Applied Numerical Mathematics 192 (2023) 297–318

Contents lists available at ScienceDirect

Applied Numerical Mathematics

journal homepage: www.elsevier.com/locate/apnum

A variable projection method for large-scale inverse problems with ℓ^1 regularization

Matthias Chung [a,*], Rosemary A. Renaut [b]

[a] Department of Mathematics, Emory University, Atlanta, GA, United States of America
[b] School of Mathematical and Statistical Sciences, Arizona State University, Tempe, AZ, United States of America

ARTICLE INFO

Article history:
Received 29 March 2023
Received in revised form 21 June 2023
Accepted 23 June 2023
Available online 28 June 2023

Keywords:
Variable projection
Alternating Direction Method of Multipliers (ADMM)
Regularization
Inverse problems
χ^2 test

ABSTRACT

Inference by means of mathematical modeling from a collection of observations remains a crucial tool for scientific discovery and is ubiquitous in application areas such as signal compression, imaging restoration, and supervised machine learning. With ever-increasing model complexities and larger data sets, new specially designed methods are urgently needed to recover meaningful quantities of interest. We consider the broad spectrum of linear inverse problems where the aim is to reconstruct quantities with a sparse representation on some vector space. We provide a new variable projection augmented Lagrangian algorithm to solve the underlying ℓ^1 regularized inverse problem that is both efficient and effective. We present the proof of convergence for an algorithm using an inexact step for the projected problem at each iteration. The performance and convergence properties for various imaging problems are investigated. The efficiency of the algorithm makes it feasible to automatically find the regularization parameter, here illustrated using an argument based on the degrees of freedom of the objective function equipped with a bisection algorithm for root-finding.

1. Introduction

Many scientific problems are modeled through the linear relationship

$$\mathbf{b} = \mathbf{A}\mathbf{x}_{\text{true}} + \boldsymbol{\varepsilon}. \tag{1}$$

Here $\mathbf{A} \in \mathbb{R}^{m\times n}$ is the forward process that maps from an unknown true solution, $\mathbf{x}_{\text{true}} \in \mathbb{R}^n$, to true observations, $\mathbf{b}_{\text{true}} \in \mathbb{R}^m$. Practically, the observations are given by $\mathbf{b} = \mathbf{b}_{\text{true}} + \boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon}$ represents the noise contamination of the data. For the models of interest, we assume that $\mathbf{A}$ is obtained via the discretization of an underlying physical model which is ill-posed (i.e., a solution does not exist, is not unique, or does not depend continuously on the data [44]) yielding $\mathbf{A}$ that is numerically ill-conditioned. Given $\mathbf{A}$ and $\mathbf{b}$, as well as some desired characteristics of $\mathbf{x}_{\text{true}}$, the aim in *inverse problems* is to obtain an approximate solution $\widehat{\mathbf{x}}$ that inherits the desired properties of $\mathbf{x}_{\text{true}}$ [48]. Such inverse problems arise in many different fields, such as, but not limited to, medical imaging, geophysics, and signal processing [49,62,73]. Efficient and effective algorithms to solve these large-scale problems are of high relevance.

* Corresponding author.
E-mail addresses: matthias.chung@emory.edu (M. Chung), renaut@asu.edu (R.A. Renaut).

https://doi.org/10.1016/j.apnum.2023.06.015
0168-9274/

For the solution of eq. (1) we seek

$$\widehat{\mathbf{x}} \in \arg\min_{\mathbf{x}} \varphi(\mathbf{x}) = \tfrac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 + \mu\|\mathbf{D}\mathbf{x}\|_1, \tag{2}$$

in which the inversion process is stabilized using the *regularization* $\|\mathbf{D}\mathbf{x}\|_1$ under the assumption of sparsity for $\mathbf{D}\mathbf{x}$. Here $\|\cdot\|_p$ denotes the ℓ^p-norm, $\mathbf{D} \in \mathbb{R}^{\ell\times n}$ is a predefined matrix, and $\mu > 0$ is a regularization parameter that weights the relevant contributions of the ℓ^2-data loss and the ℓ^1-regularizer. Equation (2) gained major attention, in particular, in the early 2000s, due to its connection to compressed sensing and the reconstruction of solutions with sparsity properties in the range of $\mathbf{D}$ [16]. The ℓ^1-regularization in eq. (2) is also referred to as the (generalized, if $\mathbf{D} \neq \mathbf{I}_n$) *least absolute shrinkage and selection operator* (lasso) regression or *basis pursuit denoising* (BPDN), [79,80]. This contrasts with ridge regression, or Tikhonov regularization, in which the ℓ^1-norm in eq. (2) is replaced by the ℓ^2 norm, yielding

$$\min_{\mathbf{x}} \tfrac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 + \tfrac{\mu^2}{2}\|\mathbf{D}\mathbf{x}\|_2^2. \tag{3}$$

Equations (2) and (3) have a Bayesian interpretation, where $\widehat{\mathbf{x}}$ represents the maximum a-posteriori (MAP) estimate of a posterior with a linear model, Gaussian likelihood and a particular Laplace prior, or Gaussian prior, respectively, [15,82]. We further note that eq. (2) is a specialization of the more general ℓ^p-ℓ^q regularization in which the data fit and regularization terms are measured in the p and q-norms, respectively, rather than with $p = 2$ and $q = 1$, as here, [18].

The introduction of methods including the Alternating Direction Method of Multipliers (ADMM), Split Bregman, majorization-minimization, as well as iteratively reweighted norms for general ℓ^p-ℓ^q formulations, and the Fast Iterative Shrinkage-Thresholding Algorithm [13,33,36,53,58,71], have made it computationally feasible to solve large-scale problems described by eq. (2), e.g. [6,29,30,37]. Nevertheless, compared to ℓ^2-regularized linear least-squares (Tikhonov), the need to use these nonlinear optimization methods even with the introduction of limited memory generalized Krylov methods makes the solution of the ℓ^1-regularized problem computationally more expensive [13,79]. Furthermore, for real-world applications, the regularization parameter μ in each case is generally unknown and the need to select an appropriate μ can require multiple solves of the given regularized problem for different choices of μ. While efficient solvers are crucial in either case, here our focus is on the design of an efficient solver for eq. (2) and the validation of the new algorithm, not only for a selection of inverse problems, but also in the context of the efficient estimate of a suitable μ using a χ^2 *degrees of freedom* (DF) argument [60].

Main Contributions: We present a new *variable projection* augmented Lagrangian algorithm (VPAL) for solving eq. (2). First, this approach is novel through its use of *variable projection* in conjunction with inexact solves in an alternating direction algorithm. Indeed, as contrasted with the standard ADMM algorithm, which uses the separability of the underlying objective function to perform an alternating minimization that may be exact or inexact, the presented VPAL algorithm applies *variable projection* to recast the minimization with respect to a single variable. Under the standard condition that the matrix $\mathbf{A}$ has full column rank, we prove that the presented algorithm that uses an inexact solve at each iteration converges to the minimum of eq. (2). Numerical evidence of the low computational complexity of the VPAL algorithm is provided. Second, taking note of the efficiency of VPAL, it is augmented with an automated root-finding algorithm to select μ using a χ^2-test based on a DF argument for eq. (2) in the context of image restoration. The same approach can be used to select μ by imposing the discrepancy principle for the data fit term in eq. (2) [63], or within any other parameter-choice method in which a root-finding algorithm is suitable. Our findings are corroborated by numerical experiments on various imaging restoration and projection problems.

This work is organized as follows. In Section 2 we introduce further notation and provide the background on ℓ^1-regularization methods in Section 2.1, as well as methods to estimate μ in Section 2.2. We present the new algorithm in Section 3, with the algorithm development in Section 3.1 and a discussion of the convergence in Section 3.2. The bisection root-finding algorithm to estimate μ in the context of image restoration is briefly discussed in Section 4. Numerical investigations are provided in Section 5, and we conclude our work with an overview of conclusions and topics for future research in Section 6.

2. Background

We briefly elaborate in Section 2.1 on standard approaches for the solution of eq. (2) using the iterative ADMM algorithm, as outlined in Algorithm 1 [5,14,17,29,31,35,66]. This reveals that the main computational limitation of ADMM is the need to solve at least one problem equivalent to a Tikhonov solve, as in eq. (3), at each ADMM iteration. Regularization parameter selection methods are addressed in Section 2.2.

2.1. Methods to solve generalized lasso problems

Introducing $\mathbf{y} \in \mathbb{R}^{\ell}$ with $\mathbf{y} = \mathbf{D}\mathbf{x}$, eq. (2) is equivalent to

$$\min_{\mathbf{x},\mathbf{y}} f(\mathbf{x},\mathbf{y}) = \tfrac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 + \mu\|\mathbf{y}\|_1 \quad \text{subject to } \mathbf{D}\mathbf{x}-\mathbf{y} = \mathbf{0}. \tag{4}$$

Now, following the well-established *augmented Lagrangian* framework, a quadratic penalty term is added to eq. (4) yielding

$$\mathcal{L}_{\text{aug}}(\mathbf{x}, \mathbf{y}, \mathbf{z}; \lambda) = f(\mathbf{x}, \mathbf{y}) + \mathbf{z}^\top(\mathbf{D}\mathbf{x} - \mathbf{y}) + \tfrac{\lambda^2}{2} \|\mathbf{D}\mathbf{x} - \mathbf{y}\|_2^2, \tag{5}$$

where $\mathbf{z} \in \mathbb{R}^\ell$ denotes the vector of Lagrange multipliers and λ is a scalar penalty parameter [51,72] and [67, Chapter 17]. Merging the last two quadratic terms in eq. (5) gives

$$\mathcal{L}_{\text{aug}}(\mathbf{x}, \mathbf{y}, \mathbf{c}; \lambda) = f(\mathbf{x}, \mathbf{y}) + \tfrac{\lambda^2}{2} \|\mathbf{D}\mathbf{x} - \mathbf{y} + \mathbf{c}\|_2^2 - \tfrac{\lambda^2}{2} \|\mathbf{c}\|_2^2, \tag{6}$$

where $\mathbf{c} = \mathbf{z}/\lambda^2$ is a scaled Lagrangian multiplier. For fixed $\mathbf{c}$, eq. (6) can be solved using an *alternating direction* approach to minimize the augmented Lagrangian $\mathcal{L}_{\text{aug}}(\mathbf{x}, \mathbf{y}, \mathbf{c}; \lambda)$ with respect to $\mathbf{x}$ and $\mathbf{y}$.

From the first-order optimality condition, we expect $\nabla_{\mathbf{x},\mathbf{y}}\, \mathcal{L}_{\text{aug}}(\mathbf{x}_{k+1}, \mathbf{y}_{k+1}, \mathbf{c}_k; \lambda) \approx \mathbf{0}$, whenever $(\mathbf{x}_{k+1}, \mathbf{y}_{k+1})$ approximately minimizes $\mathcal{L}_{\text{aug}}(\cdot, \cdot, \mathbf{c}_k; \lambda)$. But, we see that $\mathbf{D}\mathbf{x}_{k+1} - \mathbf{y}_{k+1} + \mathbf{c}_k$ approximates the corresponding Lagrange multiplier by noting that $\nabla_{\mathbf{x},\mathbf{y}}\, \mathcal{L}_{\text{aug}}(\mathbf{x}, \mathbf{y}, \mathbf{c}_k; \lambda) = \nabla_{\mathbf{x},\mathbf{y}} f(\mathbf{x}, \mathbf{y}) + \lambda^2 [\mathbf{D}; -\mathbf{I}_\ell]^\top (\mathbf{D}\mathbf{x} - \mathbf{y} + \mathbf{c}_k)$. In consequence we obtain the *method of multipliers* [51,72], in which, for initial $\mathbf{c}_0 \in \mathbb{R}^\ell$ and a choice of λ, we iterate to convergence over

$$(\mathbf{x}_{k+1}, \mathbf{y}_{k+1}) = \arg\min_{\mathbf{x},\mathbf{y}} \mathcal{L}_{\text{aug}}(\mathbf{x}, \mathbf{y}, \mathbf{c}_k; \lambda) \tag{7a}$$

$$\mathbf{c}_{k+1} = \mathbf{D}\mathbf{x}_{k+1} - \mathbf{y}_{k+1} + \mathbf{c}_k. \tag{7b}$$

The main computational effort associated with implementing eqs. (7a) and (7b) lies in solving eq. (7a). Hence, splitting the optimization problem eq. (7a) with respect to $\mathbf{x}$ and $\mathbf{y}$, and replacing $\mathbf{c}_k$ by $\mathbf{c}$ for ease of notation, an equivalent objective function to $\mathcal{L}_{\text{aug}}(\mathbf{x}, \mathbf{y}, \mathbf{c}; \lambda)$ is given by $h : \mathbb{R}^n \times \mathbb{R}^\ell \to \mathbb{R}$ with

$$h(\mathbf{x}, \mathbf{y}) = \tfrac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \tfrac{\lambda^2}{2} \|\mathbf{D}\mathbf{x} - \mathbf{y} + \mathbf{c}\|_2^2 + \mu \|\mathbf{y}\|_1. \tag{8}$$

Optimizing with an *alternating direction* approach offers the potential to reduce computational complexity. This leads to the widely used *alternating direction method of multipliers* (ADMM), where $\mathbf{x}_{k+1}$ and $\mathbf{y}_{k+1}$ in eq. (7a) are approximately obtained by performing the alternating direction optimizations

$$\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} h(\mathbf{x}, \mathbf{y}_k), \tag{9a}$$

$$\mathbf{y}_{k+1} = \arg\min_{\mathbf{y}} h(\mathbf{x}_{k+1}, \mathbf{y}). \tag{9b}$$

Considering the specific form of eq. (4), each of these updates yields a problem that offers a computational advantage. Ignoring constant terms, eq. (9a) reduces to a linear least-squares problem of the form

$$\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} \tfrac{1}{2} \left\| \begin{bmatrix} \mathbf{A} \\ \lambda \mathbf{D} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{b} \\ \lambda\,(\mathbf{y}_k - \mathbf{c}_k) \end{bmatrix} \right\|_2^2, \tag{10}$$

for which there are many efficient approaches using either direct or inexact solvers [43]. On the other hand, eq. (9b) reduces to

$$\mathbf{y}_{k+1} = \arg\min_{\mathbf{y}} \mu \|\mathbf{y}\|_1 + \tfrac{\lambda^2}{2} \left\|\mathbf{d}_{k+1} - \mathbf{y}\right\|_2^2, \tag{11}$$

where $\mathbf{d}_{k+1} = \mathbf{D}\mathbf{x}_{k+1} + \mathbf{c}_k$. Equation (11) is a well-known *shrinkage* problem that has the explicit solution

$$\mathbf{y}_{k+1} = \text{sign}(\mathbf{d}_{k+1}) \odot \left(\left|\mathbf{d}_{k+1}\right| - \frac{\mu}{\lambda^2} \mathbf{1}_\ell \right)_+. \tag{12}$$

Here, the element-wise function $(\cdot)_+$ is defined by $(w)_+ = w$, for $w > 0$, and $w = 0$, otherwise, $\odot$ denotes the Hadamard product, and $|\cdot|$ is the element-wise absolute value. Consequently, the update eq. (12) is of low computational complexity. Combining eqs. (10) and (12) with eq. (7b), yields the ADMM algorithm for the solution of eq. (2) as summarized in Algorithm 1, here with the assumption of an exact solve for eq. (10). Stopping criteria for ADMM are outlined in [10].

The computational complexity of a *plain* ADMM algorithm, as described in Algorithm 1, is $\mathcal{O}(\widetilde{K}(m+\ell)n^2)$, where $\widetilde{K}$ refers to the required number of iterations. There is, however, also substantial literature on the use of efficient direct factorizations, inexact solves, algorithms based on generalized Krylov spaces with limited memory, as well as preconditioning to improve the computational efficiency when solving eq. (10) [13,17,19,45,56]. Our focus takes a different direction, in which we recast Algorithm 1 using a *variable projection* approach [40,69]. Note that here we assume a non-adaptive approach for the parameters λ and μ that are held fixed.

