\right)$ for $i = 1$ to N do $\quad \mathbf{X}_i^{(k+1)} = \mathbf{X}_i^{(k)} + t_k\left(\text{refold}\left(\text{shrink}(2\widehat{X}_{(i)} - X_{i,(i)}^{(k)}, \gamma')\right) - \widehat{\mathbf{X}}\right)$ end $\mathbf{X}_0^{(k+1)} = \mathbf{X}_0^{(k)} + t_k\left(\text{prox}_{\gamma' f_0}(2\widehat{\mathbf{X}} - \mathbf{X}_0^{(k)}) - \widehat{\mathbf{X}}\right)$ k = k+1
output: $\mathbf{X} = \text{mean}(\mathcal{X}^{(k)})$

4.2. Convergence of the Douglas-Rachford splitting

In order to state a convergence theorem of the Douglas-Rachford splitting algorithm, we need the notion of domain ($\text{dom}(f)$) of a function f:
Define $\text{dom}(f) := \{x \in \mathcal{H} \mid f(x) < \infty\}$ and

$$\text{dom}(f) - \text{dom}(g) := \left\{x_1 - x_2 \in \mathcal{H} \;\middle|\; x_1 \in \text{dom}(f) \wedge x_2 \in \text{dom}(g)\right\}.$$

The following theorem states the convergence properties of the Douglas-Rachford splitting algorithm (5):

Theorem 4.1 ([13, 14, 15])**.** *Let $f, g \in \Gamma_0(\mathcal{H})$ satisfy $S := \arg\min_{x\in\mathcal{H}}\{f(x) + g(x)\} \neq \emptyset$. Suppose that*

$$\text{cone}(\text{dom}(f) - \text{dom}(g)) := \bigcup_{\lambda>0}\left\{\lambda x \mid x \in \text{dom}(f) - \text{dom}(g)\right\} \tag{12}$$

is a closed subspace of $\mathcal{H}$. Then the sequence $(x_n)_{n=0}^{\infty}$ generated by (5) converges weakly to a point in $(\text{prox}_{\gamma g})^{-1}(S)$. This holds for any initial value $x_0 \in \mathcal{H}$, any $\gamma \in (0,\infty)$ and any $(t_n)_{n\geq 0} \subset [0,2]$ that satisfies $\sum_{n\geq 0} t_n(2-t_n) = \infty$.

We will show that the condition on (12) holds for the DR-TR algorithm.

Lemma 4.2. *The qualifying condition on (12), i.e. the condition that the set defined in (12) is a closed subspace, holds for the functions f and g as defined in (7).*

Proof. The domain of f, $\text{dom}(f)$, is the whole space $\mathcal{H}_0$, as $f(\mathcal{Z}) = \frac{\lambda}{2}||\mathcal{A}(\mathbf{Z}_0) - b||_2^2 + \sum_{i=1}^{N}||Z_{i,(i)}||_* < \infty$, $\forall \mathcal{Z} \in \mathcal{H}_0$. From (7) it follows immediately that

$$\text{dom}(g) = \{\mathcal{X} \mid \mathbf{X}_0 = \mathbf{X}_1 = \ldots = \mathbf{X}_N\} \neq \emptyset.$$

Thus, $\text{dom}(f) - \text{dom}(g) = \mathcal{H}_0$, therefore $\text{cone}(\mathcal{H}_0) = \mathcal{H}_0$ and the qualifying condition on (12) holds. □

Applying Lemma 4.2 to Theorem 4.1, we can now state the convergence of algorithm (DR-TR):

Theorem 4.3. *Let all parameters of the algorithm (DR-TR) be chosen as in Theorem 4.1, let $c_\lambda = 1$. Then, mean($\mathbf{X}^{(k)}$) converges to a minimizer of (4).*

5. Alternative approach: ADM

In this section we will derive a second algorithm for the solution of Problem 3.1 based on the classical ADM method. ADM goes back to the seventies [21, 22] It is known that ADM is an application of the Douglas-Rachford splitting to the dual problem of the minimization of $(f + g)(x)$ [23, 24]. With this in mind, the algorithm ADM-TR which we derive in the following is also a Douglas-Rachford-type algorithm. We chose to present ADM as separate algorithm as it simplifies the presentation.

The minimization problem that ADM solves is [25]:

$$\underset{x\in\mathbb{R}^q, y\in\mathbb{R}^m}{\textbf{minimize}} \quad f(x)+g(y) \quad \textbf{s. t.} \quad x\in C_x,\ y\in C_y,\ Gx=y \tag{13}$$

Here, $f : \mathbb{R}^q \to \mathbb{R}$ and $g : \mathbb{R}^m \to \mathbb{R}$ are convex functions, G is a $m \times q$ matrix and $C_x \subset \mathbb{R}^q$ and $C_y \subset \mathbb{R}^m$ are nonempty polyhedral sets. (A polyhedral set P is one that is specified by a finite collection of linear inequalities).

By introducing a Lagrange multiplier $w \in \mathbb{R}^m$ to the equality constraint $Gx = y$, we can consider the augmented Lagrangian function

$$\mathcal{L}_A(x, y, w) = f(x) + g(y) - \langle w, Gx - y\rangle + \frac{\beta}{2}||Gx - y||_2^2.$$

The alternating direction method of multipliers is given by ([25], eq.(4.79)-(4.81)):

$$\begin{cases} x^{(k+1)} & \leftarrow \underset{x\in C_x}{\arg\min}\ \mathcal{L}_A(x, y^{(k)}, w^{(k)}) \\ y^{(k+1)} & \leftarrow \underset{y\in C_y}{\arg\min}\ \mathcal{L}_A(x^{(k+1)}, y, w^{(k)}) \\ w^{(k+1)} & \leftarrow w^{(k)} - \beta(Gx^{(k+1)} - y^{(k+1)}) \end{cases} \tag{14}$$

The parameter β is any positive number, and the initial vectors $w^{(0)}$ and $y^{(0)}$ are arbitrary. The advantage of these minimization problems (compared to a direct joint minimization of $\arg\min_{(x,y)} \mathcal{L}_A(x, y, w)$) is that the functions f and g and the constraint sets C_x and C_y have been decoupled.

