Electronic Transactions on Numerical Analysis.
Volume 61, pp. 66–91, 2024.

ISSN 1068–9613.
DOI: 10.1553/etna_vol61s66

ETNA
Kent State University and
Johann Radon Institute (RICAM)

SOFTWARE FOR LIMITED MEMORY RESTARTED ℓ^p-ℓ^q MINIMIZATION METHODS USING GENERALIZED KRYLOV SUBSPACES*

ALESSANDRO BUCCINI† AND LOTHAR REICHEL‡

Dedicated to our friend Giuseppe Rodriguez on the occasion of his 60th birthday.

Abstract. This paper describes software for the solution of finite-dimensional minimization problems with two terms, a fidelity term and a regularization term. The sum of the p-norm of the former and the q-norm of the latter is minimized, where $0 < p, q \leq 2$. We note that the "p-norm" is not a norm when $0 < p < 1$, and similarly for the "q-norm". This kind of minimization problems arises when solving linear discrete ill-posed problems, such as certain problems in image restoration. They also find applications in statistics. Recently, limited-memory restarted numerical methods that are well suited for the solution of large-scale minimization problems of this kind were described by the authors in [Adv. Comput. Math., 49 (2023), Art. 26]. These methods are based on the application of restarted generalized Krylov subspaces. This paper presents software for these solution methods.

Key words. ℓ^p-ℓ^q minimization, inverse problem, regression, iterative method

AMS subject classifications. 65F10, 65R32, 90C26

1. Introduction.

Let $A \in \mathbb{R}^{m\times n}$ be a large ill-conditioned matrix, whose singular values decay to zero with increasing index number with no significant gap between the small singular values, and let the vector $\mathbf{b}^\delta \in \mathbb{R}^m$ represent available data. The data is assumed to be contaminated by an error $\boldsymbol{\eta} \in \mathbb{R}^m$ that may, for instance, be caused by measurement inaccuracies. Denote the unknown error-free vector associated with $\mathbf{b}^\delta$ by $\mathbf{b} \in \mathbb{R}^m$. We would like to determine the solution $\mathbf{x}^\dagger$ of minimal Euclidean norm of the linear system of equations

$$A\mathbf{x} = \mathbf{b}, \tag{1.1}$$

which is assumed to be consistent. Since the right-hand side $\mathbf{b}$ is not available, it may be tempting to instead solve

$$A\mathbf{x} = \mathbf{b}^\delta. \tag{1.2}$$

However, this system might not have a solution since A may be rank-deficient. Moreover, even if it has a solution, then this solution typically is a useless approximation of $\mathbf{x}^\dagger$ due to the error $\boldsymbol{\eta}$ in $\mathbf{b}^\delta$ and the ill-conditioning of A.

A possible approach to circumvent this difficulty is to compute an approximation of $\mathbf{x}^\dagger$ by solving an ℓ^p-ℓ^q minimization problem

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \left\{ \frac{1}{p} \left\| A\mathbf{x} - \mathbf{b}^\delta \right\|_p^p + \frac{\mu}{q} \left\| L\mathbf{x} \right\|_q^q \right\}, \tag{1.3}$$

where $0 < p, q \leq 2$ and $\|\mathbf{x}\|_p^p = \sum_{j=1}^n |x_j|^p$. With a slight abuse of notation, we will refer to $\|\mathbf{x}\|_p$ as the p-norm of $\mathbf{x}$ for any $p > 0$, even though it is not a norm for $p < 1$. The first term in the right-hand side of (1.3) is referred to as the *fidelity term* and the second term as the *regularization term*. The *regularization parameter* $\mu > 0$ balances the influence of these two

*Received March 5, 2024. Accepted June 2, 2024. Published online on July 3, 2024. Recommended by G. Rodriguez.

†Department of Mathematics and Computer Science, University of Cagliari, 09124 Cagliari, Italy (alessandro.buccini@unica.it).

‡Department of Mathematical Sciences, Kent State University, Kent, OH 44242, USA (reichel@math.kent.edu).

ETNA
Kent State University and
Johann Radon Institute (RICAM)

terms on the solution $\mathbf{x}^*$ of (1.3). The matrix $L \in \mathbb{R}^{s\times n}$ often is referred to as a *regularization matrix*. We will assume that

$$\mathcal{N}(A) \cap \mathcal{N}(L) = \{\mathbf{0}\}, \tag{1.4}$$

where $\mathcal{N}(M)$ denotes the null space of the matrix M. The matrix L in (1.3) typically easily can be chosen so that condition (1.4) is satisfied. Violation of this condition implies that the solution of (1.3) is not unique.

When $p = q = 2$, problem (1.3) is a Tikhonov regularization problem in general form, and efficient solution methods are available; see, e.g., [3,6,8,22,27]. We will not dwell on this situation further. Our main interest is in minimization problems (1.3) with $0 < \min\{p, q\} \leq 1$. The problem (1.3) is convex when $\min\{p, q\} \geq 1$ and non-smooth if either $p = 1$ or $q = 1$. If $\min\{p, q\} < 1$, then the problem is non-convex and non-smooth. The well-known fast iterative shrinkage-thresholding algorithm (FISTA) (see [2]) can be applied to solve (1.3) when $p = 2$ and $q = 1$; a comparison of FISTA and restarted generalized Krylov subspace methods is presented in [14]. Applications of the minimization problems (1.3) include image restoration [20,23], as well as the solution of other large-scale linear discrete ill-posed problems. Also large-scale regression problems in statistics with non-convex loss and penalty [4,26,35], as well as saturated and supersaturated design [5], require the solution of problems of the form (1.3).

The choices of the parameters p and q in (1.3) are important. A Bayesian derivation of the model (1.3) was given in [4]. Here, we provide a brief intuitive discussion on how to choose these parameters, starting with the parameter p. Let $\mathbf{b}$ denote the unknown noise-free data vector associated with the noise-contaminated vector $\mathbf{b}^\delta$, i.e., $\mathbf{b}^\delta = \mathbf{b} + \boldsymbol{\eta}$, where the vector $\boldsymbol{\eta}$ represents the noise. Assume for the moment that $\mathbf{b} \in \mathrm{range}(A)$. Then, the solution of minimal norm, $\mathbf{x}^\dagger$, of (1.1) satisfies $A\mathbf{x}^\dagger = \mathbf{b}$. This suggests that if $\boldsymbol{\eta}$ represents white Gaussian noise, then $p = 2$ is an appropriate choice. On the other hand, when $\boldsymbol{\eta}$ contains impulse noise (defined below), one generally would like to choose a "p-norm" that weighs outliers less than the Euclidean norm. One therefore typically chooses $0 < p \leq 1$ in this situation.

We say that the data vector $\mathbf{b}^\delta = [(\mathbf{b}^\delta)_1, (\mathbf{b}^\delta)_2, \ldots, (\mathbf{b}^\delta)_m]^T$ is corrupted by impulse noise if

$$(\mathbf{b}^\delta)_j = \begin{cases} r_j & \text{with probability } \sigma, \\ (\mathbf{b})_j & \text{with probability } 1-\sigma, \end{cases} \qquad j = 1, 2, \ldots, m,$$

where $0 < \sigma < 1$ and r_j is the realization of a random variable with uniform distribution in the dynamic range of the entries of $\mathbf{b}$. In the presence of impulse noise in $\mathbf{b}^\delta$, the choice $p = 0$ in (1.3) would appear to be appropriate, where the 0-norm of a vector counts the number of nonvanishing entries of the vector. However, the solution of minimization problems (1.3) with $p = 0$ is NP-hard and, therefore, not attractive to use in computations. It is popular to use the 1-norm instead of the 0-norm, because it secures convexity; see, e.g., [2]. However, p-norms with $0 < p < 1$ provide better approximations of the 0-norm than the 1-norm; see, e.g., [13] for an illustration. A Bayesian justification for choosing $0 < p < 1$ is presented in [4].

The value of $q > 0$ can be determined by using a-priori information about the sparsity of the desired solution $\mathbf{x}^*$ of (1.3); typically, the smaller the value q is, the fewer nonvanishing entries in the solution $\mathbf{x}^*$ of (1.3) are. Liu and Barber [26] provide a justification for using $q < 1$ and propose the choice $q = 2/3$. In imaging problems, the restored image usually has a sparse representation when expressed in terms of wavelet or framelet bases, i.e., many of the coefficients in representations of an image in these bases vanish. In this situation, letting

ETNA
Kent State University and
Johann Radon Institute (RICAM)

$0 < q < 1$ usually provides satisfactory restorations; see, e.g., [11] for illustrations. Moreover, the discrete total variation operator applied to the desired solution often is sparse in this kind of application and this justifies letting $0 < q < 1$; see, e.g., [23] for illustrations. Recently, Lanza et al. [24, 25] described minimization methods for use in image restoration that adaptively determine p and q locally. These methods are very interesting, but they are more complicated than the methods considered in the present paper.

Finally, the regularization parameter μ in (1.3) has to be chosen. Several approaches have been described in the literature. They are based on the discrepancy principle, cross validation, generalized cross validation, as well as on other techniques; see [10–13, 29, 31] and references therein for discussions and illustrations. The choice of technique depends on whether $\mathbf{b}^\delta$ is contaminated by impulse noise, and when it is not, whether an estimate for the norm of $\boldsymbol{\eta}$ is available. The determination of a regularization matrix L is discussed in [7].

It is the purpose of this paper to describe and present software for the solution of the minimization problem (1.3). The iterative majorization-minimization (MM) methods described by Huang et al. [20] and Lanza et al. [23] determine an approximate solution in a generalized Krylov subspace. The dimension of the subspace is increased by one with each iteration until a satisfactory approximate solution of (1.3) has been determined. However, when the number of iterations is large, the memory requirement for large-scale problems may be prohibitively large. Moreover, the computational effort required to determine an orthogonal basis for the solution subspace may be substantial. This prompted the development of a restarted method, which allows the use of a solution subspace of fairly small dimension, say ≤ 30, also when many iterations are required to solve (1.3) to desired accuracy; see [14]. Restarting is achieved by reinitializing the solution subspace periodically, similarly as in the restarted GMRES method; see [33] for a discussion of the latter method. Computed examples in [14] show the restarted method to determine an approximate solution of desired accuracy of (1.3) faster than when restarting is not employed. This paper outlines the restarted method and describes its implementation in MATLAB. A Python version of part of this software is available in the toolbox `TRIPs-Py` [30]. Note that the Python implementation does not implement the restarted strategy and some of the rules for choosing the regularization parameter that we describe in this paper.

The organization of this paper is as follows: Section 2 briefly reviews the restarted solution method for (1.3) and describes two algorithms. Several rules for choosing the regularization parameter are described in Section 3. A discussion of the software that accompanies this paper[1] is provided in Section 4. Section 5 illustrates the performances of our software and compares our approach to the one described in [35]. Concluding remarks can be found in Section 6.

2. Majorization-minimization methods for large-scale problems. We briefly review two iterative methods described in [20] for the solution of minimization problems of the form (1.3). Each iteration with these methods consists of a majorization step and a minimization step. The majorization step determines a quadratic functional that majorizes a smoothed functional $\mathcal{J}_\varepsilon$ related to the problem (1.3). The majorant and its gradient agree with those of the smoothed functional at the current approximation $\mathbf{x}^{(k)}$ of the sought minimum. The minimization step minimizes the majorant; its unique minimizer is the new approximation, $\mathbf{x}^{(k+1)}$, of the solution $\mathbf{x}^*$. Two approaches to determine quadratic majorants are described in [20]. We will outline both. The discussion of this section follows [14, Sections 2–3] as well as [20, Sections 2–5] quite closely.