Algorithm 1 Alternating Direction Method of Multipliers (ADMM) [31]

input $\mathbf{A}$, $\mathbf{b}$, $\mathbf{D}$, μ, and λ
initialize $\mathbf{c}_0 = \mathbf{x}_0 = \mathbf{y}_0 = \mathbf{0}$, and $k = 0$
while not converged **do**
  $\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} \ \frac{1}{2}\|\mathbf{Ax} - \mathbf{b}\|_2^2 + \frac{\lambda^2}{2}\|\mathbf{Dx} - \mathbf{y}_k + \mathbf{c}_k\|_2^2$
  $\mathbf{y}_{k+1} = \arg\min_{\mathbf{y}} \ \mu\|\mathbf{y}\|_1 + \frac{\lambda^2}{2}\left\|\mathbf{Dx}_{k+1} - \mathbf{y} + \mathbf{c}_k\right\|_2^2$
  $\mathbf{c}_{k+1} = \mathbf{c}_k + \mathbf{Dx}_{k+1} - \mathbf{y}_{k+1}$
  $k = k + 1$
end while
output $\mathbf{x}_k$

2.2. Regularization parameter selection

There is substantial literature on determining the regularization parameter μ in eq. (3), for which details are available in texts such as [4,47,48] or in literature focused on a review of parameter selection approaches [12]. Such methods range from techniques that do not need any information about the statistical distribution of the noise in the data, such as the L-curve that trades off between the data fit and the regularizer [48], the method of generalized cross-validation [39], the residual whiteness principle (RWP) based on the correlation of the components in the residual [59], and supervised learning techniques [2,68]. Many other approaches are statistically-based and require that some underlying noise distribution is assumed, including the well-known Morozov discrepancy principle (DP) [63,82]. The reader is referred to this extensive literature on parameter choice approaches, including for $\ell^p - \ell^q$ formulations, the assumption that eq. (2) defines an image restoration problem, or under certain algorithmic considerations to solve eq. (2) [12]. It has also been suggested to consider the classes of parameter choice methods as either non-stationary or stationary [12]. In the non-stationary case, (equivalently adaptive), μ is automatically updated throughout a given algorithm, such as the iterative majorization-minimization algorithm. In the stationary case, an *optimal* μ is obtained based on a limited number of full solves for a judiciously chosen set of μ values. Here in section 4 we will present a stationary approach but argue that the presented bisection algorithm is efficient and effective, requiring very few full solutions of eq. (2).

In the statistical literature it has been shown that the family of solutions for eq. (2) is piecewise linear, namely the regularization path for solutions as μ varies has a piecewise linear property in μ [28]. Thus, path-following on μ has been used to analyze the properties of the solution with μ and to design algorithms that determine the bounds $\mu_{\min} < \mu < \mu_{\max}$ [3,25,55,80,85]. We note also, that in the context of Algorithm 1 for data with Poisson noise, it has been suggested that one may use a DP [63] for the data fit residual in eq. (10) to select the Lagrange parameter using a Newton algorithm, assuming a fixed μ [78]. An approach for the case of Gaussian noise is also discussed in [66].

Generally, the lack of an analytic expression for the solution of eq. (2) and the associated computational demands for large-scale problems, presents challenges in both defining a method for optimally selecting μ and with finding an optimal estimate efficiently. In all cases it is therefore imperative to consider whether any new algorithm for the solution of eq. (2) can be used effectively and robustly in a parameter choice method. Here we will address this in the context of a stationary bisection algorithm to be described in section 4, based on the following observations in the context of image restoration.

For the solution of the generalized lasso problem when $\mathbf{A}$ is a blurring operator of an image and $\mathbf{D} \neq \mathbf{I}_n$, a MAP estimator for μ can be derived under the assumption of Gaussian noise in $\mathbf{b}$ [42]. For naturally occurring images, if the TV functional defined by $\mathbf{y} = \mathbf{Dx}$ is Laplace distributed[1] with mean $\boldsymbol{\theta} \in \mathbb{R}^\ell$ and variance $2\beta^2\mathbf{I}_\ell$, i.e., $\mathbf{y} \sim \mathcal{L}(\boldsymbol{\theta}, 2\beta^2\mathbf{I}_\ell)$, then the underlying image $\mathbf{x}$ is said to be *differentially Laplacian*. In this case, under the assumption $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_m)$, the MAP estimator for $\widehat{\mathbf{x}}(\mu)$, here denoted by $\widehat{\mathbf{x}}(\mu_{\text{map}})$, is given by eq. (2) when $\mu = \mu_{\text{map}} = \sigma^2/\beta$ [60, eq. (9)]. For naturally occurring images the use of μ_{map} seems appropriate, but to obtain μ_{map} we require β. In the ideal case we can use the estimate $\beta = \text{std}(\mathbf{Db})/\sqrt{2}$ [61, Algorithm 1], where std denotes the standard deviation. For images that are significantly blurred, however, it is unlikely that β obtained from $\mathbf{b}$ is a good estimate for the true β associated with the unknown image $\mathbf{x}$. Alternatively, we may also use a χ^2-test on eq. (2)

$$\|\mathbf{A}\widehat{\mathbf{x}}(\mu) - \mathbf{b}\|_2^2 + \mu\|\mathbf{D}\widehat{\mathbf{x}}(\mu)\|_1 \cong m\sigma^2, \tag{13}$$

when both $\mathbf{A}$ and $\mathbf{D}$ have full column rank, [60, Theorem 4] and we define μ_{χ^2} to be the μ that satisfies eq. (13).

Finding μ_{χ^2} to satisfy eq. (13) is a natural extension of existing techniques that use the DP applied to the residual, in which, for example, μ in eq. (3) is found by requiring $\|\mathbf{Ax}(\mu) - \mathbf{b}\|_2^2 \approx m\sigma^2$, assuming there are m degrees of freedom in the residual. Arguments based on the degrees of freedom have paved the way to identifying an optimal μ for the lasso problem, $\mathbf{D} = \mathbf{I}_n$ [24,27,28,80,81,85]. Given an efficient algorithm to solve the lasso problem, it becomes computationally feasible to find an optimal μ that satisfies the DP. We focus on verifying that our efficient VPAL solver can be suitably integrated to efficiently find μ_{χ^2} for the generalized lasso problem using a root-finding algorithm.

[1] A random variable y follows a Laplace distribution with mean θ and variance $2\beta^2$, denoted $y \sim \mathcal{L}(\theta, 2\beta^2)$, if its probability density function is $y = 1/2\beta \exp((|y - \theta|)/\beta)$.

Finally, while the results presented in [60] demonstrate that solutions obtained using μ_{χ^2} are improved as compared to those obtained using μ_{map}, [60, Algorithm 2], as we have also confirmed from results not presented here, it must be noted that we cannot immediately apply eq. (13) to find a suitable μ for matrices $\mathbf{A}$ that arise in other models, such as projection, i.e., $\mathbf{A}$ is not a smoothing operator. Still, for the restoration of blurred and noisy images, eq. (13) can be used in the context of a root-finding algorithm to provide an estimate for a suitable μ_{χ^2}. Here we give only the necessary details, but note that the approach provided is feasible when $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_m)$ and σ^2 is available. For colored noise, a suitable whitening transform can be applied to modify eq. (2), effectively, using a weighted norm in the data fit term, see e.g., [61]. We reiterate that the approach presented here is directly applicable only for the restoration of naturally occurring images, [42], and as such can be regarded as an alternative to the RWP [57].

3. Variable Projected Augmented Lagrangian

As discussed in Section 2.1 the computational efficiency of Algorithm 1 is dominated by the effort of repeatedly solving the least-squares problem eq. (10) at step 4 of Algorithm 1, regardless of whether exact or inexact solves are involved. The focus, therefore, of our approach is to significantly reduce the computational cost that arises in obtaining the updates in eq. (7a) by rewriting eqs. (9a) and (9b) in terms of a single term obtained using *variable projection*. Within this reformulation, we suggest a specific inexact solve for $\mathbf{x}_{k+1}$, using a single conjugate gradient (CG) step. As compared to many other efficient algorithms in the literature, the presented algorithm proceeds without the need to maintain any Krylov, or generalized Krylov, subspace, and is, moreover, not specific to a particular choice of kernel giving rise to the matrix $\mathbf{A}$ in eq. (1) [13,18]. We develop the algorithm in Section 3.1 and present the analysis of convergence in Section 3.2 where the convergence is not tied to the particular CG update strategy.

3.1. Algorithm development

We briefly review the under-utilized *variable projection* technique, which is a hybrid of two approaches, (1) alternating direction optimization and (2) block coordinate descent [40,69]. First, we assume a general function $h(\mathbf{x}, \mathbf{y})$ that can be decomposed into two sets of independent variables $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^\ell$. Further, we assume, for simplicity, that h is strictly convex, has compact lower-level sets, and is sufficiently differentiable to ensure convergence to a unique solution $(\widehat{\mathbf{x}}, \widehat{\mathbf{y}})$ [76]. Then, we may use separability at each iteration of the numerical optimization scheme to first optimize over $\mathbf{x}$ while keeping $\mathbf{y}$ constant, and then optimize with respect to $\mathbf{y}$, while keeping $\mathbf{x}$ constant, as utilized in ADMM eq. (9a) and eq. (9b). Another approach applies a *block coordinate descent* for each variable, the iterates, initialized with arbitrary $\mathbf{x}_0, \mathbf{y}_0$, proceed until convergence via the alternating steps

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{s}_{\mathbf{x}}(h, \mathbf{x}_k, \mathbf{y}_k), \tag{14a}$$

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \beta_k \mathbf{s}_{\mathbf{y}}(h, \mathbf{x}_{k+1}, \mathbf{y}_k). \tag{14b}$$

Here, $\mathbf{s}_{\mathbf{x}}$ and $\mathbf{s}_{\mathbf{y}}$ refer to appropriate descent directions with corresponding step sizes α_k and β_k. Within each iteration k, the function h is kept constant with respect to one of the variables while the other performs, for instance, a gradient descent or Newton step with appropriate step size control [83]. Now, forming the hybrid we note that rather than using eq. (14a) directly, we can instead replace eq. (14b) by the single iterative step

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \alpha_k \mathbf{s}_{\mathbf{y}}(h, \underset{\mathbf{x}\in\mathbb{R}^n}{\arg\min}\ h(\mathbf{x}, \mathbf{y}_k), \mathbf{y}_k). \tag{15}$$

With this reformulation, we have now applied the key idea behind *variable projection*, namely the elimination of one set of variables $\mathbf{x}$ by projecting the optimization problem onto a reduced subspace associated with the other set of variables $\mathbf{y}$ [38,74].

Variable projection approaches were specifically developed for *separable nonlinear least-squares* problems, such as $\min_{(\mathbf{x},\mathbf{y})\in\mathbb{R}^{n+\ell}} \|\mathbf{A}(\mathbf{y})\mathbf{x} - \mathbf{b}\|_2^2$ where for fixed $\mathbf{y}$ the optimization problem exhibits a *linear* least-squares problem in $\mathbf{x}$ which may easily be solved. This approach has been successfully applied beyond standard settings, for example, in super-resolution applications, for separable deep neural networks, and within stochastic approximation frameworks, see [21,64,65] for details. Variable projection methods show their full potential when one of the variables can be efficiently eliminated.

For Eq. (7a) calculating $\mathbf{y}$ exactly while keeping $\mathbf{x}$ constant is computationally tractable due to the use of the shrinkage step eq. (12). On the other hand, gains in computational efficiency can be achieved by exploiting an *inexact* solve for $\mathbf{x}$. This new perspective of utilizing *variable projection* resonates with the use of inexact solves of the linear system within ADMM [37]. At the same time, our approach is not the same as using an inexact solve within ADMM. Specifically, the proposed inexact update directly incorporates the shrinkage step in the minimization of eq. (15) before updating the Lagrange multiplier $\mathbf{c}$. While there are various efficient update strategies that may be utilized to find $\mathbf{x}$ as needed in the formulation of eq. (15), such as LBFGS or Krylov subspace type methods [67,70], we focus here on updating $\mathbf{x}$ by performing a single CG step. Specifically, we use the update

$$\mathbf{x}^{(j+1)} = \mathbf{x}^{(j)} - \alpha_j \mathbf{g}_j, \tag{16}$$

where the descent direction $\mathbf{g}_j$ is a vector of length n given by

$$\mathbf{g}_j = \begin{bmatrix} \mathbf{A}^\top & \lambda \mathbf{D}^\top \end{bmatrix} \mathbf{r}_j \qquad \text{with} \qquad \mathbf{r}_j = \begin{bmatrix} \mathbf{A}\mathbf{x}^{(j)} - \mathbf{b} \\ \lambda \mathbf{D}\mathbf{x}^{(j)} - \lambda \left(\mathbf{y}^{(j)} - \mathbf{c}_k \right) \end{bmatrix},$$

and the optimal step length α_j may be computed by

$$\alpha_j = \arg\min_{\alpha} h_{\text{proj}}(\mathbf{x}^{(j)} - \alpha \mathbf{g}_j). \tag{17}$$

The projected function $h_{\text{proj}} : \mathbb{R}^n \to \mathbb{R}$ is defined via the continuous shrinkage mapping $\mathbf{Z} : \mathbb{R}^n \to \mathbb{R}^\ell$ for eq. (12)

$$\mathbf{Z}(\mathbf{x}) = \text{sign}(\mathbf{D}\mathbf{x} + \mathbf{c}) \odot \left(|\mathbf{D}\mathbf{x} + \mathbf{c}| - \frac{\mu}{\lambda^2} \mathbf{1}_\ell \right)_+, \tag{18}$$

giving

$$h_{\text{proj}}(\mathbf{x}) = \tfrac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \tfrac{\lambda^2}{2} \|\mathbf{D}\mathbf{x} - \mathbf{Z}(\mathbf{x}) + \mathbf{c}\|_2^2 + \mu \|\mathbf{Z}(\mathbf{x})\|_1, \tag{19}$$

[52,70]. Note that the optimal calculation of α_j requires only $\mathcal{O}(n)$ operations once $\mathbf{g}_j$, $\mathbf{A}\mathbf{g}_j$, and $\mathbf{D}\mathbf{g}_j$ have been obtained. Adopting this update strategy yields the new *variable projection augmented Lagrangian* (VPAL) method for solving eq. (2) as summarized in Algorithm 2, given for fixed parameters λ and μ.

Algorithm 2 Variable Projected Augmented Lagrangian (VPAL)

input $\mathbf{A}$, $\mathbf{b}$, $\mathbf{D}$, μ, λ
initialize $\mathbf{c}_0 = \mathbf{x}_0 = \mathbf{y}_0 = \mathbf{0}$, and set $k = 0$
while not converged **do**
  set $j = 0$, $\mathbf{x}^{(0)} = \mathbf{x}_k$, $\mathbf{y}^{(0)} = \mathbf{y}_k$
  while not converged **do**
    calculate residual $\mathbf{r}_j = \begin{bmatrix} \mathbf{A}\mathbf{x}^{(j)} - \mathbf{b} \\ \lambda \mathbf{D}\mathbf{x}^{(j)} - \lambda \left(\mathbf{y}^{(j)} - \mathbf{c}_k \right) \end{bmatrix}$,
    calculate direction $\mathbf{g}_j = \begin{bmatrix} \mathbf{A}^\top & \lambda \mathbf{D}^\top \end{bmatrix} \mathbf{r}_j$
    set $\alpha_j = \arg\min_\alpha h_{\text{proj}}(\mathbf{x}^{(j)} - \alpha \mathbf{g}_j)$
    update $\mathbf{x}^{(j+1)} = \mathbf{x}^{(j)} - \alpha_j \mathbf{g}_j$
    $\mathbf{y}^{(j+1)} = \arg\min_{\mathbf{y}} \ \mu \|\mathbf{y}\|_1 + \frac{\lambda^2}{2} \|\mathbf{D}\mathbf{x}^{(j+1)} - \mathbf{y} + \mathbf{c}_k\|_2^2$
    $j = j + 1$
  end while
  set $\mathbf{x}_{k+1} = \mathbf{x}^{(j)}$ and $\mathbf{y}_{k+1} = \mathbf{y}^{(j)}$
  set $\mathbf{c}_{k+1} = \mathbf{c}_k + \mathbf{D}\mathbf{x}_{k+1} - \mathbf{y}_{k+1}$
  $k = k + 1$
end while
output $\mathbf{x}_k$

Remark 1. Within VPAL our ansatz for applying a variable projection technique is "*inverted*" as compared to the standard approaches. Specifically, a typical variable projection method would seek to optimize over the variable that determines the linear least squares problem, here it would be $\mathbf{x}$, and would update the variable occurring nonlinearly, here $\mathbf{y}$, using a standard gradient or Newton update, [40,69]. Here we reverse the roles of the variables.