Theorem 5.1 ([25], Proposition 4.2)**.** *Assume that the optimal solution set $X^* \subset \mathbb{R}^q \times \mathbb{R}^m$ of problem (13) is nonempty. Furthermore, assume that either C_x is bounded or else that the matrix G^*G is invertible.*
A sequence $\{x^{(k)}, y^{(k)}, w^{(k)}\}$ generated by algorithm (14) is bounded, and every limit point of $\{x^{(k)}\}$ is an optimal solution of the original problem (13).

5.1. Rephrasing the low-n-rank tensor pursuit as ADM problem

In order to apply the ADM method to problem (4), we need to transform it into the form $f(x)+g(y)$ of separate variables x and y. Therefore we perform variable splitting and attribute a separate variable to each unfolding of $\mathbf{X}$.

Let $\mathbf{Y}_1, \ldots, \mathbf{Y}_N$ be new tensor-valued variables, which represent the N different mode-n unfoldings $X_{(1)}, \ldots X_{(N)}$ of the tensor $\mathbf{X}$, i.e., introduce the new variables:

$$\mathbf{Y}_i \in \mathbb{R}^{n_1\times\ldots\times n_N} \quad \text{such that} \quad Y_{i,(i)} = X_{(i)},\ \forall i \in \{1, \ldots, N\}.$$

With these new variables $\mathbf{Y}_i$, we can rephrase (4) as follows:

$$\begin{aligned} \underset{\mathbf{X},\mathbf{Y}_i\in\mathcal{T}}{\textbf{minimize}} \quad & \sum_{i=1}^{N} ||Y_{i,(i)}||_* + \frac{\lambda}{2}||\mathcal{A}(\mathbf{X}) - b||_2^2 \\ \textbf{subject to} \quad & \mathbf{Y}_i = \mathbf{X} \quad \forall i \in \{1,\dots,N\} \end{aligned} \tag{15}$$

It becomes clear that (15) has the same structure as (13) when first defining $\mathcal{K} := \underbrace{\mathcal{T}\times\mathcal{T}\times\ldots\times\mathcal{T}}_{N \text{ times}}$, $\mathcal{Z}\in\mathcal{K}, \mathcal{Z} = (\mathbf{Z}_1,\dots,\mathbf{Z}_N)^T$. Then we can define the constraint sets $C_x := \mathcal{T}$, $C_y := \mathcal{K}$, and the functions $f : \mathcal{T}\to\mathbb{R}$ and $g : \mathcal{K}\to\mathbb{R}$:

$$\begin{cases} f(\mathbf{X}) := \dfrac{\lambda}{2}||\mathcal{A}(\mathbf{X}) - b||_2^2, \\ g(\mathcal{Y}) := \displaystyle\sum_{i=1}^{N} g_i(\mathbf{Y}_i) = \sum_{i=1}^{N} ||Y_{i,(i)}||_* . \end{cases}$$

The coupling between $\mathbf{X}$ and $\mathcal{Y}$ is given by

$$\mathbf{X} = \mathbf{Y}_i \ \forall i \ \Rightarrow \mathcal{Y} = [\mathcal{I}_\mathcal{T},\dots,\mathcal{I}_\mathcal{T}]^T\,\mathbf{X} = G\mathbf{X},$$

where $\mathcal{I}_\mathcal{T}$ is the identity operator on $\mathcal{T}$. We define the inner product and induced norm on the product space $\mathcal{K}$ as $\langle\mathcal{X},\mathcal{Y}\rangle_\mathcal{K} := \sum_{i=1}^{N}\langle\mathbf{X}_i,\mathbf{Y}_i\rangle_F$ and $||\mathcal{X}||_\mathcal{K} := \sqrt{\sum_{i=1}^{N}||\mathbf{X}_i||_F^2}$. In (13), the functions are defined over real-valued vectors. However, with our definition of the inner product and the norm on $\mathcal{K}$, we can identify the elements of $\mathcal{K}$ with their vectorized form (rewrite the entries of the set of tensors as one long vector). The inner product on $\mathcal{K}$ then maps isometrically to the usual vector product and the norm becomes the vector-ℓ_2-norm. Therefore, minimization over $(\mathcal{T},\mathcal{K})$ is equivalent to minimization over $(\mathbb{R}^q,\mathbb{R}^m)$, where $q = \prod_{i=1}^{N} n_i$ equals the number of entries of an element of $\mathcal{T}$ and $m = Nq$ is the number of entries of N tensors that form an element in $\mathcal{K}$.

The augmented Lagrangian of (15) becomes

$$\begin{aligned} \mathcal{L}_A(\mathbf{X},\mathcal{Y},\mathcal{W}) &= f(\mathbf{X}) + g(\mathcal{Y}) - \langle\mathcal{W}, G\mathbf{X}-\mathcal{Y}\rangle_\mathcal{K} + \frac{\beta}{2}||G\mathbf{X}-\mathcal{Y}||_\mathcal{K} \\ &= \frac{\lambda}{2}||\mathcal{A}(\mathbf{X}) - b||_2^2 + \sum_{i=1}^{N}\left(||Y_{i,(i)}||_* - \langle\mathbf{W}_i, \mathbf{X}-\mathbf{Y}_i\rangle + \frac{\beta}{2}||\mathbf{X}-\mathbf{Y}_i||_F^2\right) \end{aligned}$$

We can now directly apply ADM with this augmented Lagrangian function.