[1] https://etna.ricam.oeaw.ac.at/volumes/2021-2030/vol61/addition/p66.php

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Majorization step. Introduce the functional

$$\mathcal{J}(\mathbf{x}) = \frac{1}{p}\left\|A\mathbf{x}-\mathbf{b}^{\delta}\right\|_p^p + \frac{\mu}{q}\left\|L\mathbf{x}\right\|_q^q \tag{2.1}$$

associated with the minimization problem (1.3). We are primarily interested in the situation when $0 < \min\{p,q\} < 1$. Then, the functional (2.1) is neither convex nor differentiable. The construction of the quadratic majorants mentioned above requires the functional to be continuously differentiable. We therefore introduce the smoothed functional

$$\mathcal{J}_\varepsilon(\mathbf{x}) = \frac{1}{p}\sum_{j=1}^{m}\Phi_{p,\varepsilon}\left((A\mathbf{x}-\mathbf{b}^{\delta})_j\right) + \frac{\mu}{q}\sum_{j=1}^{s}\Phi_{q,\varepsilon}\left((L\mathbf{x})_j\right)$$

for some small parameter $\varepsilon > 0$, where

$$\Phi_{s,\varepsilon}(t) = \begin{cases} |t|^s & \text{for } s > 1, \\ \left(t^2+\varepsilon^2\right)^{s/2} & \text{for } 0 < s \le 1, \end{cases} \tag{2.2}$$

is a differentiable function of t. We note that $\mathcal{J}_\varepsilon(\mathbf{x})$ is everywhere differentiable and seek to compute a solution $\mathbf{x}_\varepsilon^*$ of the smoothed problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} \mathcal{J}_\varepsilon(\mathbf{x}). \tag{2.3}$$

When $\min\{p,q\} > 1$, the functional $\mathcal{J}_\varepsilon(\mathbf{x})$ is strictly convex and therefore has a unique minimum. However, when $0 < \min\{p,q\} < 1$, the functional (2.3) is not convex. Then, the methods described seek to determine a local minimum or stationary point of $\mathcal{J}_\varepsilon$.

Let $\mathbf{x}^{(k)}$ be an available approximate solution of the problem (2.3). The majorant referred to as "adaptive" in [20] is determined by constructing a quadratic approximation of each component of $\mathcal{J}_\varepsilon(\mathbf{x})$ at $\mathbf{x}^{(k)}$. The approximation is furnished by the adaptive quadratic tangent majorant $\mathcal{Q}^A(\mathbf{x},\mathbf{x}^{(k)})$ of $\mathcal{J}_\varepsilon(\mathbf{x})$ at $\mathbf{x}^{(k)}$. It satisfies

$$\begin{cases} \mathcal{Q}^A(\mathbf{x}^{(k)},\mathbf{x}^{(k)}) = \mathcal{J}_\varepsilon(\mathbf{x}^{(k)}), \\ \nabla_{\mathbf{x}}\mathcal{Q}^A(\mathbf{x}^{(k)},\mathbf{x}^{(k)}) = \nabla_{\mathbf{x}}\mathcal{J}_\varepsilon(\mathbf{x}^{(k)}), \\ \mathcal{Q}^A(\mathbf{x},\mathbf{x}^{(k)}) \ge \mathcal{J}_\varepsilon(\mathbf{x}) \ \forall \mathbf{x}\in\mathbb{R}^n, \\ \mathbf{x} \to \mathcal{Q}^A(\mathbf{x},\mathbf{x}^{(k)}) \quad \text{is a quadratic functional;} \end{cases} \tag{2.4}$$

see [20] for details. Here, $\nabla_{\mathbf{x}}$ denotes the gradient with respect to $\mathbf{x}$. Typically, each component of $\mathcal{J}_\varepsilon(\mathbf{x})$ at $\mathbf{x}^{(k)}$ is approximated by a different quadratic polynomial. The majorant is constructed by evaluating the residual vectors

$$\mathbf{v}^{(k)} = A\mathbf{x}^{(k)} - \mathbf{b}^{\delta}, \qquad \mathbf{u}^{(k)} = L\mathbf{x}^{(k)},$$

which define the *weight vectors*

$$\boldsymbol{\omega}_{\text{fid}}^{A,(k)} = \left(\left(\mathbf{v}^{(k)}\right)^2 + \varepsilon^2\mathbf{1}\right)^{p/2-1}, \qquad \boldsymbol{\omega}_{\text{reg}}^{A,(k)} = \left(\left(\mathbf{u}^{(k)}\right)^2 + \varepsilon^2\mathbf{1}\right)^{q/2-1},$$

where $\mathbf{1} = [1,1,\ldots,1]^T$, the superscript ^{T} denotes transposition, and all operations are element-wise. The weight vectors determine the diagonal matrices

$$W_{\text{fid}}^{(k)} = \operatorname{diag}\left(\boldsymbol{\omega}_{\text{fid}}^{A,(k)}\right) \quad \text{and} \quad W_{\text{reg}}^{(k)} = \operatorname{diag}\left(\boldsymbol{\omega}_{\text{reg}}^{A,(k)}\right),$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

which are used in the construction of the adaptive quadratic tangent majorant of $\mathcal{J}_\varepsilon$ at $\mathbf{x}^{(k)}$,

$$\mathcal{Q}^A(\mathbf{x},\mathbf{x}^{(k)}) = \frac{1}{2}\left\| \left(W_{\mathrm{fid}}^{(k)}\right)^{1/2} \left(A\mathbf{x}-\mathbf{b}^\delta\right)\right\|_2^2 + \frac{\mu}{2}\left\| \left(W_{\mathrm{reg}}^{(k)}\right)^{1/2} L\mathbf{x}\right\|_2^2 + c,$$

where $c \in \mathbb{R}$ is a constant that is independent of $\mathbf{x}$. Let $\mathbf{x}^{(k+1)}$ denote the minimizer of $\mathcal{Q}^A(\mathbf{x},\mathbf{x}^{(k)})$. It is the next approximate solution of (2.3). We discuss the computation of an approximation of $\mathbf{x}^{(k+1)}$ below. This approximation also will be denoted by $\mathbf{x}^{(k+1)}$.

We turn to the construction of the majorant that in [20] is referred to as "fixed". This majorant is determined by computing quadratic polynomial majorants for each component of $\mathcal{J}_\varepsilon(\mathbf{x})$ at $\mathbf{x}^{(k)}$ with the leading coefficients of all quadratic polynomials chosen to be the same. This results in a simplification of the computations, when compared to using the adaptive majorant. However, the minimization method obtained with fixed majorants typically requires more iteration steps than the method based on adaptive majorants to satisfy the stopping criterion. Overall, in our experience, the fixed approach requires less computational work, even though more iterations are performed.

The *weight vectors* for the fixed majorant are given by

$$\begin{aligned}
\boldsymbol{\omega}_{\mathrm{fid}}^{F,(k)} &= \mathbf{v}^{(k)}\left(\mathbf{1}-\left(\frac{\left(\mathbf{v}^{(k)}\right)^2+\varepsilon^2\mathbf{1}}{\varepsilon^2}\right)^{p/2-1}\right),\\
\boldsymbol{\omega}_{\mathrm{reg}}^{F,(k)} &= \mathbf{u}^{(k)}\left(\mathbf{1}-\left(\frac{\left(\mathbf{u}^{(k)}\right)^2+\varepsilon^2\mathbf{1}}{\varepsilon^2}\right)^{q/2-1}\right),
\end{aligned} \tag{2.5}$$

where all operations are element-wise. We obtain the fixed quadratic tangent majorant

$$\begin{aligned}
\mathcal{Q}^F(\mathbf{x},\mathbf{x}^{(k)}) = \frac{1}{2}&\left(\left\|A\mathbf{x}-\mathbf{b}^\delta\right\|_2^2 - 2\left\langle \boldsymbol{\omega}_{\mathrm{fid}}^{F,(k)}, A\mathbf{x}\right\rangle\right)\\
&+\frac{\mu}{2}\varepsilon^{q-p}\left(\|L\mathbf{x}\|_2^2 - 2\left\langle \boldsymbol{\omega}_{\mathrm{reg}}^{F,(k)}, L\mathbf{x}\right\rangle\right) + c,
\end{aligned}$$

of $\mathcal{J}_\varepsilon$ at $\mathbf{x}^{(k)}$. Here, $\langle\cdot,\cdot\rangle$ denotes the standard inner product and the constant $c \in \mathbb{R}$ is independent of $\mathbf{x}$. The functional $\mathcal{Q}^F(\mathbf{x},\mathbf{x}^{(k)})$ satisfies the properties (2.4) with $\mathcal{Q}^A(\mathbf{x},\mathbf{x}^{(k)})$ replaced by $\mathcal{Q}^F(\mathbf{x},\mathbf{x}^{(k)})$; see [20] for details.

Minimization step. We now consider the minimization of $\mathcal{Q}^A$ and $\mathcal{Q}^F$ when the matrices $A \in \mathbb{R}^{m\times n}$ and $L \in \mathbb{R}^{s\times n}$ are large. To reduce the computational effort, we seek to determine approximate solutions in solution subspaces $\mathcal{V}_{\widehat{k}}$ of fairly small dimension $\widehat{k}$. Let k be the number of iterations. The dimension $\widehat{k}$ usually increases with k and, therefore, $\widehat{k} \geq k$. However, this is not be the case when a restarting technique is used; see below. In particular, $\widehat{k} \ll \min\{m,n,s\}$. Let the columns of the matrix $V_{\widehat{k}} \in \mathbb{R}^{n\times\widehat{k}}$ form an orthonormal basis for $\mathcal{V}_{\widehat{k}}$. We determine approximations of the minima of $\mathcal{Q}^A$ and $\mathcal{Q}^F$ of the form

$$\mathbf{x}^{(k+1)} = V_{\widehat{k}}\mathbf{y}^{(k+1)}, \tag{2.6}$$

where $\mathbf{y}^{(k+1)} \in \mathbb{R}^{\widehat{k}}$.

When using adaptive majorants, we solve

$$\min_{\mathbf{x}\in\mathcal{V}_{\widehat{k}}} \frac{1}{2}\left\| \left(W_{\mathrm{fid}}^{(k)}\right)^{1/2} \left(A\mathbf{x}-\mathbf{b}^\delta\right)\right\|_2^2 + \frac{\mu}{2}\left\| \left(W_{\mathrm{reg}}^{(k)}\right)^{1/2} L\mathbf{x}\right\|_2^2 \tag{2.7}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

for $\mathbf{x}^{(k+1)}$. This is equivalent to computing the solution $\mathbf{y}^{(k+1)}$ of

$$\min_{\mathbf{y}\in\mathbb{R}^{\widehat{k}}} \frac{1}{2}\left\| \left(W_{\text{fid}}^{(k)}\right)^{1/2} (AV_{\widehat{k}}\mathbf{y} - \mathbf{b}^{\delta}) \right\|_2^2 + \frac{\mu}{2}\left\| \left(W_{\text{reg}}^{(k)}\right)^{1/2} LV_{\widehat{k}}\mathbf{y} \right\|_2^2. \tag{2.8}$$

Introduce the economic QR factorizations

$$\begin{aligned} \left(W_{\text{fid}}^{(k)}\right)^{1/2} AV_{\widehat{k}} &= Q_A R_A, \qquad Q_A \in \mathbb{R}^{m\times\widehat{k}},\ \ R_A \in \mathbb{R}^{\widehat{k}\times\widehat{k}}, \\ \left(W_{\text{reg}}^{(k)}\right)^{1/2} LV_{\widehat{k}} &= Q_L R_L, \qquad Q_L \in \mathbb{R}^{s\times\widehat{k}},\ \ R_L \in \mathbb{R}^{\widehat{k}\times\widehat{k}}, \end{aligned} \tag{2.9}$$

and compute

$$\mathbf{y}^{(k+1)} = \arg\min_{\mathbf{y}\in\mathbb{R}^{\widehat{k}}} \frac{1}{2}\left\| R_A\mathbf{y} - Q_A^T \left(W_{\text{fid}}^{(k)}\right)^{1/2} \mathbf{b}^{\delta} \right\|_2^2 + \frac{\mu}{2}\left\| R_L\mathbf{y} \right\|_2^2.$$

Assume that

$$\mathcal{N}\left(\left(W_{\text{fid}}^{(k)}\right)^{1/2} AV_{\widehat{k}}\right) \cap \mathcal{N}\left(\left(W_{\text{fid}}^{(k)}\right)^{1/2} LV_{\widehat{k}}\right) = \{\mathbf{0}\}.$$

This is typically true in applications of interest to us. Then, the solution $\mathbf{y}^{(k+1)}$ is unique, and the approximate minimizer of $\mathcal{Q}^A(\mathbf{x}, \mathbf{x}^{(k)})$ is given by (2.6). The computation of $\mathbf{y}^{(k+1)}$ can be performed cheaply with direct methods. In our numerical implementation we use the `backslash` command in Matlab that performs a QR factorization of $\begin{bmatrix} R_A \\ \sqrt{\mu}R_L \end{bmatrix}$.