Remark 2. The underlying formulation for the *variable projection* approach minimizes the single nonlinear function $h_{\text{proj}}(\mathbf{x})$, using shrinkage eq. (12) at each CG step. In contrast, in ADMM there are two distinct minimizations at each iteration, as given in eqs. (9a) and (9b). We see that the step to obtain $\mathbf{x}_{k+1}$ in eq. (9a) minimizes the nonlinear function $h(\mathbf{x}, \mathbf{y})$ given in eq. (8) with $\mathbf{y}$ held fixed. In particular, while alternating direction methods such as ADMM solve eq. (2) by using a linear update to find $\mathbf{x}$, the variable projection approach uses a nonlinear optimization step that directly incorporates changes in $\mathbf{y}$ when finding $\mathbf{x}$. These observations apply regardless of whether the updates employed are exact or inexact and it is easy to verify with simple examples that each algorithm minimizes the different functions.

Remark 3. The idea of using an explicit CG with inexact solves within an alternating optimization is not new. Indeed it is quite standard to use an inexact update step for eq. (10) within an ADMM algorithm. For instance, using just a few CG steps or other iterative methods has been proposed and analyzed [26,41,45,77]. We follow this approach within the context of a variable projection framework.

Remark 4. An optimal step size for the projected function h_{proj} as computed in eq. (17) is not required. Rather it may be sufficient to use an efficient step size selection that provides a decrease along the search direction. Since $\mathbf{y}^{(j)} \approx \mathbf{Z}(\mathbf{x}^{(j)} - \alpha_j \mathbf{g}_j)$, especially in later iterations, we may replace this term in h_{proj}, resulting in a linear least squares problem where the (linearized) optimal step size can be computed efficiently by

$$\alpha_j = \frac{\mathbf{g}_j^\top \mathbf{g}_j}{\mathbf{t}_j^\top \mathbf{t}_j}, \qquad \text{where} \qquad \mathbf{t}_j = \begin{bmatrix} \mathbf{A} \\ \lambda \mathbf{D} \end{bmatrix} \mathbf{g}_j.$$

Empirically, we observe that the linearized step size selection is very close to the optimal step size $\widehat{\alpha}$, approximately underestimated by about 10%, and is computationally less expensive (experiment not shown).

Remark 5. In eq. (16) the dominant costs to obtain the update $\mathbf{x}^{(j+1)}$ are the matrix-vector products needed to generate $\mathbf{g}_j$ and $\mathbf{t}_j$. There are two matrix-vector multiplications with $\mathbf{A}$, two with $\mathbf{D}$, and one each with their respective transposes. This corresponds to two multiplications with matrices of sizes $(m+\ell) \times n$ and $n \times (m+\ell)$ yielding a computational complexity of $\mathcal{O}((m+\ell)n)$. Hence, the computational complexity of Algorithm 2 is $\mathcal{O}(K J (m+\ell)n)$, where J refers to an average of the number of inner iterations for the inner while loop (steps 5 to 12 in Algorithm 2) and K is the number of outer iterations. We remark also that the algorithm is a limited memory implementation, at any step we only need to calculate the given matrix-vector products and store a limited number of vectors for the next step. No information about the underlying space for the solution is used, whether in terms of maintaining the basis for the space, or in terms of performing any additional factorizations on the space for efficient updating.

In the practical implementation of Algorithm 2, we eliminate the while loop over $\mathbf{x}^{(j)}$ and $\mathbf{y}^{(j)}$ (steps 5 to 12) and only perform a single update of $\mathbf{x}_k$ and $\mathbf{y}_k$, a common approach, e.g., see [37, page 331]. This reduces the need to provide suitable stopping conditions for carrying out the inexact solve and reduces the computational complexity of a practical implementation of VPAL to $\mathcal{O}(K(m+\ell)n)$. We observe fast convergence as highlighted in Section 5. To terminate the outer iteration k we use stopping criteria that are adapted from standard criteria for terminating iterations in unconstrained optimization, i.e., given a user-specified tolerance $\tau > 0$, we terminate the algorithm when both $h(\mathbf{x}_k, \mathbf{y}_k) - h(\mathbf{x}_{k+1}, \mathbf{y}_{k+1}) \leq \tau(1 + h(\mathbf{x}_{k+1}, \mathbf{y}_{k+1}))$ and $\|\mathbf{x}_k - \mathbf{x}_{k+1}\|_\infty \leq \sqrt{\tau}(1 + \|\mathbf{x}_{k+1}\|_\infty)$ [34]. A simplified Matlab source code of the VPAL method illustrated on a simple denoising example is provided in Appendix A. We also provide a more elaborated Matlab code which is available and maintained at `www.github.com/matthiaschung/vpal`.

3.2. Convergence

There is extensive literature on the convergence of the ADMM algorithm, including results that provide convergence rates, address the use of inexact solves, and consider modifications of the given ADMM algorithm to include additional terms, or allow for separability with respect to more than two blocks of variables, [1,46]. Generally, the convergence rate results rely on suitable error bounds for the solutions, such as absolute summability of the errors arising through the iteration on eq. (10) [19,26,50]. Comprehensive overviews of the convergence results are available for example in [10,14,56,84]. For our analysis we do not seek to provide convergence rates, consequently, our approach is to present a general result without applying any specific error bounds on the inexact solves used to minimize eq. (19). This result relies on the well-established convergence analysis of augmented Lagrangian methods for the sequence of solves for $\mathbf{x}_k$ [8–10,52,72]. Thus, the focus of the proof is to show that the solves obtained via variable projection corresponding to inexact solves in the inner loop, steps 5 to 12 of Algorithm 2, are sufficient to solve eq. (7a). Our convergence result for Algorithm 2 is summarized in Theorem 3.4. To be complete we note first, without proof, the convexity of $h(\mathbf{x}, \mathbf{y})$.

Lemma 3.1. *Suppose that* $\mathbf{A}$ *has full column rank. Then, for arbitrary but fixed* $\mathbf{c}$ *and* $\mu, \lambda > 0$, $h(\mathbf{x}, \mathbf{y})$ *is strictly convex and has a unique minimizer* $(\widehat{\mathbf{x}}, \widehat{\mathbf{y}})$.

This result is well-known and the proof is omitted. Now, using $\mathbf{Z}$ and h_{proj} defined in eqs. (18) and (19) we obtain the following results.

Lemma 3.2. *Let* $\mu, \lambda > 0$, *then the following statements are true.*

1. *For any fixed* $\mathbf{x}$ *there exists a* $\mathbf{y}$ *such that* $h_{\text{proj}}(\mathbf{x}) = h(\mathbf{x}, \mathbf{y})$.
2. *For any* $\mathbf{y}$ *and arbitrary fixed* $\mathbf{x}$ *the inequality* $h_{\text{proj}}(\mathbf{x}) \leq h(\mathbf{x}, \mathbf{y})$ *holds.*
3. *Let* $(\widehat{\mathbf{x}}, \widehat{\mathbf{y}})$ *be the (unique) global minimizer of* $h(\mathbf{x}, \mathbf{y})$. *Then* $\widehat{\mathbf{x}}$ *is the (unique) global minimizer of* $h_{\text{proj}}(\mathbf{x})$.

Proof. 1. The first statement follows directly by the definitions, eqs. (8), (18) and (19), i.e., if $\mathbf{y} = \mathbf{Z}(\mathbf{x})$, then $h_{\text{proj}}(\mathbf{x}) = h(\mathbf{x}, \mathbf{y})$.
2. Due to optimality of $\mathbf{Z}(\mathbf{x}) = \arg\min_{\mathbf{y}} h(\mathbf{x}, \mathbf{y})$ for arbitrary fixed $\mathbf{x}$, $h(\mathbf{x}, \mathbf{Z}(\mathbf{x})) \leq h(\mathbf{x}, \mathbf{y})$ follows.

3. By eqs. (8) and (19) we have $h_{\text{proj}}(\mathbf{x}) = h(\mathbf{x}, \mathbf{Z}(\mathbf{x})) \geq h(\widehat{\mathbf{x}}, \widehat{\mathbf{y}})$ for any $\mathbf{x} \in \mathbb{R}^n$. With Lemma 3.2 item 2, we have $h_{\text{proj}}(\widehat{\mathbf{x}}) \leq h(\widehat{\mathbf{x}}, \widehat{\mathbf{y}})$, thus a global minimizer for $h_{\text{proj}}(\mathbf{x})$ is obtained at $\widehat{\mathbf{x}}$, i.e., $h_{\text{proj}}(\widehat{\mathbf{x}}) = h(\widehat{\mathbf{x}}, \mathbf{Z}(\widehat{\mathbf{x}}))$. Further, we must have $\mathbf{Z}(\widehat{\mathbf{x}}) = \widehat{\mathbf{y}}$, since $\mathbf{Z}(\widehat{\mathbf{x}})$ minimizes $h(\widehat{\mathbf{x}}, \cdot)$ with respect to the second component and noticing that $\mathbf{Z}(\widehat{\mathbf{x}})$ is its unique minimizer.
Next, assume $(\widehat{\mathbf{x}}, \widehat{\mathbf{y}})$ is the *unique* global minimizer of $h(\mathbf{x}, \mathbf{y})$ and assume further that there exists an $\overline{\mathbf{x}} \neq \widehat{\mathbf{x}}$ that is also a global minimizer of $h_{\text{proj}}(\mathbf{x})$, i.e., $h_{\text{proj}}(\overline{\mathbf{x}}) = h_{\text{proj}}(\widehat{\mathbf{x}})$. By this assumption we have that $h(\widehat{\mathbf{x}}, \widehat{\mathbf{y}}) = h(\widehat{\mathbf{x}}, \mathbf{Z}(\widehat{\mathbf{x}})) = h_{\text{proj}}(\widehat{\mathbf{x}}) = h_{\text{proj}}(\overline{\mathbf{x}}) = h(\overline{\mathbf{x}}, \mathbf{Z}(\overline{\mathbf{x}}))$. Hence, $(\overline{\mathbf{x}}, \mathbf{Z}(\overline{\mathbf{x}})) \neq (\widehat{\mathbf{x}}, \widehat{\mathbf{y}})$ is a global minimizer of $h(\mathbf{x}, \mathbf{y})$. But this contradicts the assumption that $(\widehat{\mathbf{x}}, \widehat{\mathbf{y}})$ is the unique global minimizer of $h(\mathbf{x}, \mathbf{y})$, and thus $\widehat{\mathbf{x}}$ is the unique global minimizer of $h_{\text{proj}}(\mathbf{x})$. □

It remains to be shown that the global minimizer of h_{proj}, given by $\widehat{\mathbf{x}}$, is the only minimizer.

Lemma 3.3. *Assume* $\mathbf{A}$ *has full column rank,* $\mathbf{c}$ *is fixed, and* $\mu, \lambda > 0$*; then all minimizers of* h_{proj} *are global.*

Proof. Note, $h_{\text{proj}}(\mathbf{x})$ has a local minimum at $\overline{\mathbf{x}}$ if there exists a neighborhood $\mathcal{D}(\overline{\mathbf{x}})$ of $\overline{\mathbf{x}}$ such that $h_{\text{proj}}(\mathbf{x}) \geq h_{\text{proj}}(\overline{\mathbf{x}})$ for all $\mathbf{x}$ in $\mathcal{D}(\overline{\mathbf{x}})$. Let $(\widehat{\mathbf{x}}, \widehat{\mathbf{y}}) = (\widehat{\mathbf{x}}, \mathbf{Z}(\widehat{\mathbf{x}}))$ denote the unique global minimum of $h(\mathbf{x}, \mathbf{y})$ according to Lemma 3.1. Since $h(\mathbf{x}, \mathbf{y})$ is strictly convex, $h(\mathbf{x}, \mathbf{y})$ is inevitably strictly convex in each of its components $\mathbf{x}$ and $\mathbf{y}$. Let $\overline{\mathbf{y}}$ be arbitrary but fixed, e.g., $\overline{\mathbf{y}} = \mathbf{Z}(\overline{\mathbf{x}})$; then for any $\overline{\mathbf{x}} \in \mathbb{R}^n$ with $\overline{\mathbf{x}} \neq \widehat{\mathbf{x}}$ and $\epsilon > 0$, there exists an $\mathbf{x}^* \neq \overline{\mathbf{x}}$ with $\left\|\mathbf{x}^* - \overline{\mathbf{x}}\right\|_2 < \epsilon$ such that

$$h(\mathbf{x}^*, \overline{\mathbf{y}}) < h(\overline{\mathbf{x}}, \overline{\mathbf{y}}) = h_{\text{proj}}(\overline{\mathbf{x}})$$

due to the strict convexity in $\mathbf{x}$. Further, by Lemma 3.2 item 1 we have $h_{\text{proj}}(\mathbf{x}^*) = h(\mathbf{x}^*, \mathbf{Z}(\mathbf{x}^*))$, and, by Lemma 3.2 item 2, $h(\mathbf{x}^*, \mathbf{Z}(\mathbf{x}^*)) \leq h(\mathbf{x}^*, \overline{\mathbf{y}})$. Together, we have

$$h_{\text{proj}}(\mathbf{x}^*) = h(\mathbf{x}^*, \mathbf{Z}(\mathbf{x}^*)) \leq h(\mathbf{x}^*, \overline{\mathbf{y}}) < h(\overline{\mathbf{x}}, \overline{\mathbf{y}}) = h_{\text{proj}}(\overline{\mathbf{x}}).$$

Hence, there does *not* exist a neighborhood $\mathcal{D}(\overline{\mathbf{x}})$ around $\overline{\mathbf{x}}$ for which $h_{\text{proj}}(\mathbf{x}) \geq h_{\text{proj}}(\overline{\mathbf{x}})$ for all $\mathbf{x} \in \mathcal{D}(\overline{\mathbf{x}})$, and $\overline{\mathbf{x}} \neq \widehat{\mathbf{x}}$ cannot be a local minimizer. For $\overline{\mathbf{x}} = \widehat{\mathbf{x}}$ we have $h_{\text{proj}}(\widehat{\mathbf{x}}) = h(\widehat{\mathbf{x}}, \mathbf{Z}(\widehat{\mathbf{x}}))$ and there is nothing to show. □

Taking the results of Lemmas 3.2 and 3.3 together we arrive at the main convergence result, in which we now consider the minimization at step k of Algorithm 2.