5.2. ADM: update step for the $\mathbf{Y}$*-variables*

First, we will discuss the minimization of $\mathcal{L}_A(\mathcal{X},\mathcal{Y},\mathcal{W})$ with respect to the variable $\mathcal{Y}$. By noticing that the function $g(\mathcal{Y})$ is a sum of independent non-negative functions $g_i(\mathbf{Y}_i)$, we can find the minimizer of g by finding each minimizer of $g_i(\mathbf{Y}_i)$ separately. So let us pick one of the variables $\mathbf{Y}_i$, say $\mathbf{Y}_j$, and let us consider all other variables $(\mathbf{X},\mathcal{W},\mathbf{Y}_1,\dots,\mathbf{Y}_{j-1},\mathbf{Y}_{j+1},\dots,\mathbf{Y}_N)$ to be constant. We will calculate the minimizer in the unfolded regime, paralleling the calculation of the proximal map for the Douglas-Rachford splitting.

$$\begin{aligned} &\left(\underset{\mathbf{Y}_j\in\mathcal{T}}{\arg\min}\ \mathcal{L}_A(\mathbf{Y}_j)\right)_{(j)} \\ &= \underset{Y_{j,(j)}}{\arg\min}\left\{||Y_{j,(j)}||_* - \langle W_{j,(j)}, X_{(j)} - Y_{j,(j)}\rangle + \frac{\beta}{2}||X_{(j)} - Y_{j,(j)}||_F^2\right\} \end{aligned}$$

$$= \underset{Y_{j,(j)}}{\arg\min} \left\{ \frac{1}{\beta} ||Y_{j,(j)}||_* + \frac{1}{2} \left|\left| Y_{j,(j)} - \left(X_{(j)} - \frac{1}{\beta} W_{j,(j)} \right) \right|\right|_F^2 + \text{const.} \right\}$$
$$= \text{shrink}\left(X_{(j)} - \frac{1}{\beta} W_{j,(j)} \ , \ \frac{1}{\beta} \right)$$

Here, we completed the square and applied (8) (for $X_{(j)}$ and $W_{j,(j)}$ constant).

The minimizer $(\mathbf{Y}_j)_{\min}$ is thus

$$(\mathbf{Y}_j)_{\min} = \text{refold}\left(\text{shrink}\left(\mathbf{X}_{(j)} - \frac{1}{\beta} \mathbf{W}_{j,(j)} \ , \ \frac{1}{\beta} \right) \right).$$

Its calculation contains one application of the shrinkage operator followed by a refolding of the resulting matrix into a tensor. The application of the shrinkage operator requires the calculation of one (partial) singular value decomposition.

The derivation was completely independent of the choice of j, so the above minimization can be performed for any $\mathbf{Y}_i$, $i \in \{1, \ldots, N\}$. The minimizer $\mathcal{Y}^{(k+1)}$ is therefore

$$\mathcal{Y}^{(k+1)} = ((\mathbf{Y}_1)_{\min}, \ldots, (\mathbf{Y}_N)_{\min})^T .$$

5.3. ADM: update step for the $\mathbf{X}$*-variable – exact version*

Now we will fix all variables except $\mathbf{X}$ and minimize $\mathcal{L}_A$ over $\mathbf{X}$. The resulting minimization problem is the minimization of a quadratic function:

$$\underset{\mathbf{X} \in \mathcal{T}}{\textbf{minimize}} \quad \mathcal{L}_A(\mathbf{X}) = \frac{\lambda}{2} ||\mathcal{A}(\mathbf{X}) - b||_2^2 - \sum_{i=1}^N \langle \mathbf{W}_i, \mathbf{X} - \mathbf{Y}_i \rangle + \sum_{i=1}^N \frac{\beta}{2} ||\mathbf{X} - \mathbf{Y}_i||_F^2, \tag{16}$$

The objective function is differentiable, so the minimizer $\mathbf{X}_{\min}$ is characterized by $\frac{\partial \mathcal{L}_A(\mathbf{X})}{\partial \mathbf{X}} = 0$. The gradient of $\mathcal{L}_A$ is given by:

$$\frac{\partial \mathcal{L}_A(\mathbf{X})}{\partial \mathbf{X}} = \lambda \mathcal{A}^* \left(\mathcal{A}(\mathbf{X}) - b \right) - \sum_{i=1}^N \mathbf{W}_i + N\beta \mathbf{X} - \sum_{i=1}^N \beta \mathbf{Y}_i$$

Thus, by setting the above formula equal to zero, we obtain

$$\mathbf{X}_{\min} = \left(\lambda \mathcal{A}^* \mathcal{A} + N\beta \mathcal{I} \right)^{-1} \left[\sum_{i=1}^N \mathbf{W}_i + \sum_{i=1}^N \beta \mathbf{Y}_i + \lambda \mathcal{A}^* b \right]. \tag{17}$$

In the case of tensor completion we can use (10) and readily calculate the inverse:

$$\left[\left(\lambda \mathcal{A}_{\text{TC}}^* \mathcal{A}_{\text{TC}} + N\beta \mathcal{I} \right)^{-1} (\mathbf{Z}) \right](\xi) = \begin{cases} (\lambda + N\beta)^{-1} \mathbf{Z}(\xi), & \text{if } \xi \in \Omega \\ (N\beta)^{-1} \mathbf{Z}(\xi), & \text{otherwise} \end{cases}$$

Thus, (17) can be computed as

$$\mathbf{X}_{\min}^{TC}(\xi) = \begin{cases} (\lambda + N\beta)^{-1} \left[\sum_{i=1}^N \mathbf{W}_i + \sum_{i=1}^N \beta \mathbf{Y}_i + \lambda \mathcal{A}^* b \right](\xi), & \text{if } \xi \in \Omega \\ (N\beta)^{-1} \left[\sum_{i=1}^N \mathbf{W}_i + \sum_{i=1}^N \beta \mathbf{Y}_i \right](\xi), & \text{otherwise} \end{cases}$$