Note that the matrix Q_L is not used in the algorithm, and, therefore, one may employ a Q-less factorization to reduce the computational cost and memory requirement. However, for simplicity, we do not explore this possibility here.

To proceed, we enlarge the solution subspace $\mathcal{V}_k$ by including the normalized residual of the normal equations associated with (2.7). Thus, let

$$\mathbf{r}^{(k+1)} = A^T W_{\text{fid}}^{(k)} \left(A\mathbf{x}^{(k+1)} - \mathbf{b}^{\delta}\right) + \mu L^T W_{\text{reg}}^{(k)} L\mathbf{x}^{(k+1)}.$$

The columns of the matrix

$$V_{\widehat{k}+1} = \left[V_{\widehat{k}}, \mathbf{r}^{(k+1)}/\|\mathbf{r}^{(k+1)}\|_2\right]$$

furnish an orthonormal basis for the new solution subspace $\mathcal{V}_{\widehat{k}+1}$. Note that the vector $\mathbf{r}^{(k+1)}$ is proportional to the gradient of $\mathcal{Q}^A(\mathbf{x}, \mathbf{x}^{(k)})$ restricted to $\mathcal{V}_{\widehat{k}}$ at $\mathbf{x} = \mathbf{x}^{(k+1)}$. We refer to the solution subspace $\mathcal{V}_{\widehat{k}+1} = \text{range}(V_{\widehat{k}+1})$ as a *generalized Krylov subspace*. This way to determine a solution subspace also has been applied to solve Tikhonov regularization problems in general form in [22]. Note that the evaluation of $\mathbf{r}^{(k+1)}$ requires only one matrix-vector product with each one of the matrices A^T and L^T, since one can use the QR factorizations (2.9) and the relation (2.6) to avoid forming matrix-vector products with the matrices A and L. Moreover, we store and update the "skinny" matrices $AV_{\widehat{k}}$ and $LV_{\widehat{k}}$ at each iteration to reduce the computational cost. In our software, we let $\mathcal{V}_0 = \text{span}\left\{\mathbf{x}^{(0)}\right\}$. We may, for instance, let $\mathbf{x}_0 = A^T\mathbf{b}^{\delta}$. Then, $\widehat{k} = k+1$.

Summarizing, each iteration of the adaptive approach requires one matrix-vector product evaluation with each one of the matrices A, L, A^T, and L^T, as well as the computation of

ETNA
Kent State University and
Johann Radon Institute (RICAM)

economic QR factorizations of two tall and skinny matrices, whose column numbers increase by one with each iteration. The latter computations can be quite demanding if the matrices A and L are large and many iterations are required. The algorithm stores the three matrices $V_{\widehat{k}}$, $AV_{\widehat{k}}$, and $LV_{\widehat{k}}$. To keep the storage requirement bounded independently of the number of iterations, our software for the solution of (1.3) reinitializes the solution subspace as well as the matrices $V_{\widehat{k}}$, $AV_{\widehat{k}}$, and $LV_{\widehat{k}}$ periodically. This is implemented in the algorithms described below.

We turn to the fixed approach. The weight vectors are now given by (2.5), and we would like to solve the minimization problem

$$\min_{\mathbf{x}\in\mathcal{V}_{\widehat{k}}} \frac{1}{2}\left(\left\|A\mathbf{x}-\mathbf{b}^{\delta}\right\|_2^2 - 2\left\langle \boldsymbol{\omega}_{\mathrm{fid}}^{F,(k)}, A\mathbf{x}\right\rangle\right) + \frac{\eta}{2}\left(\|L\mathbf{x}\|_2^2 - 2\left\langle \boldsymbol{\omega}_{\mathrm{reg}}^{F,(k)}, L\mathbf{x}\right\rangle\right) \tag{2.10}$$

for $\mathbf{x}^{(k+1)}$, where $\eta = \mu\varepsilon^{q-p}$. This problem can be expressed as

$$\min_{\mathbf{y}\in\mathbb{R}^{\widehat{k}}} \left\|AV_{\widehat{k}}\mathbf{y}-\mathbf{b}^{\delta}-\boldsymbol{\omega}_{\mathrm{fid}}^{F,(k)}\right\|_2^2 + \eta\left\|LV_{\widehat{k}}\mathbf{y}-\boldsymbol{\omega}_{\mathrm{reg}}^{F,(k)}\right\|_2^2. \tag{2.11}$$

The solution $\mathbf{y}^{(k+1)}$ of (2.11) yields the solution $\mathbf{x}^{(k+1)} = V_{\widehat{k}}\mathbf{y}^{(k+1)}$ of (2.10). Introduce the economic QR factorizations

$$\begin{aligned} AV_{\widehat{k}} &= Q_A R_A, \qquad Q_A \in \mathbb{R}^{m\times\widehat{k}}, \quad R_A \in \mathbb{R}^{\widehat{k}\times\widehat{k}}, \\ LV_{\widehat{k}} &= Q_L R_L, \qquad Q_L \in \mathbb{R}^{s\times\widehat{k}}, \quad R_L \in \mathbb{R}^{\widehat{k}\times\widehat{k}}. \end{aligned}$$

Substituting these factorizations into (2.11) gives

$$\mathbf{y}^{(k+1)} = \arg\min_{\mathbf{y}\in\mathbb{R}^{\widehat{k}}} \left\| \begin{bmatrix} R_A \\ \sqrt{\eta}R_L \end{bmatrix}\mathbf{y} - \begin{bmatrix} Q_A^T\left(\mathbf{b}^{\delta}+\boldsymbol{\omega}_{\mathrm{fid}}^{F,(k)}\right) \\ \sqrt{\eta}Q_L^T\boldsymbol{\omega}_{\mathrm{reg}}^{F,(k)} \end{bmatrix}\right\|_2^2.$$

Similarly to the adaptive case, we compute $\mathbf{y}^{(k+1)}$ with the `backslash` command, i.e., we perform a QR factorization of $\begin{bmatrix} R_A \\ \sqrt{\eta}R_L \end{bmatrix}$. Having computed $\mathbf{y}^{(k+1)}$ and $\mathbf{x}^{(k+1)}$, we enlarge the solution subspace by including the residual

$$\mathbf{r}^{(k+1)} = A^T\left(A\mathbf{x}^{(k+1)} - \left(\mathbf{b}^{\delta}+\boldsymbol{\omega}_{\mathrm{fid}}^{F,(k)}\right)\right) + \eta L^T\left(L\mathbf{x}^{(k+1)} - \boldsymbol{\omega}_{\mathrm{reg}}^{F,(k)}\right)$$

of the normal equations associated with (2.10). Thus, we let $\mathbf{v}_{\mathrm{new}} = \mathbf{r}^{(k+1)}/\left\|\mathbf{r}^{(k+1)}\right\|_2$. Then, the columns of the matrix $V_{\widehat{k}+1} = [V_{\widehat{k}}, \mathbf{v}_{\mathrm{new}}]$ form an orthonormal basis for the solution subspace $\mathcal{V}_{\widehat{k}+1}$. Note that the residual is proportional to the gradient of $\mathcal{Q}^F(\mathbf{x}, \mathbf{x}^{(k)})$ restricted to $\mathcal{V}_{\widehat{k}}$ at $\mathbf{x} = \mathbf{x}^{(k+1)}$.

Differently from (2.8), the least-squares problem (2.11) does not have a diagonal scaling matrix that is updated at each iteration. We therefore can compute the QR factorizations of $AV_{\widehat{k}+1}$ and $LV_{\widehat{k}+1}$ by updating the QR factorizations of $AV_{\widehat{k}}$ and $LV_{\widehat{k}}$, respectively. This reduces the computational work and makes each iteration with the fixed approach cheaper than each corresponding iteration with the adaptive approach. Updating formulas for the QR factorization can be found in [15, 20]. We remark that the fixed approach might require more iterations than the adaptive approach to satisfy the stopping criterion, but still be faster. For certain "difficult" minimization problems that arise in nonlinear regression, we found the adaptive approach to perform better; see [4]. The choice of approach should depend on the problem to be solved.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Similarly as when using adaptive majorants, each iteration with fixed majorants requires one matrix-vector product evaluation with each one of the matrices A, L, A^T, and L^T. The memory requirements of the fixed and adaptive approaches are essentially the same and grows linearly with the number of iterations. Therefore, when the matrix A is large, the memory requirement may be substantial when many iterations are required to satisfy the stopping criterion. Moreover, the arithmetic cost of computing QR factorizations in the adaptive approach and for updating QR factorizations in the fixed approach grows quadratically and linearly, respectively, with the number of iterations. We described in [14] how to reduce the memory requirement and the average computational effort per iteration by reinitializing the solution subspace periodically. We will outline this below. Computed examples reported in [14] indicate that reinitializing the solution subspace periodically does not reduce the quality of the computed solution.

Let $K_{\max}$ denote the maximal permitted dimension of the solution subspaces and assume that for a certain positive integer $\widehat{k}$, we have $V_{\widehat{k}} \in \mathbb{R}^{n\times K_{\max}}$. We would like to avoid the dimension of the solution subspace to increase further and therefore restart the MM algorithms by letting $\widetilde{\mathbf{x}} = \mathbf{x}^{(k)}$ and $\mathbf{x}^{(k+1)} = \widetilde{V}_{\widetilde{k}+1}\mathbf{y}^{(k+1)}$ with $\widetilde{V}_{\widetilde{k}+1} = \widetilde{\mathbf{x}}/\left\|\widetilde{\mathbf{x}}\right\|_2$. This is implemented by the adaptive and fixed MM algorithms of this paper; see Algorithms 1 and 2, respectively. Convergence properties of the iterates determined by Algorithms 1 and 2 are shown in [14].

3. Choice of the regularization parameter. This section briefly reviews the regularization parameter choice rules that are implemented in the software package. We distinguish two kinds of rules: stationary and nonstationary ones. In the stationary rules the regularization parameter is fixed throughout the iterations. This, usually, implies that one has to run the selected algorithm for several choices of μ and then pick the "best" computed approximate solution according to some criteria. In the nonstationary rules, the regularization parameter is changed at each iteration. Usually only a single run of the method is required.

3.1. Stationary rules. We outline the two stationary rules that are implemented in our package, Cross Validation (CV) and Modified Cross Validation (MCV) described in [12]. Note that these rules are heuristic, i.e., they may fail to determine a suitable value of the regularization parameter for certain problems. Nevertheless, they perform well in general and therefore are of interest; they require very little knowledge about the properties of the noise. We refer to Kindermann [21] for discussions and analyses of some heuristic rules.