Theorem 3.4. *Given optimization problem eq. (2), where* $\mathbf{A}$ *has full column rank,* $\mu > 0$*, and* $\lambda > 0$ *sufficiently large, Algorithm 2 converges to the unique minimizer* $\widehat{\mathbf{x}}$ *of eq. (2).*

Proof. As noted above, the convergence of the outer loop (steps 3 to 16 of Algorithm 2) is well established for augmented Lagrangian methods, see [8,9] for details. It remains to be shown that the variable projection corresponding to the inner loop, steps 5 to 12 of Algorithm 2, solves eq. (7a). Lemmas 3.1 (item 1 to item 3) and 3.3 ensure uniqueness of the minimizer $\widehat{\mathbf{x}}$ of $h_{\text{proj}}(\mathbf{x})$ and correspondingly to $h(\mathbf{x}, \mathbf{y})$. Due to optimality for $\mathbf{y} = \mathbf{Z}(\mathbf{x})$ and noting that the subgradient of $h(\mathbf{x}, \mathbf{y})$ with respect to $\mathbf{y}$ is given by $\mathfrak{h}_{\mathbf{y}} = \lambda^2 (-\mathbf{D}\mathbf{x} + \mathbf{y} - \mathbf{c}) + \mu (\text{sign}(\mathbf{y}) \odot \mathbf{1}_\ell)$, we have

$$\lambda^2 (-\mathbf{D}\mathbf{x} + \mathbf{Z}(\mathbf{x}) - \mathbf{c}) + \mu (\text{sign}(\mathbf{Z}(\mathbf{x})) \odot \mathbf{1}_\ell) = \mathbf{0}.$$

Therefore,

$$\mathbf{s} = -\left(\mathbf{A}^\top (\mathbf{A}\mathbf{x} - \mathbf{b}) + \lambda^2 \mathbf{D}^\top (\mathbf{D}\mathbf{x} + \mathbf{c} - \mathbf{Z}(\mathbf{x}))\right) \tag{20}$$

is a descent direction of $h_{\text{proj}}(\mathbf{x})$ corresponding to $-\mathbf{g}_j$ in Algorithm 2. Paired with an optimal step size α_j as defined in eq. (17), an efficient descent is ensured until the subgradient optimality conditions are fulfilled. □

Remark 6. Notice that in eq. (20) we use the search direction $\mathbf{s} = -\mathbf{g}_j$, however, other choices are available. In fact, the proof remains valid for any direction $\mathbf{s} = -\mathbf{B}_j \mathbf{g}_j$ with $\mathbf{B}_j$ being symmetric positive definite and therefore includes (inexact) Newton and quasi Newton updates. Further, if we instead replace steps 5 to 12 in Algorithm 2 with an exact minimization of h_{proj}, then the convergence of the VPAL algorithm will be equivalent to that for Algorithm 1 with exact solves. The inexact solves within the VPAL method provides new flexibility and adds efficiency as is demonstrated in Section 5.

4. A bisection algorithm to find μ

Although the importance of eq. (13) was presented in [60], no efficient algorithm to find μ_{χ^2} was given. We briefly describe a bisection root-finding approach that is used to estimate μ_{χ^2} and relies on finding an interval in which $F(\mu) - p\sigma^2$ changes sign, where

$$F(\mu) = \|\mathbf{A}\widehat{\mathbf{x}}(\mu) - \mathbf{b}\|_2^2 + \mu \|\mathbf{D}\widehat{\mathbf{x}}(\mu)\|_1 . \tag{21}$$

While it is clear that $F(\mu) > 0$, we also know, under the assumption that $\|\mathbf{D}\widehat{\mathbf{x}}(\mu)\|_1$ is bounded, that, at least for the case of image restoration, $F(\mu)$ decreases as $\mu \to 0$, [60, Proposition 4]. Consequently, for any choice of $p > 0$, $F(\mu) - p\sigma^2$ decreases, and may become negative if $p\sigma^2$ is sufficiently large. When the estimates for p and σ^2 are accurate we can expect, for sufficiently large μ, that $F(\mu) - p\sigma^2 > 0$. Noting now that we can also calculate $F(\mu)$ as given by eq. (21) given an algorithm to find $\widehat{\mathbf{x}}(\mu)$, and an interval in which $F(\mu) - p\sigma^2$ changes sign, then we can find μ_{χ^2} as the root of $F(\mu) - p\sigma^2 = 0$. This is the same approach as can be applied when applying the DP to the residual.

Here we adapt a standard bisection algorithm to find the root μ_{χ^2}, assuming initial estimates for $\mu_{\min}$ and $\mu_{\max}$ such that $F(\mu_{\min}) - p\sigma^2 < 0 < F(\mu_{\max}) - p\sigma^2$. First, we note that we can use the estimate $\mu_{\max} = 2\|\mathbf{A}^\top \mathbf{b}\|_\infty$. Moreover, an estimate of μ_{map} can be helpful. Even though μ_{map} may be generally inadequate as the actual estimator for a good choice of μ, it can be used to suggest an interval in which a suitable μ_{χ^2} exists. Using μ_{map} we define an interval for μ_{χ^2} by using $\mu_{\min} = 10^{-q}\mu_{\text{map}}$ and $\mu_{\max} = \min(10^q \mu_{\text{map}}, 2\|\mathbf{A}^\top \mathbf{b}\|_\infty)$ for a suitably chosen q such that $F(\mu_{\min}) - p\sigma^2 < 0 < F(\mu_{\max}) - p\sigma^2$. We can then apply bisection to seek the root in this interval. Empirically we observe that it is more efficient to use a logarithmic bisection for getting close to a suitable root, given that the range for suitable μ may be large. We further note, as observed in [60, Remarks 1 and 2], that the bisection relies not only on finding a good interval for μ such that $F(\mu) - p\sigma^2$ goes through zero, but also, as already stated, on the availability of good estimates for p and σ^2. These limitations (i.e., providing statistics on the data and an estimate of the DF) are the same as the ones arising when using the standard DP, or the χ^2-estimate on the augmented residual in eq. (3), [61,63].

Bisection is terminated when one of the following conditions is satisfied: (i) $\mu_{\max} - \mu_{\min} < \tau_2(1 + |\mu_{\min}|)$; (ii) $|\widetilde{F}(\mu_{\max}) - \widetilde{F}(\mu_{\min})| < \tau_2$; (iii) $\mu_{\max} - \mu_{\min} < \tau_1$ or (iv) a maximum number of total function evaluations are reached. First, note that the limit on function evaluations is important because each evaluation requires finding $\widehat{\mathbf{x}}(\mu)$, i.e., solving eq. (2) for a given choice of μ. Second, the estimate τ_1 determines whether or not a tight estimate for μ_{χ^2} is required. In contrast, the parameter τ_2, provides a relative error bound on μ that is relevant for large μ. It also permits adjustment of a standard bisection algorithm to reflect the confidence in the knowledge of p and/or σ^2. For example, the DP is a χ^2 test on the satisfaction of the DF in the data fit term for Tikhonov regularization and is often adjusted by the introduction of a safety parameter η. Then, rather than seeking $\|\mathbf{A}\mathbf{x}_{\text{TIK}} - \mathbf{b}\|_2^2 = m\sigma^2$, the right-hand side is adjusted to $\eta m\sigma^2$ (here, $\mathbf{x}_{\text{TIK}}$ is the solution of eq. (3) dependent on the parameter μ). In the same manner, we may adjust τ_2 using the safety parameter η as a degree of confidence on the variance σ^2 or the DF. Since we solve relative to $p\sigma^2$, we can adjust using $\eta p\sigma^2$ for safety on σ^2 or using $\eta = (p + \zeta)/p$ when considering a confidence interval on the DF. For example, replacing p by $p + \zeta$ for $\zeta = \text{confidence}(p, 0.05)$ represents a 95% confidence interval in the χ^2 distribution, as used when applying the χ^2 estimate for the augmented residual [61]. A choice based on a confidence interval is more specific than an arbitrarily chosen η. Here we use $\eta = 1$.

It should also be noted that the solution $\widehat{\mathbf{x}}(\mu)$ is defined within the algorithm for a given choice of the shrinkage parameter γ, which should not be changed during the bisection. This means that as μ increases, with soft shrinkage parameter $\gamma = \mu/\lambda^2$ fixed, λ also increases, and hence the result for a specific μ_{χ^2} is dependent on a given shrinkage threshold.

5. Numerical experiments

In the following, we perform various large-scale numerical experiments to demonstrate the benefits of VPAL. We first demonstrate the performance of VPAL in comparison to a standard ADMM method on a denoising example in Section 5.1. We investigate its scalability in 3D medical tomography inversion in Section 5.2 and discuss regularization parameter selection methods in Section 5.3. Key metrics for our numerical investigations are the relative error and relative residual, respectively, defined by

$$e(\mathbf{x}_k) = \frac{\|\mathbf{x}_k - \mathbf{x}_{\text{true}}\|_2}{\|\mathbf{x}_{\text{true}}\|_2} \text{ and } r(\mathbf{x}_k) = \frac{\|\mathbf{A}\mathbf{x}_k - \mathbf{b}\|_2}{\|\mathbf{b}\|_2}.$$

When using $e(\widehat{\mathbf{x}}(\mu, \lambda))$, or equivalently $r(\widehat{\mathbf{x}}(\mu, \lambda))$, this refers to the converged (or final) value for the error, or residual, for the solution $\widehat{\mathbf{x}}$ for a given parameter set (μ, λ).

While various iterative techniques for the solution of eq. (10) have been discussed in the literature, including a generalized Krylov approach, e.g., [11], a standard approach is to use the LSQR algorithm, also based on a standard Krylov iteration [70] (using an algorithm consistent relative stopping tolerance, here, $\tau = 10^{-4}$). This is our method of choice when comparing Algorithm 2 with Algorithm 1, and when presenting results for ℓ^2 regularization. In computations where we report timings we are using a 2013 MacPro with a 2.7 GHz 12-core Intel Xeon E5 processor with 64 GB Memory and 1866 MHz DDR3 running with macOS Big Sur and Matlab 2021a.

5.1. Denoising experiment

In the first experiment, we investigate the convergence of VPAL, compared to ADMM. We consider an image denoising problem with $\mathbf{A} = \mathbf{I}_n$, while $\mathbf{b}$ is a Matlab test image `eSFRTestImage.jpg` of size $1{,}836 \times 3{,}084$, so that $\mathbf{x}_{\text{true}}$ is of size

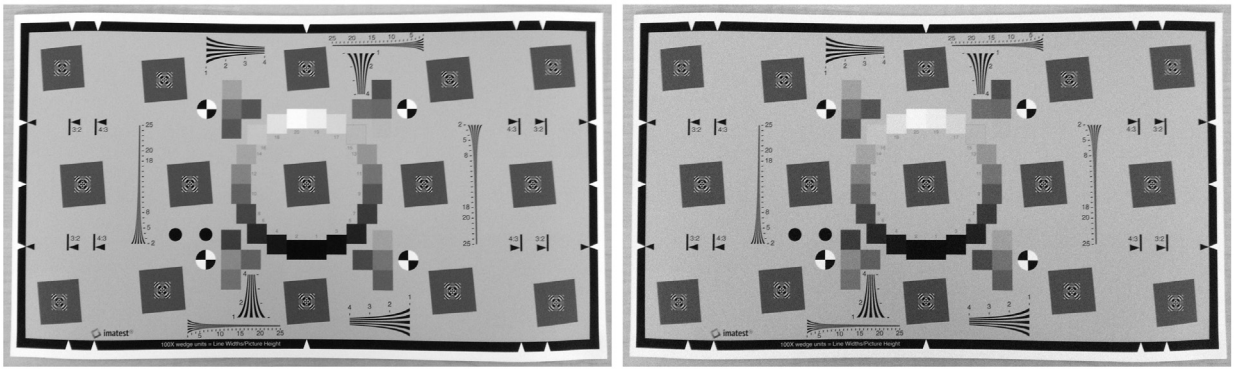

Fig. 1. On the left we show Matlab's test image `eSFRTestImage.jpg` representing $\mathbf{x}_{\text{true}}$, on the right we depict noisy observations represented by $\mathbf{b}$ with 10% Gaussian white noise which yields an SNR of 20.

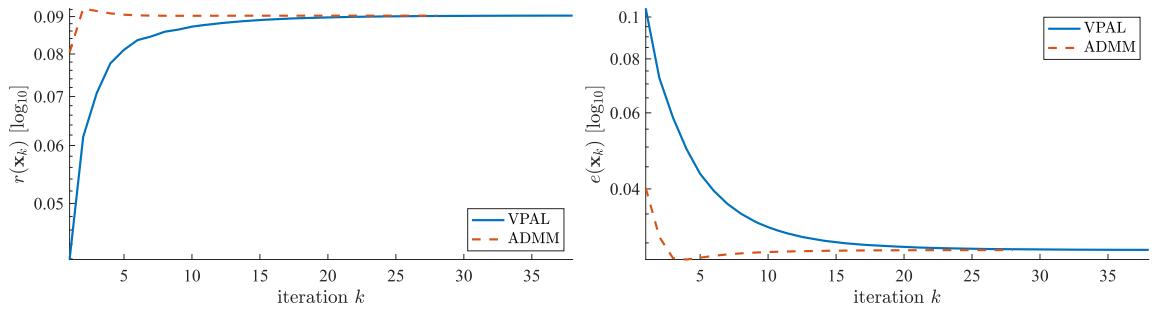

Fig. 2. The left panel displays the relative residual, while the right panel depicts the relative error for VPAL (solid blue line) and ADMM (dashed red line). Both methods converge to an approximation of $\mathbf{x}_{\text{true}}$ where the relative difference between ADMM and VPAL is negligible.

$n = 16{,}986{,}672$, see Fig. 1, left panel[2]. We contaminate $\mathbf{x}_{\text{true}}$ with 10% Gaussian white noise, which yields an SNR of 20, see Fig. 1 right panel. Here, we utilize a total variation regularization where $\mathbf{D}$ is the 2D finite difference matrix. We fix the regularization parameter as $\mu = 10$, select a stopping tolerance $\tau = 10^{-4}$, and observe the relative error $e(\mathbf{x}_k)$ and relative residual $r(\mathbf{x}_k)$ with iteration k for both methods. The final approximations are denoted by $\mathbf{x}_{\text{VPAL}}$ and $\mathbf{x}_{\text{ADMM}}$ with reconstruction errors $e(\mathbf{x}_{\text{VPAL}}) = 2.890 \cdot 10^{-2}$ and $e(\mathbf{x}_{\text{ADMM}}) = 2.887 \cdot 10^{-2}$, respectively. Results are displayed in Fig. 2, where it is shown that ADMM reaches the solution in fewer iterations than VPAL, 28 to 38 iterations, respectively. The number of outer iterations used in Algorithms 1 and 2 is, however, an ambiguous computational currency. Following the discussion above, the main computational cost of ADMM is the number of LSQR iterations utilized for each outer iteration k, while VPAL has the equivalent computational complexity of only one LSQR iteration for each k. Factoring in these computational costs, we notice that ADMM overall requires 141 LSQR iterations. In contrast VPAL requires 38 computations, indicating almost a factor of four improvement in the total number of LSQR iterations. Consequently, we expect VPAL to converge more rapidly.

Along this line, we extend our investigations and examine wall clock times, see Fig. 3. Using the same computational setup as before, we consider varying sizes of the image in Fig. 1, left panel. Note that in this setup we also slightly vary the regularization parameter $\mu \in [1, 20]$ to obtain more realistic timings for "near optimal" regularization parameters. Additionally, in our comparison, we include timings for a generalized Tikhonov approach of the form eq. (3). Here this is referred to as TIK and can be seen as a computational lower bound for VPAL and ADMM (left). Notice our comparison to a generalized Tikhonov approach comes with an asterisk, because (1) the range of good regularization parameters is largely different (to obtain close to optimal regularization parameters we choose $\widetilde{\mu} \in [0.1, 0.4]$) and (2) the Tikhonov approach leads to inferior reconstructions. Nevertheless, we confirm that despite its computational superiority, VPAL does not lack numerical accuracy in comparison to ADMM. We depict the relative errors $e(\mathbf{x}_{\text{VPAL}})$, $e(\mathbf{x}_{\text{ADMM}})$, and $e(\mathbf{x}_{\text{TIK}})$ in Fig. 3 right panel.

In Fig. 4 we show the average gain in computational speed-up of VPAL as compared to ADMM. Depicted is the ratio of the wall clock timing (ADMM/VPAL), confirming the computational advantage of VPAL with an approximate average speed-up of about 8. Additionally, we recorded the ratio of how many $\mathcal{O}((m + \ell)n)$ operations (the main computational costs of LSQR equivalent steps) on average each of the methods requires (ADMM/VPAL).

[2] A simplified demo VPAL code based on this example is provided in Appendix A. A VPAL implementation will be maintained and updated at `www.github.com/matthiaschung/vpal`.