Therefore, the update for $\mathbf{X}^{(k)}$ is simply (note that for general $\mathcal{A}$, (11) is applicable):

Exact Version: Update of $\mathbf{X}^{(k)}$
$\mathbf{X}^{(k+1)} = \begin{cases} \mathbf{X}_{\min}^{\mathrm{TC}}, & \text{if } \mathcal{A} == \mathcal{A}_{\mathrm{TC}}, \\ \mathbf{X}_{\min}, & \text{otherwise.} \end{cases}$

5.4. ADM: update step for the $\mathbf{X}$*-variable – inexact version*

The exact calculation of the minimizer of $\mathcal{L}_A(\mathbf{X})$, $\mathbf{X}_{\min}$, via (17) can be difficult to handle. The matrix representation of $\mathcal{A}$ might not be accessible and the matrix-inversion difficult to compute or expensive to store. In these cases, it can be beneficial to use the following approach. Instead of the minimizer $\mathbf{X}_{\min}$, we will just determine a value of $\mathbf{X}$ that achieves a lower value of the augmented Lagrangian function. For example, a gradient descent method can be applied with respect to $\mathbf{X}$.

Here we will introduce such an inexact scheme without giving any convergence guarantees. The augmented Lagrangian is a differentiable function in $\mathbf{X}$, so we use a gradient step. The step-size will be chosen by the method of Barzilai-Borwein[26] which uses the idea of mimicking the unavailable second order information by approximating the secant equation.
The gradient step is given by: $\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} - s\nabla\mathcal{L}_A^{(k)}(\mathbf{X}^{(k)})$,
where $\nabla\mathcal{L}_A^{(k)}(\mathbf{X}^{(k)}) := \frac{\partial\mathcal{L}_A^{(k)}(\mathbf{X})}{\partial\mathbf{X}}\Big|_{\mathbf{X}^{(k)}}$ is the gradient of $\mathcal{L}_A^{(k)}(\mathbf{X}, \mathcal{Y}^{(k)}, \mathcal{W}^{(k)})$ at $\mathbf{X}^{(k)}$.
In order to state the step-size-rule, we need to introduce the values

$$\Delta X := \mathbf{X}^{(k)} - \mathbf{X}^{(k-1)} \quad \text{and} \quad \Delta g := \nabla\mathcal{L}_A^{(k)}(\mathbf{X}^{(k)}) - \nabla\mathcal{L}_A^{(k-1)}(\mathbf{X}^{(k-1)}).$$

In each iteration, the step-size s is chosen adaptively as

$$s = \frac{|\langle \Delta X \ , \ \Delta g\rangle|}{\langle \Delta g \ , \ \Delta g\rangle} = \frac{|\langle \Delta X \ , \ \Delta g\rangle|}{||\Delta g||^2}.$$

We further used an acceleration technique to combine the previous iterate $\mathbf{X}^{(k)}$ with an intermediate value $\widetilde{\mathbf{X}}$, to obtain the next value $\mathbf{X}^{(k+1)}$. The way of combining these two values parallels the acceleration used in Nesterov's optimal methods and later used in the FISTA algorithm for sparse vector recovery[27].

The intermediate value $\widetilde{\mathbf{X}}$ will be the result of the gradient step with s chosen by the Barzilai-Borwein step-size-rule. The acceleration is given by the following equations, where we set $t_1 = 1$ and $t_{k+1} := \frac{1}{2}\left(1 + \sqrt{1 + 4t_k^2}\right)$:

$$\begin{aligned} \widetilde{\mathbf{X}} &= \mathbf{X}^{(k)} - s\nabla\mathcal{L}_A^{(k)}(\mathbf{X}^{(k)}) \\ \mathbf{X}^{(k+1)} &= \widetilde{\mathbf{X}} + \frac{t_k - 1}{t_{k+1}}\left(\widetilde{\mathbf{X}} - \mathbf{X}^{(k)}\right) \end{aligned}$$

The value $\mathbf{X}^{(k+1)}$ is not the minimizer of $\mathcal{L}_A^{(k)}(\mathbf{X})$, but a monotone decrease in the augmented Lagrangian function, i.e., $\mathcal{L}_A^{(k)}(\mathbf{X}^{(k+1)}) < \mathcal{L}_A^{(k)}(\mathbf{X}^{(k)})$, is achieved when no acceleration is used. We added the acceleration as it empirically improves the convergence rate. Putting these steps together, the inexact update of $\mathbf{X}$ becomes:

Inexact Version: Update of $\mathbf{X}^{(k)}$
input: $\mathbf{X}^{(k)}, \lambda, \beta, \mathbf{Y}_i^{(k)}, \mathbf{W}_i^{(k)}, \mathcal{A}, b$
$\nabla \mathcal{L}_A^{(k)}(\mathbf{X}^{(k)}) = \lambda \mathcal{A}^* \left(b - \mathcal{A}(\mathbf{X}^{(k)}) \right) + \sum_{i=1}^{N} \mathbf{W}_i{}^{(k)} - N\beta \mathbf{X}^{(k)} + \sum_{i=1}^{N} \beta \mathbf{Y}_i^{(k)}$ $\Delta g = \nabla \mathcal{L}_A^{(k)}(\mathbf{X}^{(k)}) - \nabla \mathcal{L}_A^{(k-1)}(\mathbf{X}^{(k-1)}), \quad s = \frac{\lvert \langle \mathbf{X}^{(k)} - \mathbf{X}^{(k-1)},\ \Delta g \rangle \rvert}{\langle \Delta g\ ,\ \Delta g \rangle},$ $\widetilde{\mathbf{X}} = \mathbf{X}^{(k)} - s \nabla \mathcal{L}_A^{(k)}(\mathbf{X}^{(k)}), \quad t_{k+1} = \frac{1}{2} \left(1 + \sqrt{1 + 4t_k^2} \right)$
output: $\mathbf{X}^{(k+1)} = \widetilde{\mathbf{X}} + \frac{t_k - 1}{t_{k+1}} \left(\widetilde{\mathbf{X}} - \mathbf{X}^{(k)} \right)$

5.5. ADM algorithm for low-n-rank tensor recovery

After discussing the minimization of the appearing subproblems, we are now in the position to present the complete ADM algorithm which we named ADM-TR.