3.1.1. Cross validation. The CV method is based on the observation that a good approximate solution $\mathbf{x}^*$ should be able to predict missing data in $\mathbf{b}^\delta$ [34]. To determine a suitable regularization parameter, we therefore remove some entries of $\mathbf{b}^\delta$ and the corresponding rows of A, then solve the slightly smaller problem so-obtained using several regularization parameters $\{\mu_j\}_{j=1,\ldots,k}$, and pick the parameter for which the associated computed solution best predicts the entries of $\mathbf{b}^\delta$ that we removed. In detail, let the set $I \subset \{1,\ldots,m\}$ contain the indices of the entries of $\mathbf{b}^\delta$ that we remove and denote the cardinality of I by d. Let $\widetilde{\mathbf{b}^\delta} \in \mathbb{R}^{m-d}$ and $\widetilde{A} \in \mathbb{R}^{(m-d)\times n}$ be the vector and matrix obtained by removing the rows with indices in I. Fix a choice of k regularization parameters $\{\mu_j\}_{j=1,\ldots,k}$ and define

$$\mathbf{x}_j = \arg\min_{\mathbf{x}} \frac{1}{p}\left\|\widetilde{A}\mathbf{x} - \widetilde{\mathbf{b}^\delta}\right\|_p^p + \frac{\mu_j}{q}\left\|L\mathbf{x}\right\|_q^q.$$

In actual computations, we use the algorithms of Section 2 to compute approximations of the vectors $\mathbf{x}_j$. This holds for all minimization problems considered in this section.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Algorithm 1: A-MM-GKS-Restarted

Let $A \in \mathbb{R}^{m\times n}$, $\mathbf{b}^\delta \in \mathbb{R}^m$. Let $\mu > 0$ be a fixed parameter and $L \in \mathbb{R}^{s\times n}$ be such that $\mathcal{N}(A) \cap \mathcal{N}(L) = \{\mathbf{0}\}$. Fix $0 < p, q \leq 2$, a maximum number of iterations K, a maximum dimension $K_{\max}$, a smoothing parameter $\varepsilon > 0$, and a tolerance $\tau > 0$. Let $\mathbf{x}^{(0)}$ be an initial guess for $\mathbf{x}^\dagger$;

$V_0 = \mathbf{x}^{(0)} / \left\|\mathbf{x}^{(0)}\right\|_2$;

Compute and store $A_0 = AV_0$ and $L_0 = LV_0$;

$\mathbf{v}^{(0)} = A\mathbf{x}^{(0)} - \mathbf{b}^\delta$;

$\mathbf{u}^{(0)} = L\mathbf{x}^{(0)}$;

for $k = 0, 1, \ldots, K$ **do**

$\boldsymbol{\omega}_{\text{fid}}^{A,(k)} = \left(\left(\mathbf{v}^{(k)}\right)^2 + \varepsilon^2 \mathbf{1}\right)^{p/2-1}$;

$\boldsymbol{\omega}_{\text{reg}}^{A,(k)} = \left(\left(\mathbf{u}^{(k)}\right)^2 + \varepsilon^2 \mathbf{1}\right)^{q/2-1}$;

$W_{\text{fid}}^{(k)} = \operatorname{diag}\left(\boldsymbol{\omega}_{\text{fid}}^{A,(k)}\right)$;

$W_{\text{reg}}^{(k)} = \operatorname{diag}\left(\boldsymbol{\omega}_{\text{reg}}^{A,(k)}\right)$;

Compute the QR factorizations $\begin{cases} \left(W_{\text{fid}}^{(k)}\right)^{1/2} A_k = Q_A R_A \\ \left(W_{\text{reg}}^{(k)}\right)^{1/2} L_k = Q_L R_L \end{cases}$;

$\mathbf{y}^{(k+1)} = \arg\min_{\mathbf{y}} \frac{1}{2} \left\| R_A \mathbf{y} - Q_A^T \left(W_{\text{fid}}^{(k)}\right)^{1/2} \mathbf{b}^\delta \right\|_2^2 + \frac{\mu}{2} \left\| R_L \mathbf{y} \right\|_2^2$;

if $k > 1$ & $\left\|\mathbf{y}^{(k+1)} - \mathbf{y}^{(k)}\right\|_2 < \tau \left\|\mathbf{y}^{(k)}\right\|_2$ **then**

break;

end

if $k + 1 \equiv 0$ *mod* $K_{\max}$ **then**

$\widetilde{\mathbf{x}} = V_k \mathbf{y}^{(k+1)}$;

$\mathbf{v}^{(k+1)} = A_k \mathbf{y}^{(k+1)} - \mathbf{b}^\delta$;

$\mathbf{u}^{(k+1)} = L_k \mathbf{y}^{(k+1)}$;

Set $V_{k+1} = \widetilde{\mathbf{x}} / \left\|\widetilde{\mathbf{x}}\right\|_2$;

Compute and store $A_{k+1} = AV_{k+1}$;

Compute and store $L_{k+1} = LV_{k+1}$;

else

$\mathbf{r}^{(k+1)} = A^T W_{\text{fid}}^{(k)} \left(A_k \mathbf{y}^{(k+1)} - \mathbf{b}^\delta\right) + \mu L^T W_{\text{reg}}^{(k)} \left(L_k \mathbf{y}^{(k+1)}\right)$;

$\mathbf{v}_{\text{new}} = \mathbf{r}^{(k+1)} / \left\|\mathbf{r}^{(k+1)}\right\|_2$;

$V_{k+1} = [V_k, \mathbf{v}_{\text{new}}]$;

$A_{k+1} = [A_k, A\mathbf{v}_{\text{new}}]$;

$L_{k+1} = [L_k, L\mathbf{v}_{\text{new}}]$;

$\mathbf{v}^{(k+1)} = A_k \mathbf{y}^{(k+1)} - \mathbf{b}^\delta$;

$\mathbf{u}^{(k+1)} = L_k \mathbf{y}^{(k+1)}$;

end

end

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Algorithm 2: F-MM-GKS-Restarted

Let $A \in \mathbb{R}^{m\times n}$, $\mathbf{b}^\delta \in \mathbb{R}^m$. Let $\mu > 0$ be a fixed parameter and $L \in \mathbb{R}^{s\times n}$ be such that $\mathcal{N}(A) \cap \mathcal{N}(L) = \{\mathbf{0}\}$. Fix $0 < p, q \leq 2$, a maximum number of iterations K, a maximum dimension $K_{\max}$, a smoothing parameter $\varepsilon > 0$, and a tolerance $\tau > 0$. Let $\mathbf{x}^{(0)}$ be an initial guess for $\mathbf{x}^\dagger$;
$V_0 = \mathbf{x}^{(0)} / \left\|\mathbf{x}^{(0)}\right\|_2$;
Compute and store $A_0 = AV_0$ and $L_0 = LV_0$;
Compute the QR factorizations $\begin{cases} A_0 = Q_A R_A \\ L_0 = Q_L R_L \end{cases}$;
$\mathbf{v}^{(0)} = A\mathbf{x}^{(0)} - \mathbf{b}^\delta$;
$\mathbf{u}^{(0)} = L\mathbf{x}^{(0)}$;
$\eta = \mu\varepsilon^{q-p}$;
for $k = 0, 1, \ldots, K$ **do**
  $\boldsymbol{\omega}_{\text{fid}}^{F,(k)} = \mathbf{v}^{(k)} \left(\mathbf{1} - \left(\frac{(\mathbf{v}^{(k)})^2 + \varepsilon^2\mathbf{1}}{\varepsilon^2}\right)^{p/2-1}\right)$;
  $\boldsymbol{\omega}_{\text{reg}}^{F,(k)} = \mathbf{u}^{(k)} \left(\mathbf{1} - \left(\frac{(\mathbf{u}^{(k)})^2 + \varepsilon^2\mathbf{1}}{\varepsilon^2}\right)^{q/2-1}\right)$;
  $\mathbf{y}^{(k+1)} = \arg\min_{\mathbf{y}} \left\|A_k\mathbf{y} - \mathbf{b}^\delta - \boldsymbol{\omega}_{\text{fid}}^{F,(k)}\right\|_2^2 + \eta \left\|L_k\mathbf{y} - \boldsymbol{\omega}_{\text{reg}}^{F,(k)}\right\|_2^2$;
  if $k > 1$ & $\left\|\mathbf{y}^{(k+1)} - \mathbf{y}^{(k)}\right\|_2 < \tau \left\|\mathbf{y}^{(k)}\right\|_2$ **then**
    break;
  end
  if $k + 1 \equiv 0$ *mod* $K_{\max}$ **then**
    $\widetilde{\mathbf{x}} = V_k\mathbf{y}^{(k+1)}$;
    $\mathbf{v}^{(k+1)} = A_k\mathbf{y}^{(k+1)} - \mathbf{b}^\delta$;
    $\mathbf{u}^{(k+1)} = L_k\mathbf{y}^{(k+1)}$;
    Set $V_{k+1} = \widetilde{\mathbf{x}} / \left\|\widetilde{\mathbf{x}}\right\|_2$;
    Compute and store $A_{k+1} = AV_{k+1}$;
    Compute and store $L_{k+1} = LV_{k+1}$;
    Compute the QR factorizations $\begin{cases} A_{k+1} = Q_A R_A \\ L_{k+1} = Q_L R_L \end{cases}$;
  else
    $\mathbf{r}^{(k+1)} = A^T \left(A_k\mathbf{y}^{(k+1)} - \mathbf{b}^\delta - \boldsymbol{\omega}_{\text{fid}}^{F,(k)}\right) + \eta L^T \left(L_k\mathbf{y}^{(k+1)} - \boldsymbol{\omega}_{\text{reg}}^{F,(k)}\right)$;
    $\mathbf{v}_{\text{new}} = \mathbf{r}^{(k+1)} / \left\|\mathbf{r}^{(k+1)}\right\|_2$;
    $V_{k+1} = [V_k, \mathbf{v}_{\text{new}}]$;
    $A_{k+1} = [A_k, A\mathbf{v}_{\text{new}}]$;
    $L_{k+1} = [L_k, L\mathbf{v}_{\text{new}}]$;
    $\mathbf{v}^{(k+1)} = A_k\mathbf{y}^{(k+1)} - \mathbf{b}^\delta$;
    $\mathbf{u}^{(k+1)} = L_k\mathbf{y}^{(k+1)}$;
    Update Q_A, Q_L, R_A, and R_L as described in [15, 20];
  end
end

ETNA
Kent State University and
Johann Radon Institute (RICAM)

We measure how well the missing data is reconstructed by $\mathbf{x}_j$ as

$$r_j = \sqrt{\sum_{i \in I} (A\mathbf{x}_j - \mathbf{b}^\delta)_i^2}$$

and choose the regularization parameter μ^* as

$$\mu^* = \mu_{j^*}, \quad \text{with} \quad j^* = \arg\min_j r_j.$$

To reduce statistical variability, we pick s different randomly chosen sets of indexes $I^{(l)}$. Each set provides a regularization parameter μ_l^*, and the final regularization parameter is determined as

$$\mu_{\mathrm{CV}} = \frac{1}{s} \sum_{l=1}^{s} \mu_l^*.$$

The final solution is obtained as

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \frac{1}{p} \left\| A\mathbf{x} - \mathbf{b}^\delta \right\|_p^p + \frac{\mu_{\mathrm{CV}}}{q} \left\| L\mathbf{x} \right\|_q^q.$$

3.1.2. Modified cross validation. Even though in the CV method the choice of the 2-norm to measure r_j seems natural, this choice might not be ideal if the data $\mathbf{b}^\delta$ is contaminated by non-Gaussian noise. To avoid relying on the noisy data $\mathbf{b}^\delta$, a variation of the CV method was introduced in [12]. The intuition is that a "good" reconstruction $\mathbf{x}^*$ should be stable with respect to loss of data. Therefore, we select two sets of indices, I_1 and I_2, and remove the entries with indices in I_1 and I_2 from $\mathbf{b}^\delta$ to obtain the data vectors $\widetilde{\mathbf{b}^\delta}_1$ and $\widetilde{\mathbf{b}^\delta}_2$, respectively. Similarly, we remove from A the rows with indices in I_1 and I_2 to obtain the matrices $\widetilde{A}_1$ and $\widetilde{A}_2$, respectively. For a choice of parameters $\{\mu_j\}_{j=1,\ldots,k}$, we compute

$$\mathbf{x}_j^{(1)} = \arg\min_{\mathbf{x}} \frac{1}{p} \left\| \widetilde{A}_1\mathbf{x} - \widetilde{\mathbf{b}^\delta}_1 \right\|_p^p + \frac{\mu_j}{q} \left\| L\mathbf{x} \right\|_q^q,$$
$$\mathbf{x}_j^{(2)} = \arg\min_{\mathbf{x}} \frac{1}{p} \left\| \widetilde{A}_2\mathbf{x} - \widetilde{\mathbf{b}^\delta}_2 \right\|_p^p + \frac{\mu_j}{q} \left\| L\mathbf{x} \right\|_q^q,$$

and denote the difference in norm between the two approximate solutions by

$$\Delta_j = \left\| \mathbf{x}_j^{(1)} - \mathbf{x}_j^{(2)} \right\|_2.$$

We choose the regularization parameter that results in solutions that are least sensitive with respect to the loss of data, i.e., we pick $\mu^* = \mu_{j^*}$ with

$$j^* = \arg\min_j \Delta_j.$$

To reduce variability, we repeat this procedure for s different pairs of index sets $I_1^{(l)}$ and $I_2^{(l)}$ to obtain s regularization parameters μ_l^* with $l = 1, \ldots, s$, and we determine the "best" regularization parameter by

$$\mu_{\mathrm{MCV}} = \frac{1}{s} \sum_{l=1}^{s} \mu_l^*.$$

The final approximate solution is computed by

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \frac{1}{p} \left\| A\mathbf{x} - \mathbf{b}^\delta \right\|_p^p + \frac{\mu_{\mathrm{MCV}}}{q} \left\| L\mathbf{x} \right\|_q^q.$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

3.2. Nonstationary rules. We review the nonstationary rules for choosing the regularization parameter μ discussed in [10, 11, 13]. Similarly to the stationary case, the first two rules are heuristic, and, therefore, they may fail to produce a suitable value of the regularization parameter. However, in our experience, this is usually not the case. The third rule is based on the discrepancy principle (DP). The convergence and regularization properties of this rule has been shown in [11]; see below for more details.