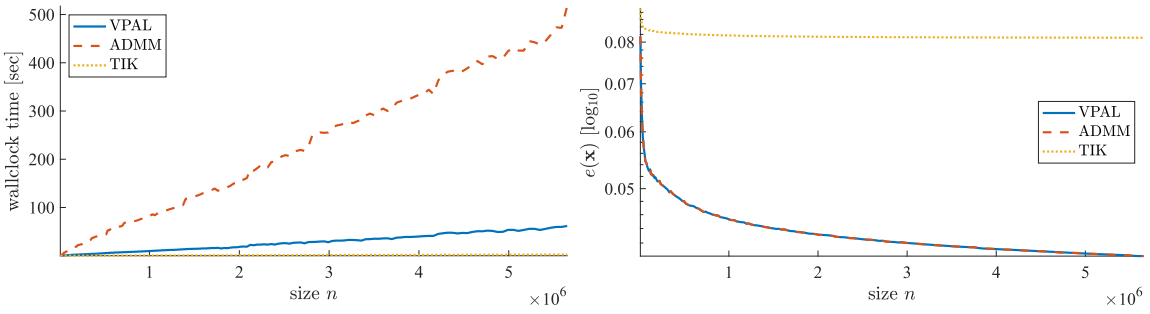

Fig. 3. The left plot shows the wall clock timing of VPAL (solid blue line), ADMM (dashed red line), and TIK (dotted yellow line) with increasing image size n. Depicted on the right are the corresponding relative reconstruction errors. Despite their differences in computational complexity, both VPAL and ADMM produce nearly identical reconstructions and are virtually indistinguishable, while TIK generates inferior reconstructions.

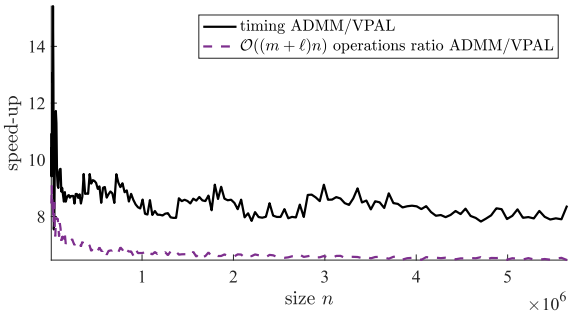

Fig. 4. Depicted is the computational gain of VPAL compared to ADMM. The solid black line shows the timing ratio VPAL/ADMM. The dashed violet line shows the ratio of $\mathcal{O}((m+\ell)n)$ operations (main computational cost) of ADMM compared to VPAL.

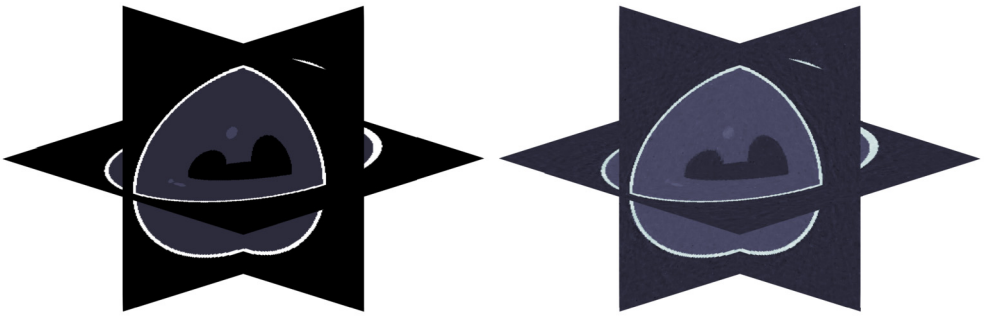

Fig. 5. The left panel illustrates the 3D Shepp-Logan phantom using slice planes, while the right panel shows the reconstructed Shepp-Logan phantom using VPAL.

Note, due to the similar curves in Fig. 4, we empirically confirm that the computational costs are dominated by the LSQR solves. Hence, this denoising experiment illustrates that VPAL compared to ADMM reconstructs images accurately at a reduced computational cost.

5.2. 3D medical tomography

To demonstrate the scalability of our VPAL method, we consider a 3D medical tomography application. As a ground truth, we consider the 3D Shepp-Logan phantom, illustrated in Fig. 5 (left panel) by slice planes. For the discretization of the Shepp-Logan phantom we use $255 \times 255 \times 255$ uniform cells, corresponding to $\mathbf{x}_{\text{true}} \in \mathbb{R}^{16,581,375}$. Data is generated by a parallel beam tomography setup using the `tomobox` toolbox [54]. We constructed projection images from 100 random directions each of size 255×255. The projection images are contaminated with 5% white noise, an SNR of approximately 26, see four sample projection images in Fig. 6. Consequently, the ray-tracing matrix $\mathbf{A}$ is of size $6{,}502{,}500 \times 16{,}581{,}375$. Note that $\mathbf{A}$ is underdetermined and therefore is outside the scope of Theorem 3.4 without convergence guarantees. As the regularization operator, we use 3D total variation. Here, $\mathbf{D}$ is of size $49{,}549{,}050 \times 16{,}581{,}375$ and uses zero boundary conditions. The regularization parameter is set to $\mu = 5$, and we use VPAL with its default tolerance set to $\tau = 10^{-6}$.

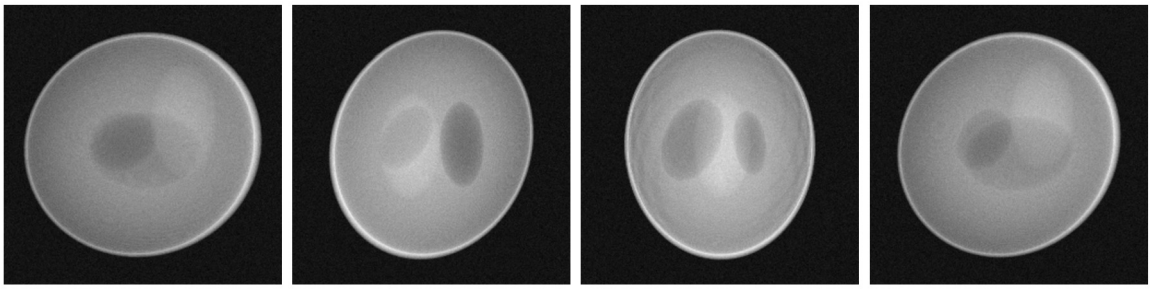

Fig. 6. Shown four projection images with 5% white noise and an SNR of approximately 26.

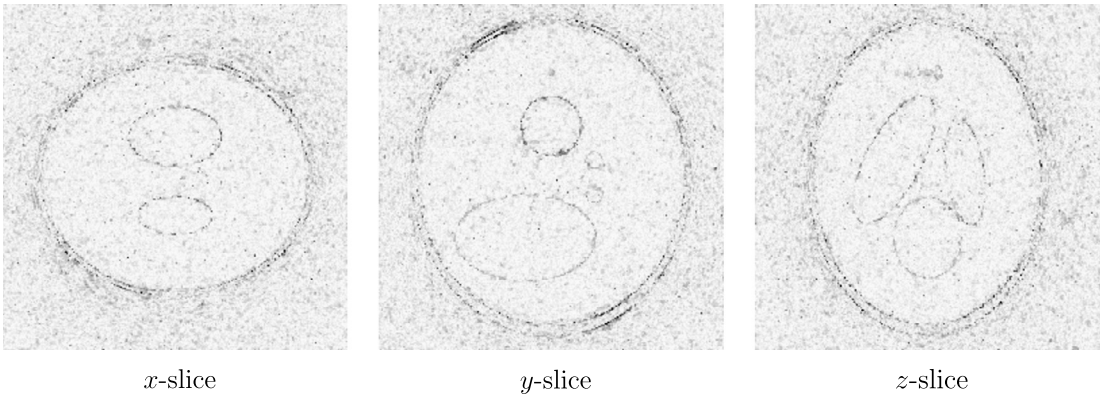

Fig. 7. Absolute error of 3D tomography image slices of the reconstruction $\mathbf{x}_{\text{VPAL}}$ in each principal (x, y, and z) coordinate direction with an inverted colormap, where darker gray values correspond to larger absolute errors with a largest absolute error of about 0.5346.

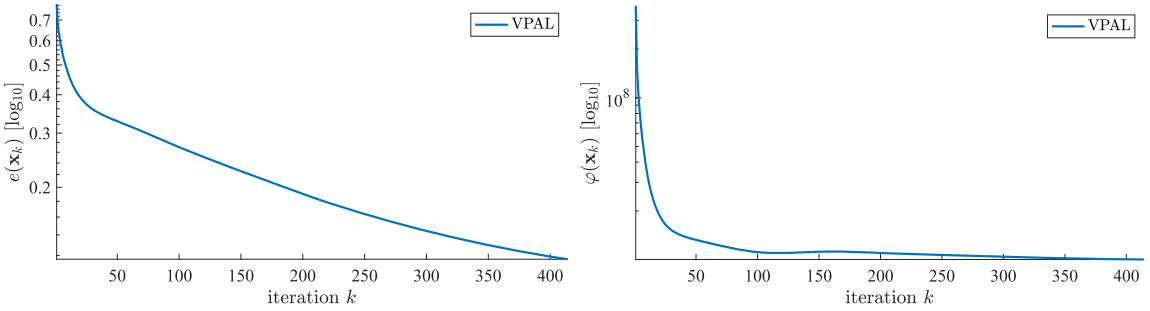

Fig. 8. The left graph shows the relative error of the reconstructions $\mathbf{x}_k$ at each iteration of VPAL of the 3D tomography problem. The right graph shows the corresponding objective function value $\varphi(\mathbf{x}_k)$ at each iteration. Note that with this under-determined $\mathbf{A}$ the relative error is monotonically decreasing, while the objective function value is not monotonically decreasing.

VPAL requires 414 iterations to converge within the given tolerance with a wall clock time of about 2.4 hours. The reconstructed 3D Shepp-Logan phantom $\mathbf{x}_{\text{VPAL}}$ is illustrated in Fig. 5 (right panel) by slice planes. The corresponding relative error is $e(\mathbf{x}_{\text{VPAL}}) = 0.1167$. A reconstruction error below 12% is noteworthy, considering that the matrix $\mathbf{A}$ is significantly underdetermined. Again, notice that a standard Tikhonov regularization ($\mathbf{D} = \mathbf{I}_n$) using LSQR is significantly faster (about 3.5 minutes); however, the relative reconstruction error is inferior. The Tikhonov approach does not generate a relative error below 35% even when utilizing an optimal regularization parameter (data not shown).

Reconstruction results are further illustrated in Figs. 7 and 8, which show the absolute error image, and relative reconstruction errors $\mathbf{x}_k$ of VPAL with the corresponding objective function value $\varphi(\mathbf{x}_k)$ at each iteration, respectively. We notice fast but not necessarily monotonically decreasing objective function values $\varphi(\mathbf{x}_k)$.

5.3. Experiments on regularization parameters

We discuss experiments using the bisection algorithm to identify μ_{χ^2}. As a test bed, we investigate a deblurring (`blur`), 2D medical tomography (`tomo`), and seismic (`seismic`) inversion problem. For each problem we utilize the Matlab Toolbox `IRtools`; see [32]. In all cases, we select Gaussian noise, and for `blur` we use a `severe shake blur` for the Hubble

space telescope, for `tomo` we use the 2D Shepp-Logan phantom, while for `seismic` we select the `tectonic` phantom. In all usages of the VPAL algorithm, we select a stopping tolerance of $\tau = 10^{-4}$ and limit the number of outer iterations to 1,000. For the bisection algorithm, we set $\tau_1 = 0.01$, $\tau_2 = 0.02$, limit the number of bisection steps to 10, and always use the safety parameter $\eta = 1$. Note that each bisection is a full solve of VPAL and thus the total maximum number of solves could be as high as 10,000 with these settings. There is a trade-off between the accuracy of the desired root and the potential for high computational overhead by carrying out the bisection. We investigate calculating the optimal μ using the χ^2 estimator for a fixed γ. We also investigate using the DP by applying bisection and root-finding for $H(\lambda) = \|\mathbf{A}\mathbf{x}(\lambda) - \mathbf{b}\|_2^2/m\sigma^2 - 1$.

We investigate each of these experiments for various image and data sizes. Our results are presented for four problem sizes with N ranging from 64 to 512, corresponding to image sizes $n = N^2$ (4,096, 16,384, 65,536, and 262,144) for problem `blur`, `tomo` and `seismic`. The size of the data vector $\mathbf{b}$ varies with the application, where $m = n$ for `blur`, $m = 16{,}380$, 32,580, 65,160, and 130,320 for `tomo` and $m = 8{,}192$, 32,768, 131,072, and 524,288 for `seismic`. These experiments correspond to $m = n$ for all the `blur` cases, $m > n$ for `seismic` (over-determined), and with `tomo` $m < n$ (under-determined) for the two larger experiments. In each of these experiments, we investigate white noise levels of 10% and 20%, corresponding to SNR of 20 and 13.98 in each case. The results are summarized in Figs. 9, 10, 11.

To describe the plots, we note that we first fix γ and estimate μ_{χ^2}, yielding the point (μ_{χ^2}, γ) indicated with the red inverted triangle. Given μ_{χ^2} we also calculate an optimal choice for λ using the DP, yielding the point $(\mu_{\chi^2}, \lambda_{\chi^2})$ indicated with the green open circle. Then for known μ_{map} we also apply the DP to find λ, yielding the point $(\mu_{\text{map}}, \lambda_{\text{map}})$ indicated with the blue triangle. The solid red and dashed blue lines are obtained by fixing μ_{χ^2} and μ_{map}, respectively, and then solving the problem for 50 choices of λ logarithmically spaced in the intervals $[\lambda_{\chi^2}/100, 100\lambda_{\chi^2}]$ and $[\lambda_{\text{map}}/100, 100\lambda_{\text{map}}]$, respectively. Note that fixing μ and varying λ is also equivalent to fixing μ and varying γ.

In these figures, the oscillations in the relative error curves occur if the algorithm did not converge within 1,000 iterations, which is more prevalent for small values of λ. Given that μ is fixed on these curves, this corresponds to taking larger values of the shrinkage parameter γ. Flat portions of the curves indicate the relative lack of sensitivity to the choice of λ (respectively γ) for a fixed μ, equivalently confirming that the optimal μ is largely independent of γ within a suitably determined range, dependent on the data and the problem.

We see immediately that finding μ_{χ^2} by the χ^2-DF test yields smaller relative errors in the solutions than when the solution is generated using μ_{map}, the circles and inverted triangles are lower than the triangles, and except where there are issues with convergence, the solid curves lie below the dashed curves. This is notwithstanding that the χ^2-DF test and the MAP estimators do not immediately apply for the `tomo` and `seismic` problems, since neither corresponds to a differentially Laplacian solution. Indeed, the Hubble space telescope image is also not a perfect example of such an image, but the results demonstrate that the approach still works reasonably well. On the other hand, comparing now inverted triangles and open circles contrasts the impact of finding μ_{χ^2} by the χ^2-DF test (inverted triangle) and then assessing whether the standard DP to find λ (open circle) might be a better option. It is particularly interesting that the χ^2-DF test does uniformly well on `tomo` and `seismic` problems, but there are a few cases with `blur` in which the DP finds a λ that yields a smaller relative error. Even in these cases, the results are good using the χ^2-DF result. In all situations, it is clear that we would not expect to find an optimal λ that is at the minimum point of the respective relative error curve, but in general, the results are acceptably close to these minimum points. Overall, these results support the approach in which we pick a shrinkage parameter γ and find μ_{χ^2} using the χ^2-DF test by bisection. The results presented for the DP approach to finding λ were provided to contrast the two directions for estimating the parameters. Indeed, it is clear that finding a μ to fit $F(\mu)$ given by eq. (21) to $m\sigma^2$ and then fitting the residual term also to $m\sigma^2$, by applying the DP to find λ, will necessarily increase the value of $F(\mu)$. This set of results demonstrates that there is no need to use the DP principle; rather, optimizing based on eq. (13) is appropriate for all the test problems.