The algorithm uses as input the linear operator $\mathcal{A}$, the measurements b and the parameters β, λ, c_β and c_λ. It iteratively minimizes (15) for increasing parameters of λ and β. When the inner iterations indicate a slow change per iteration, these parameters are increased by constant factors c_β and c_λ.

The loop is the application of the iterations of the ADM method (14) to (15). The update of $\mathbf{X}$ is optimally performed via the exact update as described in section 5.3. If the inverse function in the update rule for $\mathbf{X}$ should be difficult to calculate, an inexact update step, as given in section 5.4 can be used.

(ADM-TR): ADM algorithm for low-n-rank tensor recovery
input: $\mathcal{A}, b, \beta, \lambda, c_\beta, c_\lambda$ initialization: $\mathbf{X}^{(0)} = \mathbf{Y}_i^{(0)} = \mathbf{W}_i^{(0)} = 0,\ \forall i \in \{1, \ldots, N\},\ k = 0$
repeat until convergence if subproblem solved: $\beta = c_\beta \beta, \quad \lambda = c_\lambda \lambda$ $\mathbf{X}^{(k+1)} = \ldots$ % use exact or inexact update step for $i = 1 : N$ $\quad \mathbf{Y}_i^{(k+1)} = \text{refold} \left(\text{shrink} \left(\mathbf{X}_{(i)}^{(k+1)} - \frac{1}{\beta} \mathbf{W}_{i,(i)}^{(k)},\ \frac{1}{\beta} \right) \right)$ $\quad \mathbf{W}_i{}^{(k+1)} = \mathbf{W}_i{}^{(k)} - \beta \left(\mathbf{X}^{(k+1)} - \mathbf{Y}_i^{(k+1)} \right)$ end k = k+1
output: $\mathbf{X}^{(k)}$

Theorem 5.2 (Convergence of ADM-TR (E)). *Assume that the optimal solution set $\mathbf{X}^*$ of problem (4) is nonempty. A sequence $\{\mathbf{X}^{(k)}, \mathcal{Y}^{(k)}, \mathcal{W}^{(k)}\}$ generated by ADM-TR (E), i.e., the ADM-TR algorithm with the exact update for $\mathbf{X}^{(k+1)}$, $c_\lambda = c_\beta = 1$, is bounded, and every limit point of $\{\mathbf{X}^{(k)}\}$ is an optimal solution of the original problem (4).*

Proof. We check the assumptions of Theorem 5.1. C_x is not bounded, but G^*G is invertible as $G^*G = N\mathcal{I}_\mathcal{T}$ is a constant multiple of the identity operator on $\mathcal{T}$. Thus Theorem 5.1 can be applied to ADM-TR (E) and Theorem 5.2 follows. □

6. Numerical experiments

In this section, we evaluate the empirical performance of the proposed algorithms. We performed experiments in the tensor completion setup, where we first used randomly generated input data. We compared the results with other algorithms, namely LRTC (Low Rank Tensor Completion) [6] and the *N-way toolbox for MATLAB*. Then we applied the algorithms to the completion of third-order tensors, taken from medical imaging (MRI scans of parts of the human body) and hyperspectral data.

Remark: The computational cost of the algorithms DR-TR and ADM-TR relates as follows to the size of $n_1, \ldots, n_N$ and the size of N. The sizes n_i determine the size of the unfolded tensors and thereby influence the computational cost of one (partial) singular value decomposition which is the main operation within the shrinkage operator. A singular value decomposition of a matrix of dimensions $n \times n$ has complexity $O(n^3)$ [20]. However, using the Lanczos algorithm for computing only the singular vectors corresponding to the largest singular values speeds up the calculation [28, 29]. The order N governs the number of different unfoldings of the tensor. In total there are N shrinkage operations applied per iteration of the algorithms.

Experiments with randomly generated problem settings: In each experiment we generated a low-n-rank tensor $\mathbf{X}_0$ which we used as ground truth. Therefore, we fixed the dimension r of a 'core tensor' $S \in \mathbb{R}^{r \times \cdots \times r}$ which we filled with Gaussian distributed entries ($\sim \mathcal{N}(0,1)$). Then, we generated matrices $\Psi^{(1)}, \ldots, \Psi^{(N)}$, with $\Psi^{(i)} \in \mathbb{R}^{n_i \times r}$ and set

$$\mathbf{X}_0 := S \times_1 \Psi^{(1)} \times_2 \ldots \times_N \Psi^{(N)}.$$

With this construction, the n-rank of $\mathbf{X}_0$ equals $(r, r, \ldots, r)$ almost surely. We fixed a percentage ρ_s of the entries to be known and chose the support of the known entries uniformly at random among all supports of size $\rho_s \prod_{i=1}^{N} n_i$. The values and the locations of the known entries of $\mathbf{X}_0$ was used as input for the algorithms.

We used four different settings to test the algorithms. The order of the tensors varied from three to five, and we also varied the n-rank and the fraction ρ_s of known entries. Table 1 shows these different settings and the recovery performance for different algorithms. The parameters were set to $c_\beta = 2, c_\lambda = 2, \beta = 1, \lambda = N, \gamma' = 1/\beta$ and $t_k = 1$. We compared the ADM algorithm with inexact (ADM-TR (IE)) and exact (ADM-TR (E)) update rule for $\mathbf{X}$, the Douglas-Rachford splitting for tensor recovery (DR-TR) and the *N-way toolbox for MATLAB*.