3.2.1. Generalized cross validation. We describe the approaches proposed in [13]. Consider ℓ^2-ℓ^2 regularization

$$\mathbf{x}_\mu = \arg\min_{\mathbf{x}} \frac{1}{2} \left\| A\mathbf{x} - \mathbf{b}^\delta \right\|_2^2 + \frac{\mu}{2} \left\| L\mathbf{x} \right\|_2^2 . \tag{3.1}$$

For each $\mu > 0$, we define the function

$$G(\mu) = \frac{\left\| \mathbf{b}^\delta - A\mathbf{x}_\mu \right\|_2^2}{\left(\operatorname{trace}\left(I - A(A^T A + \mu L^T L)^{-1} A^T \right) \right)^2}. \tag{3.2}$$

Generalized Cross Validation (GCV) [18] prescribes that the regularization parameter be obtained as the minimizer of the function G, i.e.,

$$\mu_{\mathrm{GCV}} = \arg\min_{\mu} G(\mu).$$

When the adaptive approach is employed, i.e., when using Algorithm 1, at each iteration an ℓ^2-ℓ^2 functional similar to the one in (3.1) is minimized. Therefore, we can iteratively determine the regularization parameter $\mu_{\mathrm{GCV}}^{(k)}$ by applying the GCV criterion to the problem (2.8). This approach induces a nonstationary method that updates the regularization parameter at each iteration.

When the noise that corrupts the data is non-Gaussian, the 2-norm of the residual $\mathbf{b}^\delta - A\mathbf{x}_\mu$ may be very large even when μ is chosen so that $\mathbf{x}_\mu$ is an accurate approximation of the desired solution $\mathbf{x}^\dagger$, because $\mathbf{b}^\delta$ may contain large outliers. To avoid this difficulty, if $p < 2$, then we damp the outliers by smoothing the data vector $\mathbf{b}^\delta$. This is done by convolving $\mathbf{b}^\delta$ with a Gaussian filter with small variance. This modification is performed exclusively in (3.2), and the data vector $\mathbf{b}^\delta$ is not modified anywhere else. Note that, if $\mathbf{b}^\delta$ is the vectorization of a two-dimensional signal, e.g., an image, then we first reshape $\mathbf{b}^\delta$ into a matrix, convolve the matrix with a two-dimensional Gaussian filter, and, finally, reshape the smoothed matrix into a vector.

3.2.2. The residual whiteness principle. We outline the approach proposed in [10] to select the regularization parameter based on the Residual Whiteness Principle (RWP) introduced in [24].

When the noise that afflicts the data is white, e.g., Gaussian or Laplace, we can expect that, if $\mathbf{x}^*$ is an accurate approximation of the exact solution $\mathbf{x}^\dagger$, then

$$A\mathbf{x}^* - \mathbf{b}^\delta \approx A\mathbf{x}^\dagger - \mathbf{b}^\delta = \mathbf{b} - \mathbf{b}^\delta.$$

This suggests that we choose the regularization parameter μ so that the residual $A\mathbf{x}^* - \mathbf{b}^\delta$ is as "white" as possible. To measure the whiteness of a vector, in [10, 24] the norm of the normalized sample auto-correlation function was used. This function is defined by

$$\beta(\mathbf{x}) = \frac{1}{\|\mathbf{x}\|_2^2} (\mathbf{x} \star \mathbf{x}),$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

where $\star$ denotes the convolution operator.

Let $\mathbf{x}_\mu^{(k)}$ be the k-th iterate of either Algorithms 1 or 2 with parameter μ. Define the function $\mathcal{W}^{(k)} : \mathbb{R}^+ \to \mathbb{R}^+$ by

$$\mathcal{W}^{(k)}(\mu) = \left\| \beta\left(\mathbf{r}_\mu^{(k)}\right) \right\|_2^2 = \frac{\left\| \mathbf{r}_\mu^{(k)} \star \mathbf{r}_\mu^{(k)} \right\|_2^2}{\left\| \mathbf{r}_\mu^{(k)} \right\|_2^4},$$

where $\mathbf{r}_\mu^{(k)} = \mathbf{b}^\delta - A\mathbf{x}_\mu^{(k)}$. Imposing the whiteness of the residual error associated with $\mathbf{x}_\mu^{(k)}$ is equivalent to minimizing the function $\mathcal{W}^{(k)}$ with respect to μ.

Similarly to the computations for the GCV, we construct a nonstationary method where, at each iteration, the regularization parameter $\mu_{\mathrm{RWP}}^{(k)}$ is computed as

$$\mu_{\mathrm{RWP}}^{(k)} = \arg\min_\mu \mathcal{W}^{(k)}(\mu).$$

3.2.3. The discrepancy principle. We describe the nonstationary approach to determine the regularization parameter discussed in [11]. Assume that a fairly accurate estimate of the norm of the noise in the data $\mathbf{b}^\delta$ is available, namely

$$\left\| \mathbf{b}^\delta - \mathbf{b} \right\|_2 \leq \delta. \tag{3.3}$$

The discrepancy principle (DP) prescribes that the regularization parameter $\mu > 0$ be chosen as large as possible so that the corresponding computed approximate solution $\mathbf{x}_\mu$ of (1.3) satisfies

$$\left\| A\mathbf{x}_\mu - \mathbf{b}^\delta \right\|_2 \leq \tau\delta, \tag{3.4}$$

where $\tau > 1$ is a user-provided constant. When the noise is Gaussian and δ is a fairly accurate upper bound of the left-hand side of (3.3), the parameter τ often can be chosen close to unity.

We employ this criterion to determine a nonstationary sequence of regularization parameters such that, at each iteration, the DP is satisfied. Even though this approach can be used with both type of majorants, we only consider the fixed case in this paper, i.e., Algorithm 2.

Let $\mathbf{x}_\mu^{(k)}$ be the k-th iterate determined by Algorithm 2 with parameter μ. We pick $\mu_{\mathrm{DP}}^{(k)}$ such that

$$\left\| A\mathbf{x}_{\mu_{\mathrm{DP}}^{(k)}}^{(k)} - \mathbf{b}^\delta \right\|_2 = \tau\delta.$$

The parameter $\mu_{\mathrm{DP}}^{(k)}$ can be determined with a few iterations of Newton's method. The following results are shown in [11].

PROPOSITION 3.1. *Assume that A is of full column rank and let $\mathbf{x}_{\mu_{\mathrm{DP}}^{(k)}}^{(k)}$, $k = 1, 2, \ldots$, denote the iterates generated by Algorithm 2 equipped with the DP. Then, there is a subsequence $\mathbf{x}_{\mu_{\mathrm{DP}}^{(k_j)}}^{(k_j)}$ with a limit $\mathbf{x}^*$, such that*

$$\left\| A\mathbf{x}^* - \mathbf{b}^\delta \right\|_2 = \tau\delta.$$

THEOREM 3.2. *With the assumptions and notation of Proposition 3.1, let $\mathbf{x}^\delta$ denote the limit of the subsequence $\mathbf{x}_{\mu_{\mathrm{DP}}^{(k_j)}}^{(k_j)}$ with noise level δ. Then,*

$$\limsup_{\delta \searrow 0} \left\| \mathbf{x}^\delta - \mathbf{x}^\dagger \right\|_2 = 0.$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

4. Software. This sections describes the software that accompanies this paper[2]. We built a single function that carries out ℓ^p-ℓ^q minimization. The basic method can be run as

```
x=lplq(A,b);
```

Here, `A` represents the matrix A and can be either a matrix or a class for which a matrix-vector product and a matrix-vector product with the transpose are defined, and `b` is a column vector that contains the entries of $\mathbf{b}^\delta$. The output `x` contains the approximate solution as a column vector. Running the method in this way selects the default options for all parameters of the algorithms; see below.

It is possible to specify various parameters by including them in a structure as additional input variables

```
x=lplq(A,b,options);
```

Similarly as for the implementation of the `IRtools` toolbox [17], one obtains a structure containing all the default values for `options` with the command

```
default_options=lplq('defaults');
```

Table 4.1 describes the fields of the structure `options`.

TABLE 4.1:
Descriptions and accepted values for the fields of the `options` *structure.*

Field	**Description of the field**	**Accepted values**
`p`	Value of p for the ℓ^p norm.	$0 < p \leq 2$ (default 2).
`q`	Value of q for the ℓ^q norm.	$0 < q \leq 2$ (default 0.1).
`L`	Regularization matrix.	Matrix or class for which matrix-vector product and matrix-vector product with the transposition are implemented (default I).
`mu`	Value of the regularization parameter μ. If this field is populated, then this parameter is used in the iterations regardless of other options.	$\mu > 0$ (default `[]`).
`initialGuess`	Initial guess for the MM iterations. If this field is populated, then its value is used as $\mathbf{x}^{(0)}$. If this field is not populated or empty, then $\mathbf{x}^{(0)} = A^T\mathbf{b}^\delta$.	$\mathbf{x}^{(0)} \in \mathbb{R}^n$ (default `[]`).

Continued on next page

[2] https://etna.ricam.oeaw.ac.at/volumes/2021-2030/vol61/addition/p66.php

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 4.1:
Descriptions and accepted values for the fields of the `options` *structure. (Continued)*

Field	**Description of the field**	**Accepted values**
`choiceRule`	Rule for the automatic selection of the regularization parameter. If the field `mu` is non-empty, then this field is discarded.	`'CV'` for Cross Validation, `'MCV'` for Modified Cross Validation, `'GCV'` for Generalized Cross Validation (if this is selected the then adaptive majorant is used, regardless of other options), `RWP` for Residual Whiteness Principle, and `DP` for Discrepancy Principle (if this is selected the fixed majorant is used, regardless of other options, and the field `noiseNorm` must be populated) (default `'GCV'`).
`majorantType`	Type of the majorant.	`adaptive` for the adaptive majorant and `fixed` for the fixed majorant (default `adaptive`).
`epsilon`	Value of the smoothing parameter ε in (2.2).	$\varepsilon > 0$ (default 10^{-3}).
`tol`	Value of the tollerance for the stopping criterion.	tol > 0 (default 10^{-4}).
`maxIt`	Maximum number of iterations.	Any integer larger than 1 (default 100).
`noiseNorm`	Estimate of the norm of the noise δ. Must be populated if `choiceRule` is set to `'DP'`.	$\delta > 0$ or empty (default `[]`).
`tau`	Value of τ for the Discrepancy Principle in (3.4).	$\tau > 1$ (default 1.01).

Continued on next page

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 4.1:
Descriptions and accepted values for the fields of the `options` *structure. (Continued)*

Field	Description of the field	Accepted values
`size`	Dimensions for reshaping if GCV (with $p < 2$) or RWP are used. If GCV (with $p < 1$) is used and this field is populated, then the data vector $\mathbf{b}^\delta$ is reshaped to this size to carry out convolution with a Gaussian kernel to remove outliers. If the RWP is used and this field is populated, then the residual is reshaped and multi-dimensional convolution is carried out. In both cases only two- and three-dimensional vectors can be considered. If empty, the problem is considered one-dimensional.	Vectors with either 2 or 3 components such that the product of the entries is equal to m (default `[]`).
`trainingPercent`	Percentage of rows of A and elements of $\mathbf{b}^\delta$ that are not removed in either CV or MCV.	Any number between 0 and 100 (excluded) (default 90).
`trainingTestNum`	Number of times that either CV or MCV is repeated to reduce variability.	Any integer larger than or equal to 1 (default 10).
`trainingMu`	Values of μ that are tested in either CV or MCV.	Any vector with entries strictly greater than 0 (default `logspace(-3,2,10)`).
`restart`	Number of iterations after which the GKS method is restarted. If it is larger than the maximum number of iterations no restart is carried out.	Any integer number larger than 1 (default 101).
`waitbar`	Flag that determines if a waitbar is shown.	If it is set to `'off'` no waitbar is shown, if is set to `'on'` a waitbar is shown (default `'off'`).
`xTrue`	Exact solution of the problem (for testing purposes). If it is populated in the `outInfo` structure the RRE at each iteration is stored.	Column vector containing the exact solution (default `[]`).