To assess the effectiveness of finding optimal regularization parameters using the automatic approach, we also performed a parameter sweep over a grid of values for (μ, λ) for the experiments that use 10% noise, corresponding to an SNR of 20. For each point on the grid, we calculated both the relative error $e(\mathbf{x}(\mu, \lambda))$ and the χ^2 value, $|F(\mu, \lambda)/m\sigma^2 - 1|$. The left panels and right panels in Figs. 12 to 14 show the contour plots for the relative errors and χ^2 estimates, respectively. The circles correspond to the points with minimum error and the squares to the points with minimum χ^2 value. If they are close we would assert that the χ^2 is optimal for finding a good value for μ. In general, the contours are predominantly vertical, confirming that the solutions are less impacted by the choice of λ (and hence shrinkage γ) than of μ. Further, it is necessary to examine the values for the contours, given in the color bar, in order to assess whether the χ^2 is not giving a good solution. From Figs. 12 and 13 we can conclude that the difference in the relative error from using the χ^2 estimate for μ rather than the optimal in terms of the minimal error is small; all values lie within the contours for low values as shown in the colorbars. Even for the `seismic` case shown in Fig. 14 the contour level for the relative error changes only from about 0.06 to 0.09. The captions give the actual calculated relative errors for the circle and squares.

Finally, to demonstrate the applicability of the bisection algorithm for other differentially Laplacian operators $\mathbf{D}$, we present a small sample of results when $\mathbf{D}$ is the Laplace matrix. We use the same data sets `blur`, `tomo` and `seismic`, with the same noise levels corresponding to SNRs of 20 and 13.98, and the same problem sizes, and replace the TV matrix by the Laplace matrix. In Fig. 15 we collect the results for all problem sizes in one plot per test, where in each case the lines are solid, dashed, dotted and dash-dots for problem sizes 64, 128, 256 and 512, respectively. In each plot we show the relative error curves for a range of λ around the optimal μ for a specific shrinkage parameter γ (solid symbols) found using

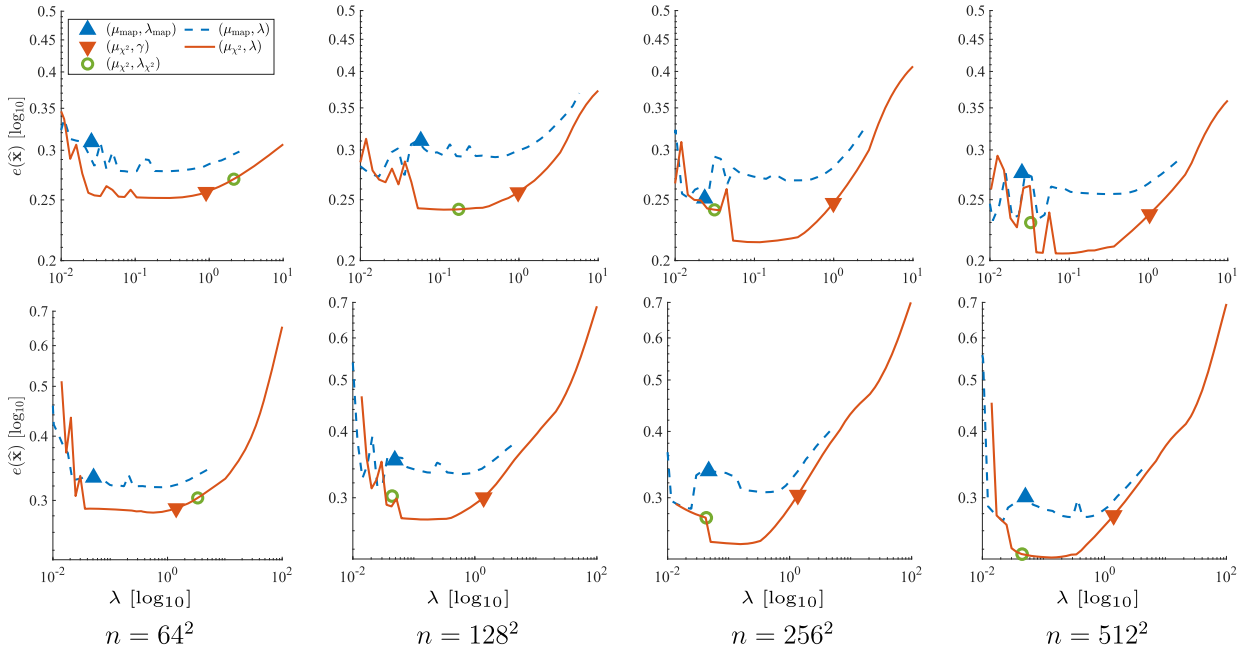

Fig. 9. Results for `blur` example for varying λ and fixed regularization parameter μ, μ_{map} and μ_{χ^2} for the solid blue and dashed red curves, respectively. Also marked for the given fixed μ choices are the optimal λ found using the DP. The first row shows relative errors for different image sizes ($n = 64^2, 128^2, 256^2$, and 512^2) with a noise level of 10% corresponding to a signal-to-noise ratio (SNR) of 20. The second row shows relative errors with a noise level of 20% corresponding to an SNR of 13.98.

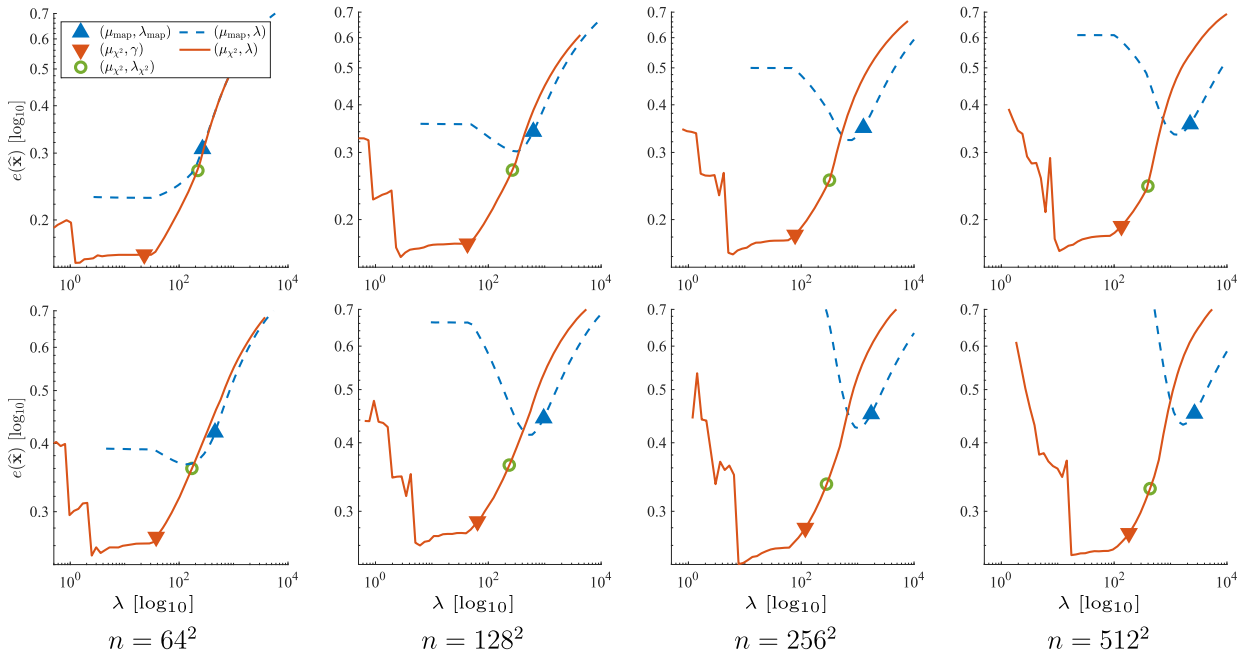

Fig. 10. Results for `tomo` example for varying λ and fixed regularization parameter μ, μ_{map} and μ_{χ^2} for the solid blue and dashed red curves, respectively. Also marked for the given fixed μ choices are the optimal λ found using the DP. The first row shows relative errors for different image sizes ($n = 64^2, 128^2, 256^2$, and 512^2) with a noise level of 10% corresponding to a signal-to-noise ratio (SNR) of 20. The second row shows relative errors with a noise level of 20% corresponding to an SNR of 13.98.

the χ^2-DF test. The results again demonstrate the ability of the bisection algorithm with the χ^2-DF test to find solutions with near-optimal relative errors.

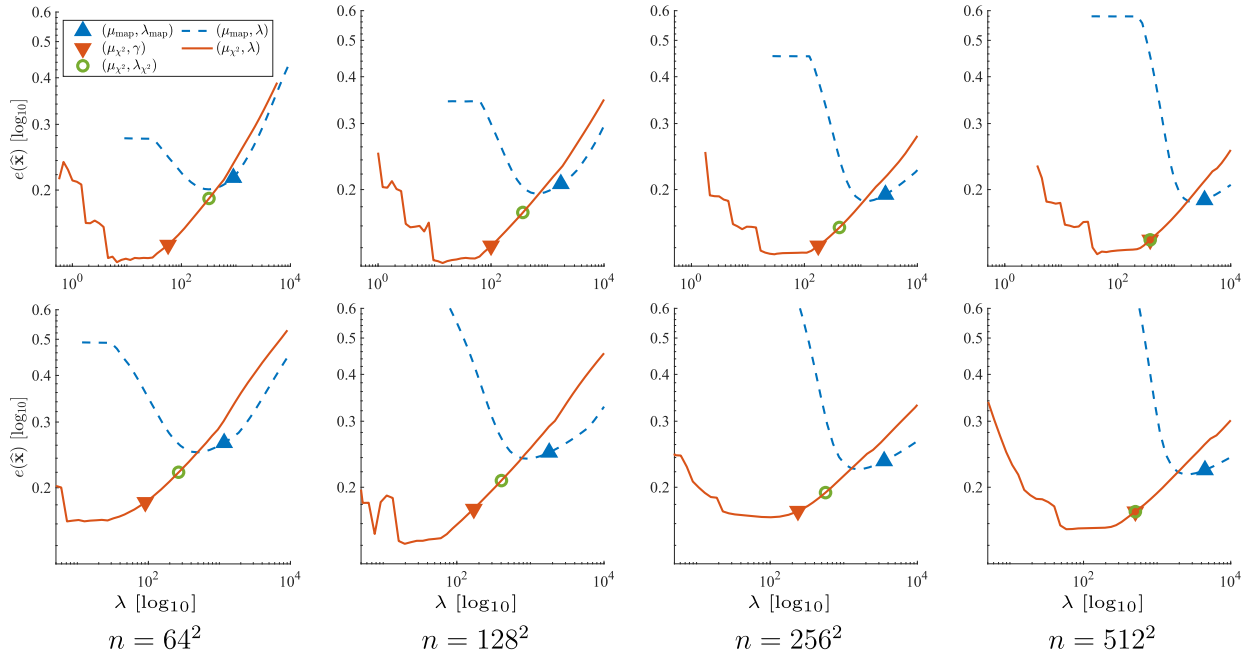

Fig. 11. Results for `seismic` example for varying λ and fixed regularization parameter μ, μ_{map} and μ_{χ^2} for solid blue and dashed red curves, respectively. Also marked for the given fixed μ choices are the optimal λ found using the DP. The first row shows relative errors for different image sizes ($n = 64^2, 128^2, 256^2$, and 512^2) with a noise level of 10% corresponding to a signal-to-noise ratio (SNR) of 20. The second row shows relative errors with a noise level of 20% corresponding to an SNR of 13.98.

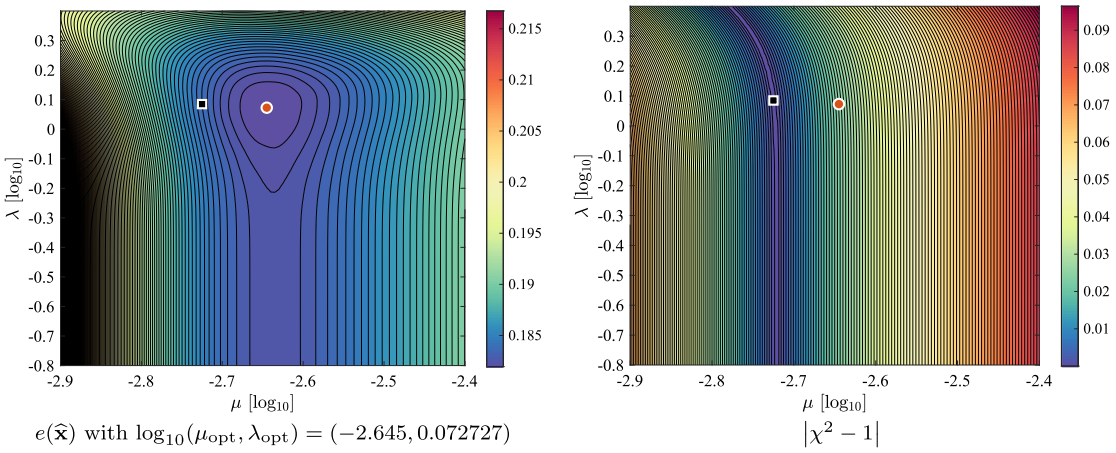

Fig. 12. Problem `blur`: Calculating the relative error and χ^2 departure from $m\sigma^2$ for a logarithmically uniform grid of points (μ, λ). The minimum relative error is at the red circle and the minimum for the χ^2 is at the black square. The obtained minimum values for the relative errors at these points are 0.1820 and 0.1840 and occur for shrinkage parameter $\gamma = 0.0016$, and 0.0013, respectively.

In summary, we observe empirically that VPAL converges particularly fast with only a few iterations, when a near-optimal regularization parameter is selected using the χ^2-DF test. Moreover, due to the efficiency of the approach, it is completely feasible to use VPAL for the multiple solves that are required in a root-finding algorithm to estimate μ_{χ^2} for fixed λ.

6. Discussion & conclusions

In this work, we presented a new method for solving generalized lasso problems using *variable projection* and inexact solves within a standard inner iteration. The resulting VPAL algorithm yields an efficient and provably convergent method. Our investigations included various numerical experiments illustrating the efficiency and effectiveness of our new VPAL method for different regularization operators **D**. The feasibility of using VPAL within the context of automatic determination of the regularization parameter for image restoration problems using a simple bisection root-finding method is also verified.

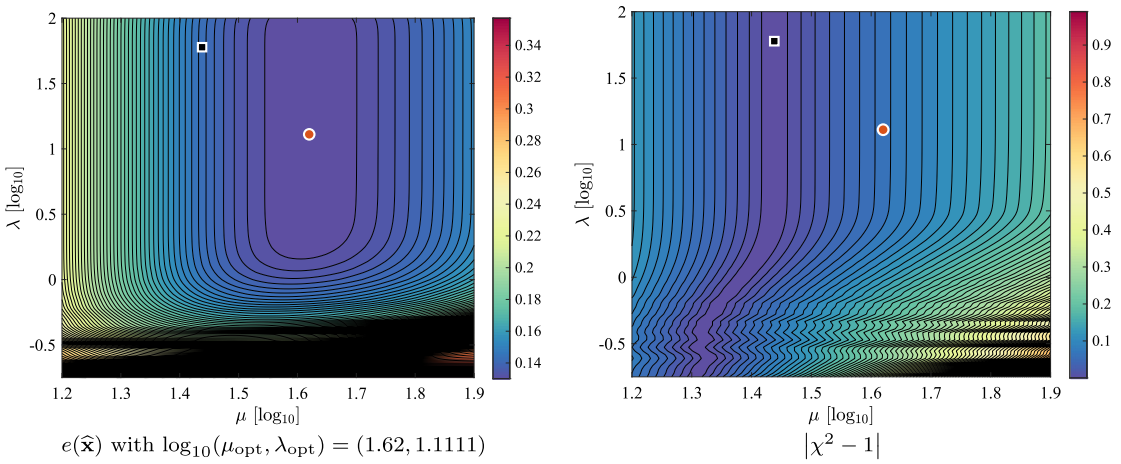

Fig. 13. Problem `tomo`: Calculating the relative error and χ^2 departure from $m\sigma^2$ for a logarithmically uniform grid of points (μ, λ). The minimum relative error is at the red circle and the minimum for the χ^2 is at the black square. The obtained minimum values for the relative errors at these points are 0.1301 and 0.1449 and occur for shrinkage parameter $\gamma = 0.2499$, and 0.0076, respectively.