The *N-way toolbox for MATLAB* fits a tensor of a given n-rank $[r_1, r_2, \ldots, r_N]$ to the given input data $\mathbf{T}_\Omega$, i.e., it essentially solves:

$$\textbf{minimize} \quad ||\mathbf{X}_\Omega - \mathbf{T}_\Omega||_F \quad \textbf{s. t.} \quad \text{n-rank}(X) = [r_1, r_2, \ldots, r_N]$$

In table 1 we show the results for two different initializations of the N-way toolbox. In N-way-E, we provided the exact n-rank of the ground-truth tensor $\mathbf{X}_0$ to the algorithm. As this is a very strong extra information, we also run the N-way toolbox providing it an incorrect model information. In the setting N-way-IM, we told the toolbox to fit a tensor whose unfoldings have rank: $\text{rank}(\mathbf{X}_{(i)}) = \text{rank}((\mathbf{X}_0)_{(i)}) + 1$.

The experiments in table 1 can be considered to vary from small-sized problems to large-sized problems. Note that the number of entries of a tensor of dimensions $20 \times 20 \times 20 \times 20 \times 20$ has $20^5 = 3,2$ million entries. The computations were performed

Table 1: Comparison of different algorithms for tensor completion

$\mathcal{T} = \mathbb{R}^{20\times20\times20\times20\times20}$, $\rho_s = 0.2$, $r = 2$				$\mathcal{T} = \mathbb{R}^{20\times20\times20\times20\times20}$, $\rho_s = 0.3$, $r = 2$			
algorithm	error	# iter	time (s)	algorithm	error	# iter	time (s)
ADM-TR (IE)	2.54e-4	2000	3676	ADM-TR (IE)	5.3e-4	509	3518
DR-TR	1.06e-5	1663	9819	DR-TR	1.1e-5	619	5630
ADM-TR (E)	1.89e-7	598	4033	ADM-TR (E)	7.8e-8	386	2587
N-way-E	2.8e-6	61	434	N-way-E	2.0e-6	40	284
N-way-IM	0.022	195	1482	N-way-IM	0.017	72	552

$\mathcal{T} = \mathbb{R}^{50\times50\times50\times50}$, $\rho_s = 0.4$, $r = 4$				$\mathcal{T} = \mathbb{R}^{20\times30\times40}$, $\rho_s = 0.6$, $r = 2$			
algorithm	error	# iter	time (s)	algorithm	error	# iter	time (s)
ADM-TR (IE)	1.9e-4	1381	16884	ADM-TR (IE)	6.2e-4	2000	138
DR-TR	1.1e-5	624	8562	DR-TR	2.0e-6	627	43
ADM-TR (E)	3.8e-8	259	3983	ADM-TR (E)	2.0e-9	205	16
N-way-E	8.0e-7	28	180	N-way-E	1.7e-6	26	1.7
N-way-IM	0.0085	36	251	N-way-IM	0.12	279	21

on standard desktop machines, equipped with a dual core 2.66 GHz processor and 8 GByte of memory.

All settings except N-way-IM returned a tensor that had a relative deviation of the ground-truth tensor $\mathbf{X}_0$ of less than 10^{-3}. The N-way toolbox with exact n-rank information outperformed our algorithms, but it depends strongly on the knowledge of the n-rank of the underlying tensor. When the mismatch to the correct n-rank was introduced, the tensor could no longer be recovered. This shows the sensitivity of the N-way toolbox on the model initialization.

Our algorithms do not use any information of the n-rank of the underlying tensor. ADM-TR (E) performed best, converging faster than both DR-TR and ADM-TR (IE). The inexact ADM algorithm performed worst in general. Applying it repeatedly for the same problem dimensions and setup, its performance varied a lot whereas ADM-TR (E) and DR-TR needed an almost constant number of iterations. For some examples, ADM-TR (IE) was faster than DR-TR but never faster than ADM-TR (E).

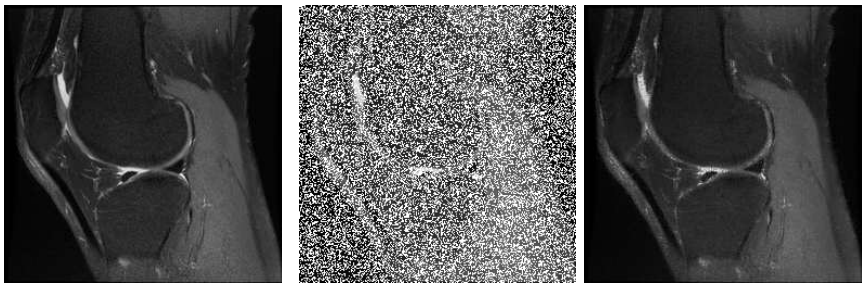
The more entries are known, i.e., for higher values of ρ_s, the more probable it becomes that the solution of the low-n-rank tensor recovery problem and the ground truth tensor coincide. Additionally, the constraints on the tensor are stronger for higher ρ_s, so the search space is smaller and the algorithms converge faster (see first and second experiment in table 1 where the setting only differs in the value of ρ_s).