ETNA
Kent State University and
Johann Radon Institute (RICAM)

It is possible to add an additional output argument

```
[x,outInfo]=lplq(A,b,options);
```

The `outInfo` structure contains additional information on the results of the algorithm. We describe in Table 4.2 the fields of this structure.

TABLE 4.2:
Descriptions of the fields of the `outInfo` *structure.*

Field	Description of the field
`iter`	Number of iterations carried out. If `choiceRule` is either `CV` or `MCV` it contains the number of iterations of the final run, i.e., the one carried out with all the data.
`mu`	Value of μ used. If the choice rule is stationary this field contains a number, if it is nonstationary it is a vector.
`res`	Vector containing the norm of the residual $\left\|A\mathbf{x}^{(k)} - \mathbf{b}^{\delta}\right\|_2$ at each iteration.
`RRE`	Vector containing the RRE $\left\|\mathbf{x}^{(k)} - \mathbf{x}^{\dagger}\right\|_2 / \left\|\mathbf{x}^{\dagger}\right\|_2$ at each iteration. This field is populated only if `options.xTrue` is non-empty.
`resCV`	Generated only if `choiceRule` is set to `CV`. It contains the residual r_j for each test carried out. Each column represents a different set of removed data and each row represents a different value of μ.
`RRECV`	Generated only if `choiceRule` is set to `CV`. It contains a matrix with the final RRE for each test carried out. Each column represents a different set of removed data and each row represents a different value of μ.
`diffMCV`	Generated only if `choiceRule` is set to `MCV`. It contains a matrix with the Δ_j for each test carried out. Each column represents a different set of removed data and each row represents a different value of μ.
`RREMCV`	Generated only if `choiceRule` is set to `MCV`. It contains a tensor with the final RRE for each test carried out. The first slice is related to the first set of indices removed I_1, the second slice is related to the second set of indices removed I_2. In each slice each column represents a different set of removed data and each row represents a different value of μ.

4.1. Supplementary software. We describe MATLAB classes that we built for the numerical experiments in this paper.

`Aclass` & `Aclass3`. We consider some deblurring problems in the numerical experiments. The space invariant deblurring problem for black and white images can be formulated as a Fredholm integral equation of the first kind of the form

$$g(s,t) = \int_{\mathbb{R}^2} k(s-u, t-v) f(u,v) \mathrm{d}u\mathrm{d}v + e(s,t),$$

where f is the unknown sharp image, k is a smooth kernel with compact support, often referred to as a point spread function (PSF), e models the error (noise), and g is a blur- and noise-contaminated image. Once discretized, the problem is of the form (1.2), where the pixels of the blurred and sharp images are ordered in lexicographical order. The number of entries of the solution $\mathbf{x}$ equals the number of pixels of the image and typically is large, and the matrix

ETNA
Kent State University and
Johann Radon Institute (RICAM)

A has a structure that depends on the boundary conditions imposed on the problem. We have

$$A = T + E + R,$$

where T is a block Toeplitz with Toeplitz blocks matrix, E is a matrix of small norm, and R is a matrix of small rank; see [19] for more details. Due to this structure, the product of the matrix A with a vector $\mathbf{x}$ is easy to evaluate: one first reshapes the vector $\mathbf{x}$ into a matrix that represents the image, then the image is padded to satisfy the desired boundary conditions. The padded image is convolved with the PSF using periodic boundary conditions; this corresponds to multiplying the padded vector with a block circulant with circulant blocks matrix C. This operation can be carried out cheaply with the aid of the `fft` algorithm. Finally, the added boundary is cropped and the blurred image is reordered into a vector. This procedure is appropriate, e.g., for reflexive and anti-reflexive boundary conditions.

The matrix-vector product with A^T is less trivial to carry out when imposing anti-reflexive boundary conditions. We use the approach in [16]. The authors suggest to substitute A^T with its approximation A', that is the blurring operator with the same boundary conditions as A, but where the PSF has been flipped. Matrix-vector products with the matrix A' can be computed by simply following the steps for the matrix-vector product described above and using C^T instead of C.

We implemented A as described as a MATLAB class and denote it by `Aclass`. The syntax for the creation of an object of this class is

```
A=Aclass(PSF,center,bc,n1,n2);
```

The field `PSF` should contain the point spread function as a matrix, the vector `center` contains the indices of the center of the PSF, `bc` is a string containing the boundary conditions; it can be

- `'zero'` for zero Dirichlet boundary conditions;
- `'periodic'` for periodic boundary conditions;
- `'reflexive'` for reflexive boundary conditions;

and the numbers `n1` and `n2` are the sizes of $\mathbf{x}$ considered as an image.

The class can carry out matrix-vector products with $\mathbf{x}$ both in vector and matrix form and will return a result of the same size as $\mathbf{x}$. For this class, we implemented the matrix-vector product operation with A and with the transpose of A, the `size` command, and the `normest` command. This latter command returns the norm of C.

For the blurring operator for color images, we implemented the MATLAB class `Aclass3`. It can be constructed by

```
A=Aclass3(PSF_R,center_R,PSF_G,center_G,PSF_B,center_B,
   Mix_channels,bc,n1,n2)
```

Color images are represented by a three-dimensional tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times 3}$. Each frontal slice represents one of the three color channels red, green, and blue (RGB). Once ordered lexicographically, the vector obtained $\mathbf{x} \in \mathbb{R}^n$, with $n = n_1 \cdot n_2 \cdot 3$, contains in the first $n_1 \cdot n_2$ entries the pixels of the red channel, in the second $n_1 \cdot n_2$ entries the pixels of the green channel, and in the last $n_1 \cdot n_2$ entries the pixels of the blue channel. We construct a blurring matrix of the form

$$A = A_M \otimes \begin{bmatrix} A_R & 0 & 0 \\ 0 & A_G & 0 \\ 0 & 0 & A_B \end{bmatrix}, \tag{4.1}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

where $\otimes$ denotes the Kronecker product, $A_R, A_G, A_B \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ are blurring matrices that blur each channel separately, and $A_M \in \mathbb{R}^{3\times 3}$ mixes the channels. In particular, let

$$\tilde{\mathbf{x}} = \begin{bmatrix} A_R & 0 & 0 \\ 0 & A_G & 0 \\ 0 & 0 & A_B \end{bmatrix} \mathbf{x},$$

where $\mathbf{x}$ is the vectorization of a color image. One can think of $\tilde{\mathbf{x}}$ as the vectorization of a color image $\tilde{X}$. Let $\tilde{X}^R_{i,j}$, $\tilde{X}^G_{i,j}$, and $\tilde{X}^B_{i,j}$, denote the values of the red, blue, and green, channels of the (i,j)th pixel of $\tilde{X}$. Let $\mathbf{y} = A\mathbf{x}$ be the vectorization of a color image Y. Then, the (i,j)th pixel of Y is

$$\begin{bmatrix} Y^R_{i,j} \\ Y^G_{i,j} \\ Y^B_{i,j} \end{bmatrix} = A_M \begin{bmatrix} \tilde{X}^R_{i,j} \\ \tilde{X}^G_{i,j} \\ \tilde{X}^B_{i,j}, \end{bmatrix}.$$

In `Aclass3` the `PSF_X` and `center_X`, with `X` either `R`, `G`, or `B`, represent the PSF and center of the respective channel, `Mix_channel` contains A_M, `bc` contains the boundary conditions, and `n1` and `n2` are n_1 and n_2, respectively. The boundary conditions available and the operations implemented are the same as for `Aclass`.

`TVclass` & `TVclass3`. A popular choice of the regularization operator is total variation [23, 32]. Assuming that the vector $\mathbf{x} \in \mathbb{R}^{n\times n}$ is a vectorized version of a, for simplicity of notation, square $\sqrt{n} \times \sqrt{n}$ matrix, we let

$$L_1 = \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \\ 1 & & & & -1 \end{bmatrix} \in \mathbb{R}^{\sqrt{n}\times\sqrt{n}}$$

and define

$$L = \begin{bmatrix} L_1 \otimes I \\ I \otimes L_1 \end{bmatrix} \in \mathbb{R}^{2n\times n}, \tag{4.2}$$

where $I \in \mathbb{R}^{\sqrt{n}\times\sqrt{n}}$ denotes the identity matrix. Since L_1 is a circulant matrix, matrix-vector products with L and its transpose can be carried out efficiently by using the `fft` algorithm. The MATLAB class `TVclass` implements this operator. It can be defined by

```
L=TVclass(n1,n2);
```

where `n1` and `n2` are the sizes of $\mathbf{x}$ regarded as a matrix. Similarly to the `Aclass`, this class can evaluate matrix-vector products with $\mathbf{x}$ both in vector and matrix form, and will return a vector or a matrix.

If $\mathbf{x}$ is the vectorization of a color image, then we let

$$L_{\text{color}} = \begin{bmatrix} L & & \\ & L & \\ & & L \end{bmatrix},$$

where L is defined in (4.2). The class `TVclass3` implements this operator and can evaluate matrix-vector products with L_{color} and its transpose. It is called by

```
L=TVclass3(n1,n2);
```

where `n1` and `n2` are the sizes of the frontal slices of $\mathbf{x}$ considered as a tensor.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

5. Numerical experiments. We describe a few computed examples that illustrate the performances of the software and first compare our software to the ℓ^2-$\ell^{2/3}$ thresholding in [35] in terms of accuracy and computational cost. Then, for each parameter choice rule, we show a numerical example that illustrates its advantage and performance. In all our experiments we set

- the exponent of the norm in the regularization term to $q = 0.1$;
- the smoothing parameter to $\varepsilon = 1$;
- the maximum number of iteration to 500;
- the number of iterations after which the generalized Krylov subspace is restarted to 30;
- the tolerance for the stopping criterion to 10^{-4}.

This choice of ε has been found to be suitable for many image restoration problems. However, for nonlinear regression problems this value of ε may be too large; see [4] for an illustration. In all our experiments, we let L be the TV operator described above.

To measure the accuracy of the computed solution, we use the Relative Restoration Error (RRE) defined by

$$RRE(\mathbf{x}) = \frac{\left\|\mathbf{x} - \mathbf{x}^\dagger\right\|_2}{\left\|\mathbf{x}^\dagger\right\|_2},$$

where $\mathbf{x}^\dagger$ denotes the exact solution of the problem. All the experiment are carried out in MATLAB 2021b on a laptop running Windows 10 with 16GB of RAM and an AMD Ryzen 7 5800HS CPU.

Comparison with ℓ^2-$\ell^{2/3}$ thresholding. We consider an image deblurring problem where the exact image is sparse. The shrinkage method proposed in [35] solves the minimization problem

$$\mathbf{x}^* = \arg\min \frac{1}{2}\left\|A\mathbf{x} - \mathbf{b}^\delta\right\|_2^2 + \frac{3\mu}{2}\left\|\mathbf{x}\right\|_{2/3}^{2/3}.$$

It is not straightforward to introduce a regularization operator in the algorithm in [35] and, therefore, we pick a sparse solution. First we set in our method $L = I$, $p = 2$, and $q = 0.1$, and subsequently we vary q between 0.1 and 1.