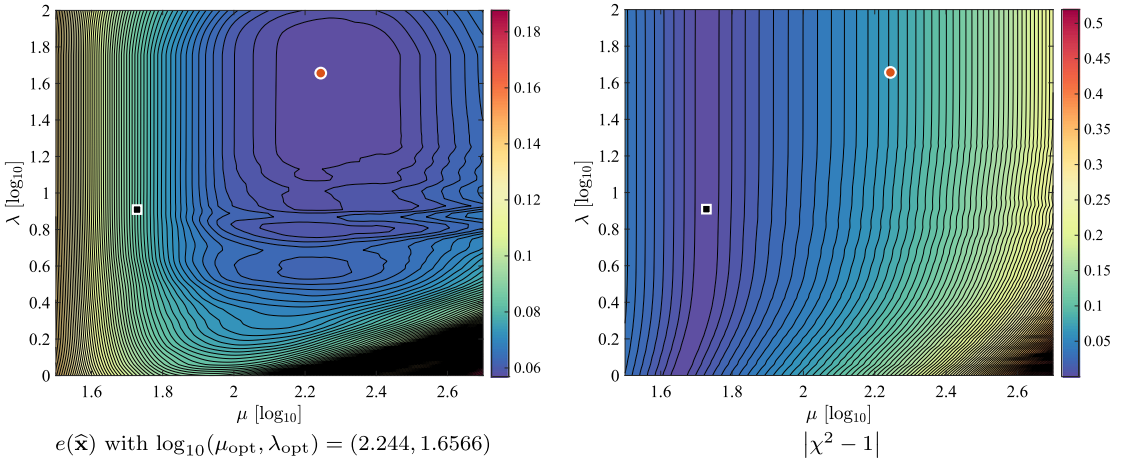

Fig. 14. Problem `seismic`: Calculating the relative error and χ^2 departure from $m\sigma^2$ for a logarithmically uniform grid of points (μ, λ). The minimum relative error is at the red circle and the minimum for the χ^2 is at the black square. The obtained minimum values for the relative errors at these points are 0.0569 and 0.0846 and occur for shrinkage parameter $\gamma = 0.0853$, and 0.8125, respectively.

Our investigation of the variable projection augmented Lagrangian method provides a new angle on solving generalized lasso problems and suggests several directions for future study. As already noted in Remark 6 necessary convergence results given in Theorem 3.4 for Algorithm 2 apply more generally. It is of future interest to investigate specific rate of convergence results for the inexact formulation, utilizing results from inexact ADMM methods e.g., [45]. Second, although we utilize a CG update for the inner iteration, other update strategies may be employed. Since this is a nonlinear problem we may for instance utilize LBFGS updates or nonlinear Krylov subspace methods. Third, estimating a good regularization parameter μ remains a costly task. In [23] iterative regularization approaches estimating μ on a subspace were investigated and demonstrated a computational advantage. It remains to be seen whether the same direction is robust when integrated within VPAL. Fourth, while here we consider $\ell^2 - \ell^1$ norm regularization, the developed approach extends also to other $\ell^p - \ell^q$ norm problems [20,18] and even to more general objective functions. We will investigate the convergence and numerical advantages and disadvantages of utilizing a variable projection approach for such $\ell^p - \ell^q$ problems and for supervised learning loss function fitting within this framework. Fifth, row action methods have been developed to solve least squares and Tikhonov-type problems of extremely large scale where the forward operator $\mathbf{A}$ is too large to keep in computer memory [22,75]. We will investigate how VPAL can be extended to such settings and investigate convergence properties and sampled regularization approaches. Sixth, we will investigate and extend our variable projection optimization method to other suitable optimization problems such as for efficiently solving the Sylvester equations [7]. Further, due to the nature of the *variable projection* approach VPAL extends naturally to nonlinear models $\mathbf{A} : \mathbb{R}^n \to \mathbb{R}^m$ which will be a topic of future investigation.

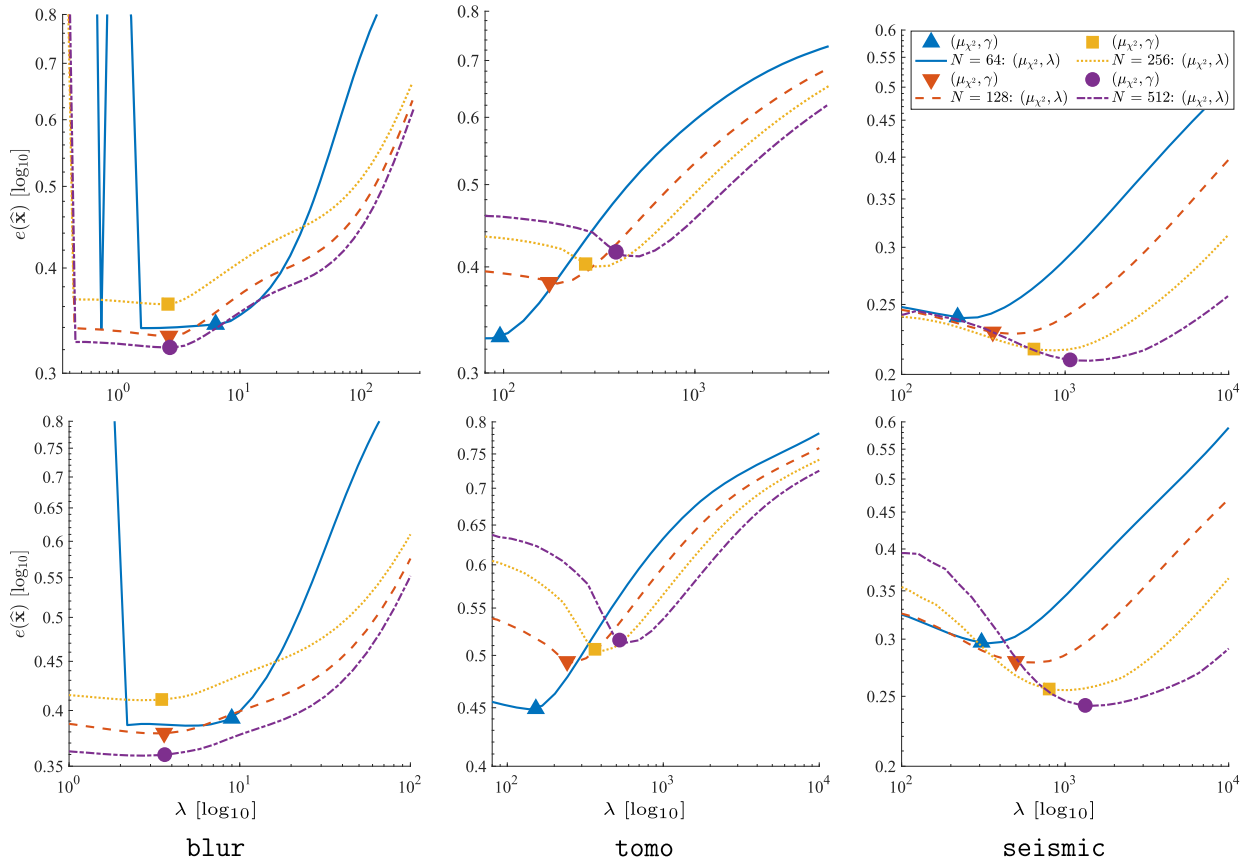

Fig. 15. Results for the `blur`, `tomo` and `seismic` problems using the Laplace operator for **D**. Here solid blue, dashed red, dotted yellow, and dash-dot purple curves and solid symbols correspond to results for problems of image sizes $n = 64^2$, 128^2, 256^2, and 512^2, respectively. The solid symbols indicate the points selected by using the bisection algorithm applied to satisfy the χ^2-DF test. The first row shows relative errors with a noise level of 10% corresponding to a signal-to-noise ratio (SNR) of 20. The second row shows relative errors with a noise level of 20% corresponding to an SNR of 13.98.

CRediT authorship contribution statement

Matthias Chung: Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. **Rosemary A. Renaut:** Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft.

Acknowledgements

Funding: This work was partially supported by the National Science Foundation (NSF) under grants [DMS-1723005, DMS-2152661] and [DMS-1913136, DMS-2152704] for Chung and Renaut, respectively. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work was initiated as a part of the SAMSI Program on Numerical Analysis in Data Science in 2020. The authors would like to thank Michael Saunders and Volker Mehrmann for their many helpful suggestions and for their constructive feedback on an early draft of this paper. The authors would also like to thank the anonymous reviewers for pointing out important connections in the literature.

Appendix A. Matlab code

A.1. `vpal.m`

```
function [x, f, info] = vpal(A, b, options)
% function [x, f, info] = vpal(A, b, options)
%
% Authors:
%   (c) Matthias Chung (e-mail: matthias.chung@emory.edu) in June 2023
%       Rosemary Renaut
%
```

```
% MATLAB Version: 9.10.0.1649659 (R2021a) Update 1
%
% Version 1.0 (simple)
%
% Description:
%   Variable projected augmented Lagrangian method for solving l_1
%   regularized problems
%
%        x = argmin_x  f(x) = 1/2||Ax-b||^2 + mu ||Dx||_1
%
%   where D is the convolution operator, mu > 0 (optional s.t. xmin <= x <= xmax)
%
% Input arguments:
%   A           - forward model (m x n) may be an object
%                 if A is scalar or empty, method is denoising
%   b           - observation vector (m x 1) (column vector)
%   options       [further options of algorithm]
%     .x            - inital guess of x [default x = 0]
%     .mu           - regularization parameter
%     .lambda       - Lagrange multiplier
%     .tol          - tolerance [default 1e-4 ]
%     .D            - dOperator object in regularization term ||Dx||_1
%     .maxIter      - maximal number of iterations [ 10 * length(b) ]
%     .display      - print to display [ {'off'} | 'iter' | 'final' ]
%
% Output arguments:
%   x           - local minimizer
%   f           - normalized loss (Ax-b)/norm(b)
%   iter        - number of SB iterations used -needed for sanity check  RR8 remove later
%   info          [additional info on algorithm]
%      .iter    - number of iterations
%      .f       - function value
%      .tol     - selected tolerance
%      .maxIter - selected maximum number of iterations
%      .mu      - regularization parameter during iteration
%      .lambda  - Lagrange multiplier during iteration lambda^2
%      .stop    - stopping criteria during iteration (f, x, maxIter)
%      .D       - d-operator
%

% initialize default input options
maxIter = 10*size(b,1); display = 'off'; tol = 1e-4; xtrue = []; lambda = 1; mu = 0;

if nargin == nargin(mfilename) % rewrite default parameters if needed
  for j = 1:2:length(options)
    eval([options{j},'= options{j+1};'])
  end
end

if nargout > 2, getInfo = 1; else, getInfo = 0; end

if getInfo % general info of method
  info.tol      = tol;
  info.maxIter  = maxIter;
end

% display and algorithm info
if strcmp(display, 'iter') || strcmp(display,'final')
  fprintf('\nvpal algorithm (simple) (c) M. Chung & R. Renaut, June 2023\n');
  if strcmp(display,'final') == 0
    fprintf('\n %-10s %-7s %-10s \n','iter','loss','stop criteria');
  end
end

if max(size(A)) == 1; n_A = size(b,1); else, n_A = size(A,2); end % get number of unknowns

if ~exist('D','var') % no operator is provided, default is identity
  D = dOperator('identity',n_A);
elseif isa(D,'dOperator') % operator is provided and of class dOperator
    m_D = D.sizes(1);
else
    m_D = size(D,1); % operator is provided and is matrix
end


c = zeros(m_D,1); y = c; normb = norm(b); f = inf; xOld = inf;             % initialize
iter = 1; lambda2 = lambda^2;
if exist('x','var')
    Dx = D*x;  r = A*x-b;
```

```
else
    Dx = c; x = zeros(n_A,1); r = -b;
end

while 1

  % step 1 Tikonov CG update
  g = A'*r + lambda2 *(D'*(Dx - (y-c)));
  Ag = A*g; Dg = D*g;
  x = x - ((g'*g)/((Ag'*Ag) + lambda2*(Dg'*Dg)))*g;

  % update Dx and residual
  r = A*x - b; Dx = D*x;

  % step 2 shrinkage
  c = Dx + c; y = sign(c).*max(abs(c) -  mu/lambda2, 0);

  % step 3 update c
  c = c - y;

  % calculate loss
  fOld = f; f = 0.5*norm(r)^2 + mu*norm(Dx,1);

  % stopping criteria
  stop1 = abs(fOld - f)  <= tol * (1 + f);
  stop2 = norm(xOld - x,'inf') <= sqrt(tol) * (1 + norm(x,'inf'));
  stop3 = iter > maxIter-1;

  if getInfo
    info.f(iter)        = f;
  end

  if strcmp(display,'iter') % display iteration results
    fprintf('%5d %14.6e %4d%1d%1d\n', iter, f, stop1, stop2, stop3);
  end

  if (stop1 && stop2) || stop3 % check stop criteria
    if stop3
      warning('Matlab:vpal:maxIter',...
        'Maximum number of iterations reached. Return with recent values.')
    end
    break;
  end

  xOld = x; iter = iter + 1;

end

if (strcmp(display,'iter') || strcmp(display,'final')) && ~stop3 % display
  fprintf('\nLocal minimizer found. Function value is %1.8e.\n', f);
end
```

A.2. *dOperator.m*

```
classdef dOperator
    % classdef dOperator
    %
    % Authors:
    %   (c) Matthias Chung (matthias.chung@emory.edu) and Rosemary Renaut in June 2023
    %
    % MATLAB Version: 9.11.0.1769968 (R2021b)
    %
    % Version 1.0 (simple)
    %
    % Description:
    %   dOperator provides a class fo finite-difference
    %
    %  D = dOperator(dimension)
    %
    % Properties:
    %     dimension  - provides dimension of object for matrix with object dimension m x n
    %     sizes      - provides the dimension of the matrix D
    %     transposed - flag if operator is transposed or not

    properties
        dimension
        sizes
        transposed
    end
```

```
    methods

        % initialize D operator
        function D = dOperator(dimension)
            D.dimension  = dimension;         % provide dimension of operator
            D.sizes = [2*prod(D.dimension)-sum(D.dimension), prod(D.dimension)];
            D.transposed = false;
        end

        % transpose method
        function D = ctranspose(D)
            D.transposed = not(D.transposed);
            D.sizes = flip(D.sizes);
        end

        % size method
        function s = size(D,dim)
            if nargin < 2
                s = D.sizes;
            else
                s = D.sizes(dim);
            end
        end

        % mtimes method
        function x = mtimes(D, x)
            if D.sizes(2)~=size(x,1), error('size mismatch'); end
            if ~D.transposed
                x = reshape(x,D.dimension(1),D.dimension(2));
                x = [reshape(diff(x,1,1), D.dimension(2)*(D.dimension(1)-1),1); ...
                    reshape(diff(x,1,2), D.dimension(1)*(D.dimension(2)-1),1)];
            else % transposed case
                z1 = zeros(1,D.dimension(2)); % augment zeros
                z2 = zeros(D.dimension(1),1); % augment zeros
                split = D.dimension(2)*(D.dimension(1)-1);
                x1 = x(1:split);
                x2 = x(split+1:end); % split
                x1 = reshape(x1, D.dimension(1)-1, D.dimension(2) );
                x2 = reshape(x2, D.dimension(1),  D.dimension(2)-1);
                x = -reshape(diff([z1;x1;z1],1,1) + diff([z2,x2,z2],1,2),prod(D.dimension),1);
            end

        end

    end
end
```

A.3. *driverDenoisingExample.m*

```
% This is a matlab driver file for a denoising example using the vpal
% method. The corresponding optimization problem is
%      min_x   1/2||x-b||^2 + mu*|Dx||_1,
% where D is the finite difference operator. This example requires the
% following files: vpal.m and dOperator.m
%
%(c) Matthias Chung (e-mail: matthias.chung@emory.edu) and Rosemary Renaut in June 2023
%
clc, clear, close all                                                    % set fresh start

rng(0)                                                                   % set random seed for consistent result
xtrue = double(imread('cameraman.tif')); [m,n] = size(xtrue);            % load image
A = 1; b = xtrue + 10*randn(m,n);                                        % setup denoising problem
D = dOperator([m,n]);                                                    % define finite difference operator
[x, ~, info] = vpal(A,b(:),{'D',D,'mu',5,'display','iter'});             % run vpal method

sgtitle('Simple Denoising Example')                                      % visualize results
subplot(1,3,1), imshow(xtrue,[]), title('true')                          % original image
subplot(1,3,2), imshow(b,[]), title('noisy image')                       % noisy image
xlabel(['rel. error: ', num2str(norm(b(:) - xtrue(:))/norm(xtrue(:)))])
subplot(1,3,3), imshow(reshape(x,m,n),[]), title('vpal reconstruction')  % reconstructed image
xlabel(['rel. error: ', num2str(norm(x - xtrue(:))/norm(xtrue(:)))])
```

References

[1] Vando A. Adona, Max L.N. Gonçalves, An inexact version of the symmetric proximal admm for solving separable convex optimization, Numer. Algorithms (2023).