Next, we compared with the work on tensor completion by Liu et al, [6]. Their algorithm solves the optimization problem

$$\underset{\mathbf{X},\mathbf{Y},\mathbf{M}}{\textbf{minimize}} \quad \frac{1}{2}\sum_{i=1}^{N}\alpha_i||\mathbf{M}-\mathbf{X}||_F^2 + \frac{1}{2}\sum_{i=1}^{N}\beta_i||\mathbf{M}-\mathbf{Y}||_F^2 + \sum_{i=1}^{N}\gamma_i||M_{(i)}||_* \qquad (18)$$

Table 2: Tensor recovery experiments with LRTC [6]

tensor dimensions	ρ	r	# iter	rel. error	time(s)
$\mathcal{T} = \mathbb{R}^{20\times20\times20\times20\times20}$	0.2	2	500	0.379	1263
			2000	0.0198	3535
$\mathcal{T} = \mathbb{R}^{20\times20\times20\times20\times20}$	0.3	2	500	0.489	1261
			2000	0.034	3992
$\mathcal{T} = \mathbb{R}^{50\times50\times50\times50}$	0.4	4	500	0.613	2216
			2000	0.021	6238
$\mathcal{T} = \mathbb{R}^{20\times30\times40}$	0.6	2	500	0.0105	6.9
			2000	0.168	26.5



(a) One slice of the original tensor (b) Input to the algorithm (60% known entries) (c) Recovery result

Figure 1: Simulations, using the KNIX data set [30]

$$\textbf{subject to} \quad \mathbf{Y}_\Omega = \mathbf{T}_\Omega$$

which they derived as substitute to their original problem formulation:

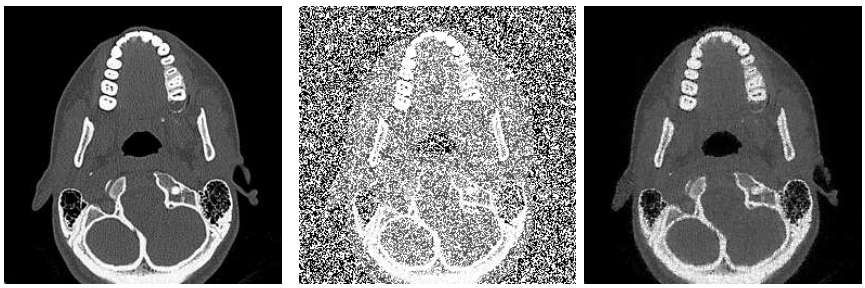
$$\underset{\mathbf{X},\mathbf{Y}}{\textbf{minimize}} \quad \frac{1}{2}||\mathbf{X}-\mathbf{Y}||_F^2 \quad \textbf{s. t.} \quad \sum_{i=1}^{N}||\mathbf{X}_{(i)}||_* \leq c \ \wedge \ \mathbf{Y}_\Omega = \mathbf{T}_\Omega$$

We set the parameters of their algorithm α, β, γ to 1. The algorithm fits a tensor to the given entries, but it fails to recover the ground truth tensor $\mathbf{X}_0$. The maximum iteration number was set once to 500, and once to 2000. A higher number of iterations gives the algorithm time to get closer to the minimizer.

An iterative scheme adjusting the parameters α and β during the runtime should result in a better fit to the ground truth tensor $\mathbf{X}_0$. The rank they reported for the matrices M_i does not directly reflect the rank of the unfoldings $\mathbf{X}_{(i)}$ as equality $M_i = \mathbf{X}_{(i)} = \mathbf{Y}_{(i)}$ is not enforced (this would require α and β to become large).

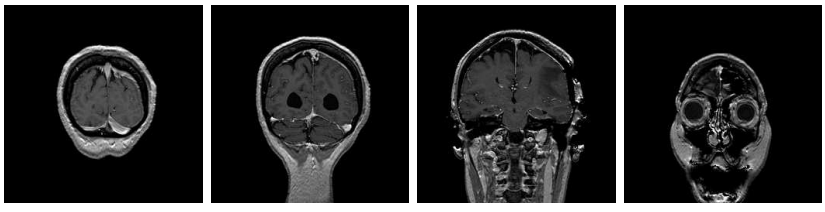
Experiments with medical/hyperspectral data: For the following experiments we used medical data from the OsiriX repository [30], which contains a large number of different data sets of mostly MRI scans; it is available online.

Figure 1 shows a recovery experiment using the KNIX data set of this repository [30]. It contains 22 slices through a human knee, each having dimension 256×256. So the input data is a third-order tensor of dimensions $256 \times 256 \times 22$.

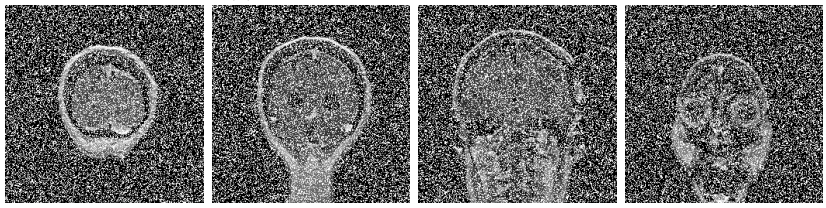


(a) One slice of the original tensor (b) Input to the algorithm (50% known entries) (c) Recovery result

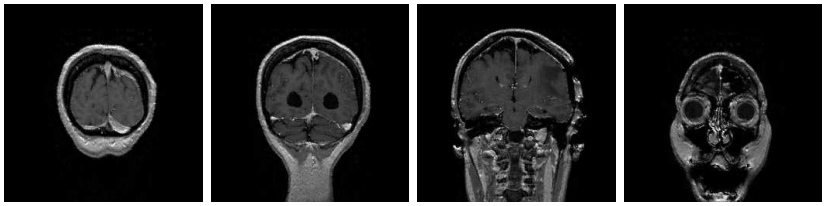
Figure 2: Simulations, using the INCISIX data set [30]



(a) Four slices of the original tensor



(b) Corresponding input to the algorithm (70% known entries)



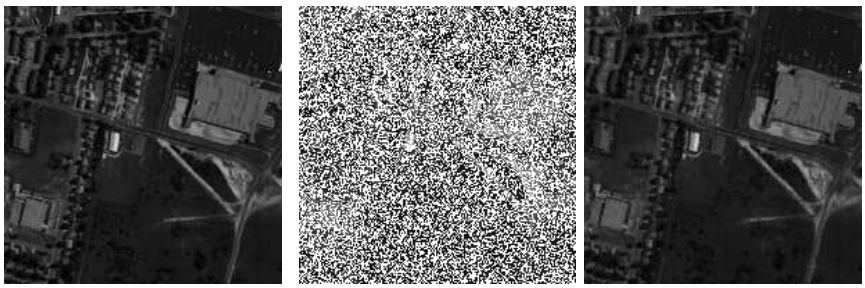
(c) Recovery result

Figure 3: BRAINIX data set [30]

We only show one of the slices exemplarily. In the left of figure 1 the ground truth is shown. The middle image shows the known entries (60% of the entries are known). On the right, we show the recovered image.