The exact image in Figure 5.1(a) is blurred with the nonsymmetric PSF shown in Figure 5.1(b). We crop the boundaries to simulate realistic data; see [19] for more details. Since the method in [35] only allows $p = 2$, we add 3% white Gaussian noise, i.e.,

$$\sigma_{\text{Gauss}} = \frac{\left\|\mathbf{b}^\delta - \mathbf{b}\right\|_2}{\left\|\mathbf{b}\right\|_2} = 0.03.$$

We use the DP to select the regularization parameter in both methods. Our algorithm uses the nonstationary approach described in Section 3.2.3. To apply the DP in [35], we run the algorithm for 15 values of μ and we pick the largest of these values for which the computed solution satisfies the DP. Table 5.1 reports the results so obtained. For the ℓ^2-$\ell^{2/3}$ thresholding method described in [35], we show results for both the DP parameter and the optimal one, i.e., the one that minimizes the RRE. The column labeled *Total time* shows the total time required to run the method for the 15 chosen values of μ, while in the *Run time* column, we display the time required to carry out the iterations with this particular choice of μ. We can observe that our method outperforms the method in [35] in terms of accuracy regardless of how the regularization parameter is chosen. Moreover, if we consider the total time required, the method [35] is computationally more demanding than our software. Finally, we display

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 5.1
Comparison with ℓ^2-$\ell^{2/3}$ thresholding: RRE obtained with both methods and CPU time. For the ℓ^2-$\ell^{2/3}$ thresholding algorithm, we show results for both the DP and the optimal choice of μ. The column Total time *reports the total required time for all* 15 *considered values of μ, while the column* Run time *shows the CPU time for a single run. The ℓ^p-ℓ^q minimization method is run using the Algorithm 2 with the DP and requires only a single run of the algorithm, therefore, the last two columns coincide.*

Method	RRE	Total time (sec.)	Run time (sec.)
$\ell^2 - \ell^{2/3}$ thresholding (DP)	0.22230	34.96	2.40
$\ell^2 - \ell^{2/3}$ thresholding (Opt.)	0.13184	34.96	1.79
$\ell^p - \ell^q$ minimization ($q = 0.1$)	0.11195	9.73	9.73

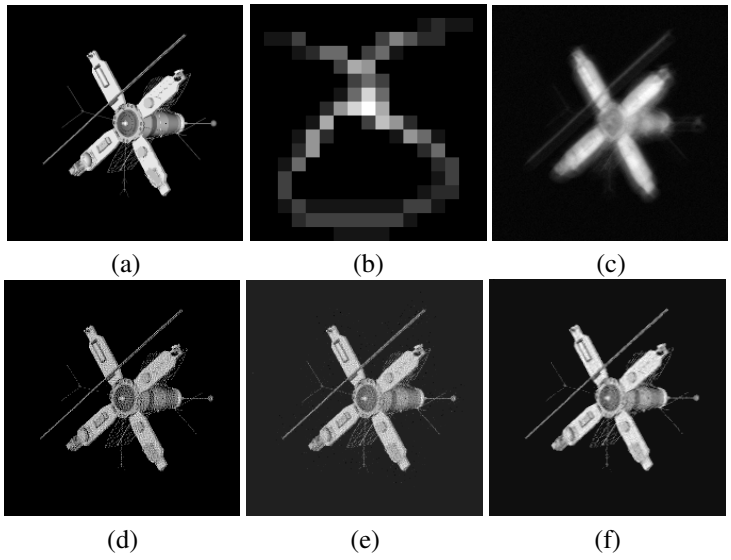

FIGURE 5.1. *Comparison with ℓ^2-$\ell^{2/3}$ thresholding: (a) exact image (222×222 pixels), (b) PSF (17×17 pixels), (c) blurred and noisy image with 3% white Gaussian noise, (d) restored image obtained with ℓ^2-$\ell^{2/3}$ thresholding (DP parameter), (e) restored image with ℓ^2-$\ell^{2/3}$ thresholding (Optimal parameter), (f) ℓ^p-ℓ^q minimization (Algorithm 2 with the DP).*

computed approximate solutions in Figure 5.1(d)–(f). Visual inspection of the reconstructions show the ℓ^2-$\ell^{2/3}$ thresholding algorithm to produce "shrunk" solutions that are of smaller norm than the exact solution. This is not an issue for our method.

An important feature of the software described in this paper is that it allows the determination of the regularization parameter μ by several methods, and also makes it possible for a user to choose the values of the parameters p and q that yield the best results. However, if suitable values of p, q, and μ are known before the beginning of the computations, then faster methods are available, such as the scheme presented by Zhang and Ye [35].

We now run Algorithm 2 with the DP for several values of q. Figure 5.2 reports the RREs so-obtained, and compares them to the RRE of the approximate solution determined with ℓ^2-$\ell^{2/3}$ thresholding. We can observe that Algorithm 2 outperforms ℓ^2-$\ell^{2/3}$ thresholding for all considered values of q. Moreover, as expected, we can observe that the RRE increases with q (except that for $q = 1$).

Cameraman. We illustrate the application and performance of Cross Validation (CV) to deblurring the Cameraman image in Figure 5.3(c). This image was obtained by blurring the exact image in Figure 5.3(a) with the motion PSF shown in Figure 5.3(b) and

ETNA
Kent State University and
Johann Radon Institute (RICAM)

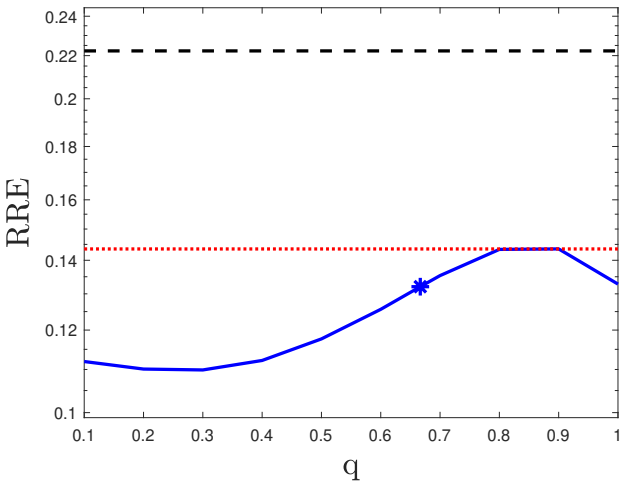

FIGURE 5.2. *Comparison with ℓ^2-$\ell^{2/3}$ thresholding: RRE obtained with Algorithm 2 using the DP compared with the RRE obtained with ℓ^2-$\ell^{2/3}$ thresholding with both the optimal and DP parameter. The solid blue graph reports the RRE obtained with Algorithm 2 for several values of q, where we mark the case $q = 2/3$ with an asterisk, the dotted red graph reports the RRE obtained with ℓ^2-$\ell^{2/3}$ thresholding with the optimal parameter, and the black dashed graph reports the RRE obtained with ℓ^2-$\ell^{2/3}$ thresholding with the DP parameter.*

adding 2% of white Gaussian noise. Since the image is general, we imposed reflexive boundary conditions. The number of different sets with removed indices, i.e., the variable `options.trainingTestNum`, is set to 3. Since the noise is Gaussian, we let $p = 2$. The values of `options.trainingPercent` and `options.trainingMu` are set to their default values. We use *fixed* majorants. This implies that the algorithm is run 30 times with the "pruned" data set and once with the complete data set. Table 5.2 reports the CPU time required for the computations and the RRE of the final result. We can see that the computational cost is reasonable if we consider that the method is run 31 times in total. The RRE is small and visual inspection of the computed solution shown in Figure 5.3(d) confirms that the computed approximation is very accurate.

Saturn. We show the performance of the MCV parameter choice rule. The Saturn image displayed in Figure 5.3(e) is blurred with the disk PSF in Figure 5.3(f), and we add 1% of white Gaussian noise and 10% of salt-and-pepper noise. This gives the blurred and noisy image shown in Figure 5.1(g). Similarly as in the previous examples, we use reflexive boundary conditions. Since the noise is mixed, we let $p = 0.8$. Similarly to the CV case, we set the values of `options.trainingPercent` and `options.trainingMu` to their defaults and set `options.trainingTestNum` to 3. Fixed majorants are used. The algorithm is run 60 time with the smaller data set and once with the full data. We can find the results in Table 5.2 and the computed solution is shown in Figure 5.3(h). The computational time is seen to be quite long, due to the large number of times that the method has to be run. But the computed solution is very accurate! This can be confirmed by visual inspection of the reconstruction and its RRE value. We would like to stress that the reconstruction of data corrupted by mixed noise is a very challenging task and that our method is able to produce very accurate solution.

Peppers. We turn to GCV. The Peppers image in Figure 5.3(i) is blurred with the Gaussian PSF in Figure 5.3(j), and we further corrupt the image with 25% salt-and-pepper noise. We show the blurred and noisy image in Figure 5.3(k). Since the image is generic, we impose

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 5.2
Performances of the proposed software in terms of accuracy (RRE) and CPU time.

Example	Parameter rule	RRE	CPU time (sec.)
Cameraman	CV	0.083239	148.41
Saturn	MCV	0.066066	422.85
Peppers	GCV	0.083682	158.77
Tomography	RWP	0.22223	3.58
Color image	DP	0.073041	2.09

reflexive boundary conditions. The parameter p is set to 0.8; all other parameters in the algorithm are set to their default values or to the values described above. Note that we populate the field `options.size` to more effectively smooth $\mathbf{b}^\delta$ in the GCV to damp the outliers in the data. Table 5.2 shows the results of the computations and the restored image is displayed in Figure 5.3(l). The computational cost is reasonable, especially when we consider that the image is larger (492×492 pixels) than in the previous examples. The computed solution is very accurate. We recall that no additional information about the noise or the problem is required besides A and $\mathbf{b}^\delta$.

Tomography. We now illustrate the RWP method for determining the regularization parameter. Consider a tomography problem built with the IRtools package [17]. The 256 pixels phantom in Figure 5.3(m) is shined on by 362 parallel rays at 90 angles equispaced between 0 and π. This determines the sinogram in Figure 5.3(n). The problem of restoring the image is underdetermined with $A \in \mathbb{R}^{32580\times 65536}$; see [28] for a discussion on tomography problems. We corrupt the data with 3% white Gaussian noise and, therefore, we set $p = 2$. Since the data is two-dimensional, we populate the field `options.size`. However, note that the code would run with no issue also if this field is left empty. We use the default majorant, i.e., the adaptive one. Table 5.2 shows results of the computations, and the computed solution is displayed in Figure 5.3(o). The table shows the computational cost to be very low and the computed solution to be quite accurate. This is confirmed by visual inspection of the reconstruction.

Color image. We conclude this section by showing the performance of the DP method. Consider the color image in Figure 5.3(p), and blur each channel with the same average PSF shown in Figure 5.3(q). The matrix A_M in (4.1) is

$$A_M = \frac{1}{10}\begin{bmatrix} 6 & 2 & 2 \\ 1 & 8 & 1 \\ 1 & 3 & 6 \end{bmatrix}.$$

We corrupt the data with 1% of white Gaussian noise and obtain the blurred and noisy image of Figure 5.3(r). Since the image is generic, we impose reflexive boundary conditions. Our method is run with $p = 2$, and we select DP as the parameter choice rule. This is specified in the field `options.noiseNorm`. Note that, since we use the DP, the majorant used is the fixed one. The results obtained are reported in Table 5.2 and the computed restoration is shown in Figure 5.3(s). We can observe that the computational cost is very low and that the computed solution is quite accurate.