[2] Babak Maboudi Afkham, Julianne Chung, Matthias Chung, Learning regularization parameters of inverse problems via deep neural networks, Inverse Probl. 37 (10) (2021).

[3] Alnur Ali, Ryan J. Tibshirani, The generalized Lasso problem and uniqueness, Electron. J. Stat. 13 (2) (2019) 2307–2347.

[4] Johnathan M. Bardsley, Computational Uncertainty Quantification for Inverse Problems, SIAM, 2018.

[5] Amir Beck, First-Order Methods in Optimization, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017.

[6] Amir Beck, Marc Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM J. Imaging Sci. 2 (1) (2009) 183–202.
[7] Peter Benner, Ren-Cang Li, Ninoslav Truhar, On the ADI method for Sylvester equations, J. Comput. Appl. Math. 233 (4) (2009) 1035–1045.
[8] Dimitri Bertsekas, Nonlinear Programming, Athena Scientific, Belmont, Massachusetts, 1995.
[9] Dimitri Bertsekas, Constrained Optimization and Lagrange Multiplier Methods, Academic Press, 2014.
[10] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends Mach. Learn. 3 (1) (2011) 1–122.
[11] Alessandro Buccini, Fast alternating direction multipliers method by generalized Krylov subspaces, J. Sci. Comput. 90 (1) (2021) 60.
[12] Alessandro Buccini, Monica Pragliola, Lothar Reichel, Fiorella Sgallari, A comparison of parameter choice rules for ℓ^p-ℓ^q minimization, Ann. Univ. Ferrara 68 (2) (2022) 441–463.
[13] Alessandro Buccini, Lothar Reichel, Limited memory restarted ℓ^p-ℓ^q minimization methods using generalized Krylov subspaces, Adv. Comput. Math. 49 (2) (2023) 26.
[14] Martin Burger, Alexander Sawatzky, Gabriele Steidl, Splitting methods in communication, imaging, science, and engineering, Chapter first order algorithms in variational image processing, in: Scientific Computation, Springer Science & Business Media, 2017, pp. 345–407.
[15] Daniela Calvetti, Erkki Somersalo, An Introduction to Bayesian Scientific Computing: Ten Lectures on Subjective Computing, vol. 2, Springer Science & Business Media, 2007.
[16] Emmanuel J. Candes, Terence Tao, Decoding by linear programming, IEEE Trans. Inf. Theory 51 (12) (2005) 4203–4215.
[17] Antonin Chambolle, Thomas Pock, A first-order primal-dual algorithm for convex problems with applications to imaging, J. Math. Imaging Vis. 40 (2011) 120–145.
[18] Raymond H. Chan, Hai-Xia Liang, Half-quadratic algorithm for ℓ_p-ℓ_q problems with applications to TV-image restoration and compressive sensing, in: Efficient Algorithms for Global Optimization Methods in Computer Vision, in: Revised Selected Papers, International Dagstuhl Seminar, Dagstuhl Castle, Germany, November 20-25, 2011, pp. 78–103, Springer, 2014.
[19] Gong Chen, Marc Teboulle, A proximal-based decomposition method for convex minimization problems, Math. Program. 64 (1) (1994) 81–101.
[20] Julianne Chung, Matthias Chung, Silvia Gazzola, Mirjeta Pasha, Efficient learning methods for large-scale optimal inversion design, in: Numerical Algebra, Control and Optimization, 2022.
[21] Julianne Chung, Matthias Chung, J. Tanner Slagel, Iterative sampled methods for massive and separable nonlinear inverse problems, in: International Conference on Scale Space and Variational Methods in Computer Vision, Springer, 2019, pp. 119–130.
[22] Julianne Chung, Matthias Chung, J. Tanner Slagel, Luis Tenorio, Sampled limited memory methods for massive linear inverse problems, Inverse Probl. 36 (5) (2020) 054001.
[23] Julianne Chung, Silvia Gazzola, Computational methods for large-scale inverse problems: a survey on hybrid projection methods, arXiv preprint, arXiv: 2105.07221, 2021.
[24] Sören Dittmer, Tobias Kluth, Peter Maass, Daniel Otero Baguer, Regularization by architecture: a deep prior approach for inverse problems, J. Math. Imaging Vis. 62 (3) (2020) 456–470.
[25] Charles Dossal, Maher Kachour, Jalal M. Fadili, Gabriel Peyré, Christophe Chesneau, The degrees of freedom of the Lasso for general design matrix, Stat. Sin. 23 (2) (2013) 809–828.
[26] Jonathan Eckstein, Dimitri P. Bertsekas, On the Douglas—Rachford splitting method and the proximal point algorithm for maximal monotone operators, Math. Program. 55 (1) (1992) 293–318.
[27] Bradley Efron, The estimation of prediction error: covariance penalties and cross-validation, J. Am. Stat. Assoc. 99 (467) (2004) 619–632.
[28] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, Least angle regression, Ann. Stat. 32 (2) (2004) 407–499.
[29] Ernie Esser, Applications of Lagrangian-based alternating direction methods and connections to split Bregman, CAM Rep. 9 (2009) 31.
[30] Masao Fukushima, Application of the alternating direction method of multipliers to separable convex programming problems, Comput. Optim. Appl. 1 (1) (1992) 93–111.
[31] Daniel Gabay, Bertrand Mercier, A dual algorithm for the solution of nonlinear variational problems via finite element approximation, Comput. Math. Appl. 2 (1) (1976) 17–40.
[32] Silvia Gazzola, Per Christian Hansen, James G. Nagy IR, Tools: a MATLAB package of iterative regularization methods and large-scale test problems, Numer. Algorithms 81 (3) (2019) 773–811.
[33] Pascal Getreuer, Rudin-Osher-Fatemi total variation denoising using split Bregman, in: Image Processing on Line, vol. 2, 2012, pp. 74–95.
[34] Philip E. Gill, Walter Murray, Margaret H. Wright, Practical Optimization, SIAM, 2019.
[35] Roland Glowinski, Americo Marroco, Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de Dirichlet non linéaires, ESAIM: Math. Model. Numer. Anal. 9 (R2) (1975) 41–76.
[36] Tom Goldstein, Brendan O'Donoghue, Simon Setzer, Richard Baraniuk, Fast alternating direction optimization methods, SIAM J. Imaging Sci. 7 (3) (2014) 1588–1623.
[37] Tom Goldstein, Stanley Osher, The split Bregman method for L1-regularized problems, SIAM J. Imaging Sci. 2 (2) (2009) 323–343.
[38] Gene Golub, Victor Pereyra, Separable nonlinear least squares: the variable projection method and its applications, Inverse Probl. 19 (2) (2003) R1.
[39] Gene H. Golub, Michael Heath, Grace Wahba, Generalized cross-validation as a method for choosing a good ridge parameter, Technometrics 21 (2) (1979) 215–223.
[40] Gene H. Golub, Victor Pereyra, The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate, SIAM J. Numer. Anal. 10 (2) (1973) 413–432.
[41] E.G. Gol'shtein, N.V. Tret'yakov, Modified Lagrangians in convex programming and their generalizations, in: Point-to-Set Maps and Mathematical Programming, Springer, 1979, pp. 86–97.
[42] Mark L. Green, Statistics of images, the TV algorithm of Rudin-Osher-Fatemi for image denoising and an improved denoising algorithm, Technical report, UCLA CAM 02-55, 2002.
[43] Anne Greenbaum, Iterative Methods for Solving Linear Systems, SIAM, 1997.
[44] Jacques Salomon Hadamard, Lectures on Cauchy's Problem in Linear Differential Equations, Yale University Press, New Haven, 1923.
[45] William W. Hager, Hongchao Zhang, Inexact alternating direction methods of multipliers for separable convex optimization, Comput. Optim. Appl. 73 (1) (2019) 201–235.
[46] William W. Hager, Hongchao Zhang, Convergence rates for an inexact ADMM applied to separable convex optimization, Comput. Optim. Appl. 77 (3) (2020) 729–754.
[47] Per Christian Hansen, Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion, SIAM, 1998.
[48] Per Christian Hansen, Discrete Inverse Problems: Insight and Algorithms, SIAM, 2010.
[49] Per Christian Hansen, James G. Nagy, Dianne P. O'Leary, Deblurring Images: Matrices, Spectra, and Filtering, SIAM, 2006.
[50] Bingsheng He, Xiaoming Yuan, On the $o(1/n)$ convergence rate of the Douglas–Rachford alternating direction method, SIAM J. Numer. Anal. 50 (2) (2012) 700–709.
[51] R. Magnus Hestenes, Multiplier and gradient methods, J. Optim. Theory Appl. 4 (5) (1969) 303–320.
[52] Magnus Rudolph Hestenes, Eduard Stiefel, et al., Methods of Conjugate Gradients for Solving Linear Systems, J. Res. Natl. Bur. Stand. 49 (6), NBS Washington, DC (1952) 2379.

[53] Guangxin Huang, Alessandro Lanza, Serena Morigi, Lothar Reichel, Fiorella Sgallari, Majorization–minimization generalized Krylov subspace methods for ℓ_p-ℓ_q optimization applied to image restoration, BIT Numer. Math. 57 (2) (2017) 351–378.
[54] J. Jorgensen, Tomobox, https://www.mathworks.com/matlabcentral/fileexchange/28496-tomobox?s_tid=prof_contriblnk. (Accessed June 2023).
[55] Seung-Jean Kim, K. Koh, M. Lustig, Stephen Boyd, Dimitry Gorinevsky, An interior-point method for large-scale ℓ_1-regularized least squares, IEEE J. Sel. Top. Signal Process. 1 (4) (2007) 606–617.
[56] Nikos Komodakis, Jean-Christophe Pesquet, Playing with duality: an overview of recent primal-dual approaches for solving large-scale optimization problems, CoRR, arXiv:1406.5429 [abs], 2014.
[57] Andreas Langer, Automated parameter selection for total variation minimization in image restoration, J. Math. Imaging Vis. 57 (2) (2017) 239–268.
[58] Alessandro Lanza, Serena Morigi, Lothar Reichel, Fiorella Sgallari, A generalized Krylov subspace method for ℓ_p-ℓ_q minimization, SIAM J. Sci. Comput. 37 (5) (2015) S30–S50.
[59] Alessandro Lanza, Monica Pragliola, Fiorella Sgallari, Residual whiteness principle for parameter-free image restoration, Electron. Trans. Numer. Anal. 53 (2020) 329–351.
[60] Jodi L. Mead, Chi-squared test for total variation regularization parameter selection, Inverse Probl. Imaging 14 (3) (2020) 401–421.
[61] Jodi L. Mead, Rosemary A. Renaut, A Newton root-finding algorithm for estimating the regularization parameter for solving ill-conditioned least squares problems, Inverse Probl. 25 (2) (2008) 025002.
[62] Max A. Meju, Geophysical Data Analysis: Understanding Inverse Problem Theory and Practice, Society of Exploration Geophysicists, 1994.
[63] Vladimir Alekseevich Morozov, On the solution of functional equations by the method of regularization, Sov. Math. Dokl. 7 (1966) 414–417.
[64] Elizabeth Newman, Julianne Chung, Matthias Chung, Lars Ruthotto, slimTrain–a stochastic approximation method for training separable deep neural networks, SIAM J. Sci. Comput. 44 (4) (2022) A2322–A2348.
[65] Elizabeth Newman, Lars Ruthotto, Joseph Hart, Bart van Bloemen Waanders, Train like a (Var) Pro: efficient training of neural networks with variable projection, SIAM J. Math. Data Sci. 3 (4) (2021) 1041–1066.
[66] Michael K. Ng, Pierre Weiss, Xiaoming Yuan, Solving constrained total-variation image restoration and reconstruction problems via alternating direction methods, SIAM J. Sci. Comput. 32 (5) (2010) 2710–2736.
[67] Jorge Nocedal, Stephen Wright, Numerical Optimization, Springer Science & Business Media, 2006.
[68] Stanley Osher, Martin Burger, Donald Goldfarb, Jinjun Xu, Wotao Yin, An iterative regularization method for total variation-based image restoration, Multiscale Model. Simul. 4 (2) (2005) 460–489.
[69] Dianne P. O'Leary, Bert W. Rust, Variable projection for nonlinear least squares problems, Comput. Optim. Appl. 54 (3) (2013) 579–593.
[70] Christopher C. Paige, Michael A. Saunders LSQR, An algorithm for sparse linear equations and sparse least squares, ACM Trans. Math. Softw. 8 (1) (1982) 43–71.
[71] Neal Parikh, Stephen Boyd, Proximal algorithms, Found. Trends Optim. 1 (3) (2014) 127–239.
[72] Michael J.D. Powell, A method for nonlinear constraints in minimization problems, Optimization (1969) 283–298.
[73] J. Carlos Santamarina, Dante Fratta, Discrete Signals and Inverse Problems: An Introduction for Engineers and Scientists, John Wiley & Sons, 2005.
[74] Jonas Sjoberg, Mats Viberg, Separable non-linear least-squares minimization-possible improvements for neural net fitting, in: Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop, IEEE, 1997, pp. 345–354.
[75] J. Tanner Slagel, Julianne Chung, Matthias Chung, David Kozak, Luis Tenorio, Sampled Tikhonov regularization for large linear inverse problems, Inverse Probl. 35 (11) (2019) 114008.
[76] Stefan M. Stefanov, Separable Programming: Theory and Methods, vol. 53, Springer Science & Business Media, 2001.
[77] Yueyang Teng, Hang Sun, Chen Guo, Yan Kang, ADMM-EM method for l_1-norm regularized weighted least squares pet reconstruction, Comput. Math. Methods Med. (2016) 2016.
[78] Tanja Teuber, Gabriele Steidl, Raymond Honfu Chan, Minimization and parameter estimation for seminorm regularization models with I-divergence constraints, Inverse Probl. 29 (3) (2013) 035007.
[79] Robert Tibshirani, Regression shrinkage and selection via the Lasso, J. R. Stat. Soc., Ser. B, Methodol. 58 (1) (1996) 267–288.
[80] Ryan J. Tibshirani, Jonathan Taylor, The solution path of the generalized Lasso, Ann. Stat. 39 (3) (2011) 1335–1371.
[81] Ryan J. Tibshirani, Jonathan Taylor, Degrees of freedom in Lasso problems, Ann. Stat. 40 (2) (2012) 1198–1232.
[82] Curt Vogel, Computational Methods for Inverse Problems, Society for Industrial and Applied Mathematics, Philadelphia, 2002.
[83] Stephen J. Wright, Coordinate descent algorithms, Math. Program. 151 (1) (2015) 3–34.
[84] Xiaoming Yuan, Shangzhi Zeng, Jin Zhang, Discerning the linear convergence of admm for structured convex optimization through the lens of variational analysis, J. Mach. Learn. Res. 21 (83) (2020) 1–75.
[85] Hui Zou, Trevor Hastie, Robert Tibshirani, On the "degrees of freedom", the Lasso, Ann. Stat. 35 (5) (2007) 2173–2192.