Next, we used the INCISIX data set. It contains 166 images of size 256×256. Figure 2 shows the recovery result for one of the images from only 50% of the entries.

We picked one other data set of [30], the BRAINIX data set. It is a three-dimensional scan of the head and brain of a subject. The data forms a third-order tensor of dimensions $256 \times 256 \times 100$. Figure 3 shows the recovery result from 70% of



(a) One slice of the original tensor (b) Input to the algorithm (50% known entries) (c) Recovery result

Figure 4: Hyperspectral data, Urban data set [31]

the entries.

We also applied our algorithms to hyperspectral data. For this experiment we used the URBAN data set from the Army Geospatial Center of the US Army Corps of Engineers [31]. We used all 69 images (each image corresponds to a different band of wavelengths of the light that was collected) and used a spatial resolution of 200×200 pixels. The recovery result is shown for one of the bands in figure 4.

7. Concluding remarks

In this work we focussed on the algorithmic aspects of tensor completion via convex optimization. Our experiments suggest, that tensors that have sufficiently low n-rank and that have their singular spaces in general position can be recovered exactly from a subset of the entries of the tensor via the solution of the low-n-rank tensor pursuit. The derivation of recovery guarantees is still an open problem, but it seems that a result along the lines of the results on matrix completion, [4, 5], is likely to hold. This possibility provides an exciting direction of future theoretical work.

Acknowledgments

S. G. acknowledges funding as a Research Fellow of the Japan Society for the Promotion of Science (JSPS), DC2 grant.

References

[1] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[2] M. Fazel, H. Hindi, and S. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings of the American Control Conference*, volume 6, pages 4734 – 4739, 2001.

[3] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum rank solutions to linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

[4] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Found. of Comput. Math.*, 9:717–772, 2008.

[5] B. Recht. A simpler approach to matrix completion. to appear in the Journal of Machine Learning Research, http://arxiv.org/abs/0910.0651, 2009.

[6] J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. In *IEEE 12th International Conference on Computer Vision*, pages 2114–2121, 2009.
[7] L.-H. Lim and P. Common. Multiarray signal processing: Tensor decomposition meets compressed sensing. *Comptes Rendus Mecanique*, 338:311–320, 2010.
[8] J. Håstad. Tensor rank is NP-complete. *J. Algorithms*, 11:644–654, 1990.
[9] C. Navasca and L. De Lathauwer. Low multilinear tensor approximation via semidefinite programming. In *17th European Signal Processing Conference (EUSIPCO), Glasgow*, Seattle, USA, 2009.
[10] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, 2006.
[11] A. Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20(1-2):89–97, 2004.
[12] J. Douglas and H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Trans. of the American Mathematical Society*, 82:421–439, 1956.
[13] P. L. Combettes and J.-C. Pesquet. A Douglas-Rachford splitting approach to nonsmooth convex variational signal recovery. *IEEE J. Sel. Top. Signal Process.*, 1(4):564–574, 2007.
[14] P. L. Combettes and J.-C. Pesquet. A proximal decomposition method for solving convex variational inverse problems. *Inverse Problems*, 25(6):065014, 2008.
[15] I. Yamada, M. Yukawa, and M. Yamagishi. Minimizing the moreau envelope of nonsmooth convex functions over the fixed point set of certain quasi-nonexpansive mappings. In H. H. Bauschke et al, editor, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, chapter 17. Springer-Verlag, New York, 2011.
[16] S. Gandy and I. Yamada. Convex optimization techniques for the efficient recovery of a sparsely corrupted low-rank matrix. *Journal of Math-for-Industry*, 2(B):147–156, 2010. MI: Global COE Program Education-and-Research Hub for Mathematics-for-Industry.
[17] J.-J. Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertien. *C.R.Acad.Sci. Paris Ser. A Math*, 244:2897–2899, 1962.
[18] P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *SIAM Multiscale Model. Simul.*, 4:1168–1200, 2005.
[19] J. F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM J. on Optimization*, 20(4):1956–1982, 2008.
[20] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins Univ. Press, 1996.
[21] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite-element approximations. *Comp. Math. Appl.*, 2:17–40, 1976.
[22] R. Glowinski and A.Marrocco. Sur l'approximation par éléments finis d'ordre un, et la résolution par pénalisation-dualité d'une classe de problèmes Dirichlets nonlinéaires. *Rev. Française d'Aut. Inf. Rech. Oper., R-2*, pages 41–76, 1975.
[23] J. Eckstein and D. P. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Math. Programming*, 55:293–318, 1992.
[24] D. Gabay. Applications of the method of multipliers to variational inequalities. In M. Fortin and R. Glowinski, editors, *Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems*. North-Holland, 1983.
[25] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation*. Prentice-Hall, Inc., 1989.
[26] J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8:141–148, 1988.
[27] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.
[28] M. W. Berry. Large-scale sparse singular value decompositions. *The International Journal of Supercomputer Applications*, 6:13–49, 1992.
[29] R. M. Larsen. PROPACK-software for large and sparse svd calculations. available from http://sun.stanford.edu/srmunk/PROPACK/.
[30] OsiriX. DICOM sample image sets repository. http://www.osirix-viewer.com, http://pubimage.hcuge.ch:8080/.
[31] Army Geospatial Center of the US Army Corps of Engineers. Sample data sets for the hypercube software. http://www.agc.army.mil/Hypercube/.