6. Conclusion and extensions. This paper illustrates the performance of two generalize Krylov subspace algorithms for the solution of large-scale minimization problems (1.3) and discusses software that implements these algorithms. Several approaches to determine the regularization parameter are implemented. Convergence results can be found in [14], where also computed examples, including a comparison with FISTA [2] are presented. Several

ETNA
Kent State University and
Johann Radon Institute (RICAM)

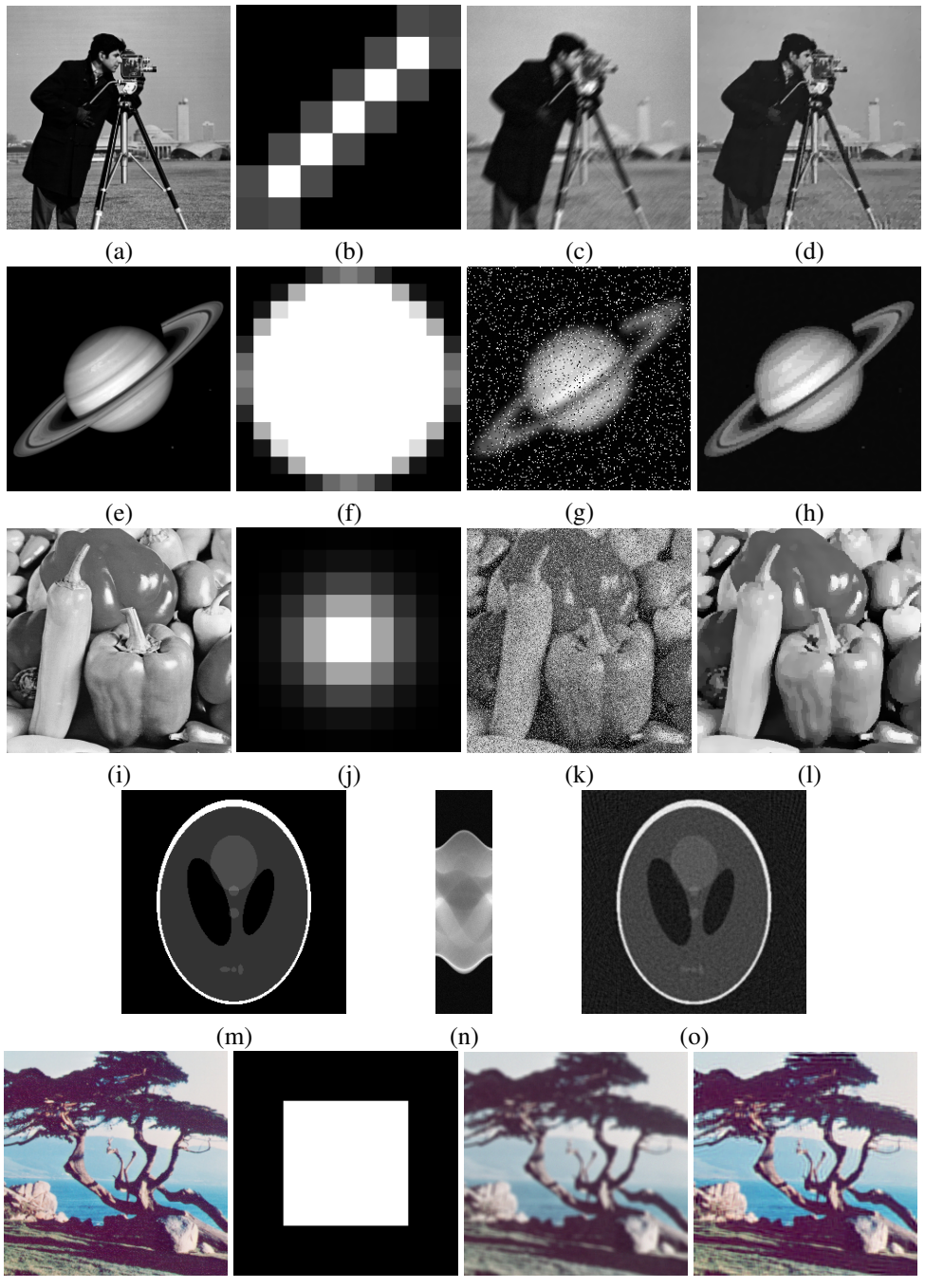

FIGURE 5.3. *Numerical experiments. Cameraman: (a) exact image (*246×246 *pixels), (b) PSF (*5×5 *pixels), (c) blurred and noisy image with* 2% *of white Gaussian noise, (d) restored image. Saturn: (e) exact image (*230×230 *pixels), (f) PSF (*13×13 *pixels), (g) blurred and noisy image with* 10% *salt and pepper noise and* 1% *white Gaussian noise, (h) restored image. Peppers: (i) exact image (*492×492 *pixels), (j) PSF (*10×10 *pixels), (k) blurred and noisy image with* 25% *salt and pepper noise, (l) restored image. Tomography: (m) exact image (*256×256 *pixels), (n) sinogram (*90 *angles,* 362 *rays) with* 3% *white Gaussian noise, (o) restored image. Color image: (p) exact image (*246×246 *pixels), (q) PSF (*9×9 *pixels), (r) blurred and noisy image with* 1% *white Gaussian noise, (s) restored image.*

ETNA
Kent State University and
Johann Radon Institute (RICAM)

extensions of this code can quite easily be developed, such as combining ℓ^p-ℓ^q minimization with the use of median filters as described in [1], and imposing a nonnegativity constraint as illustrated in [9].

Supplementary material. The accompanying software is available at
https://etna.ricam.oeaw.ac.at/volumes/2021-2030/vol61/addition/p66.php
in form of a compressed file entitled lplq.zip. Installation details are discussed in the file CodePrimer.pdf as well as in the README.md file

Acknowledgments. A.B. is partially supported by the PRIN 2022 PNRR project no. P2022PMEN2 financed by the European Union—NextGenerationEU and by the Italian Ministry of University and Research (MUR) and by Fondazione di Sardegna, Progetto biennale bando 2021, "Computational Methods and Networks in Civil Engineering (COMANCHE)". A.B. is a member of the GNCS group of INdAM that partially supported this work with the INdAM-GNCS 2023 Project "Tecniche numeriche per lo studio dei problemi inversi e l'analisi delle reti complesse" (CUP_E53C22001930001).

REFERENCES

[1] M. ALOTAIBI, A. BUCCINI, AND L. REICHEL, *Restoration of blurred images corrupted by impulse noise via median filters and ℓ^p-ℓ^q minimization*, in 21st International Conference on Computational Science and Its Applications (ICCSA), IEEE Conferences Proceedings, Los Alamitos, 2021, pp. 112–122.
[2] A. BECK AND M. TEBOULLE, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci., 2 (2009), pp. 183–202.
[3] A. BUCCINI, *Regularizing preconditioners by non-stationary iterated Tikhonov with general penalty term*, Appl. Numer. Math., 116 (2017), pp. 64–81.
[4] A. BUCCINI, O. DE LA CRUZ CABRERA, M. DONATELLI, A. MARTINELLI, AND L. REICHEL, *Large-scale regression with non-convex loss and penalty*, Appl. Numer. Math., 157 (2020), pp. 590–601.
[5] A. BUCCINI, O. DE LA CRUZ CABRERA, C. KOUKOUVINOS, M. MITROULI, AND L. REICHEL, *Variable selection in saturated and supersaturated designs via minimization*, Commun. Statist. Simulation Comput., 52 (2023), pp. 4326–4347.
[6] A. BUCCINI, M. DONATELLI, AND L. REICHEL, *Iterated Tikhonov with a general penalty term*, Numer. Linear Algebra Appl., 24 (2017), Art. e2089 (12 pages).
[7] A. BUCCINI, G. HUANG, L. REICHEL, AND F. YIN, *On the choice of regularization matrix for an ℓ_2-ℓ_q minimization method for image restoration*, Appl. Numer. Math., 164 (2021), pp. 211–221.
[8] A. BUCCINI, M. PASHA, AND L. REICHEL, *Generalized singular value decomposition with iterated Tikhonov regularization*, J. Comput. Appl. Math., 373 (2020), Art. 112276 (9 pages).
[9] A. BUCCINI, M. PASHA, AND L. REICHEL, *Modulus-based iterative methods for constrained ℓ_p-ℓ_q minimization*, Inverse Problems, 36 (2020), Art. 084001 (26 pages).
[10] A. BUCCINI, M. PRAGLIOLA, L. REICHEL, AND F. SGALLARI, *A comparison of parameter choice rules for ℓ_p-ℓ_q minimization*, Ann. Univ. Ferrara Sez. VII Sci. Mat., 68 (2022), pp. 441–463.
[11] A. BUCCINI AND L. REICHEL, *An ℓ^2-ℓ^q regularization method for large discrete ill-posed problems*, J. Sci. Comput., 78 (2019), pp. 1526–1549.
[12] ———, *An ℓ^p-ℓ^q minimization method with cross-validation for the restoration of impulse noise contaminated images*, J. Comput. Appl. Math., 375 (2020), Art. 112824 (16 pages).
[13] ———, *Generalized cross validation for ℓ^p-ℓ^q minimization*, Numer. Algorithms, 88 (2021), pp. 1595–1616.
[14] ———, *Limited memory restarted ℓ^p-ℓ^q minimization methods using generalized Krylov subspaces*, Adv. Comput. Math., 49 (2023), Art. 26 (26 pages).
[15] J. W. DANIEL, W. B. GRAGG, L. KAUFMAN, AND G. W. STEWART, *Reorthogonalization and stable algorithms for updating the Gram–Schmidt QR factorization*, Math. Comput., 30 (1976), pp. 772–795.
[16] M. DONATELLI, D. MARTIN, AND L. REICHEL, *Arnoldi methods for image deblurring with anti-reflective boundary conditions*, Appl. Math. Comput., 253 (2015), pp. 135–150.
[17] S. GAZZOLA, P. C. HANSEN, AND J. G. NAGY, *IR Tools: a MATLAB package of iterative regularization methods and large-scale test problems*, Numer. Algorithms, 81 (2019), pp. 773–811.
[18] G. H. GOLUB, M. HEATH, AND G. WAHBA, *Generalized cross-validation as a method for choosing a good ridge parameter*, Technometrics, 21 (1979), pp. 215–223.
[19] P. C. HANSEN, J. G. NAGY, AND D. P. O'LEARY, *Deblurring Images: Matrices, Spectra, and Filtering*, SIAM, Philadelphia, 2006.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

[20] G. HUANG, A. LANZA, S. MORIGI, L. REICHEL, AND F. SGALLARI, *Majorization-minimization generalized Krylov subspace methods for ℓ_p-ℓ_q optimization applied to image restoration*, BIT Numer. Math., 57 (2017), pp. 351–378.

[21] S. KINDERMANN, *Convergence analysis of minimization-based noise level-free parameter choice rules for linear ill-posed problems*, Electron. Trans. Numer. Anal., 38 (2011), pp. 233–257. https://etna.ricam.oeaw.ac.at/vol.38.2011/pp233-257.dir/pp233-257.pdf

[22] J. LAMPE, L. REICHEL, AND H. VOSS, *Large-scale Tikhonov regularization via reduction by orthogonal projection*, Linear Algebra Appl., 436 (2012), pp. 2845–2865.

[23] A. LANZA, S. MORIGI, L. REICHEL, AND F. SGALLARI, *A generalized Krylov subspace method for ℓ_p-ℓ_q minimization*, SIAM J. Sci. Comput., 37 (2015), pp. S30–S50.

[24] A. LANZA, M. PRAGLIOLA, AND F. SGALLARI, *Residual whiteness principle for parameter-free image restoration*, Electron. Trans. Numer. Anal., 53 (2020), pp. 329–351. https://etna.ricam.oeaw.ac.at/vol.53.2020/pp329-351.dir/pp329-351.pdf

[25] ———, *Parameter-free restoration of piecewise smooth images*, Electron. Trans. Numer. Anal., 59 (2023), pp. 202–229. https://etna.ricam.oeaw.ac.at/vol.59.2023/pp202-229.dir/pp202-229.pdf

[26] H. LIU AND R. F. BARBER, *Between hard and soft thresholding: optimal iterative thresholding algorithms*, Inf. Inference, 9 (2020), pp. 899–933.

[27] T. MACH, L. REICHEL, AND M. VAN BAREL, *Adaptive cross approximation for Tikhonov regularization in general form*, Numer. Algorithms, 92 (2023), pp. 815–830.

[28] F. NATTERER, *The Mathematics of Computerized Tomography*, SIAM, Philadelphia, 2001.

[29] Y. PARK, L. REICHEL, G. RODRIGUEZ, AND X. YU, *Parameter determination for Tikhonov regularization problems in general form*, J. Comput. Appl. Math., 343 (2018), pp. 12–25.

[30] M. PASHA, S. GAZZOLA, C. SANDERFORD, AND U. O. UGWU, *TRIPs-Py: Techniques for regularization of inverse problems in Python*, arXiv preprint arXiv:2402.17603, 2024. https://arxiv.org/abs/2402.17603

[31] L. REICHEL AND G. RODRIGUEZ, *Old and new parameter choice rules for discrete ill-posed problems*, Numer. Algorithms, 63 (2013), pp. 65–87.

[32] L. I. RUDIN, S. OSHER, AND E. FATEMI, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.

[33] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, 2nd. ed., SIAM, Philadelphia, 2003.

[34] M. STONE, *Cross-validatory choice and assessment of statistical prediction*, J. R. Stat. Soc. Ser. B, 36 (1974), pp. 111–147.

[35] Y. ZHANG AND W. YE, *$L_{2/3}$ regularization: Convergence of iterative thresholding algorithm*, J. Vis. Commun. Image Represen., 33 (2015), pp. 350–357.