An ℓ^p-ℓ^q minimization method with cross-validation for the restoration of impulse noise contaminated images

Alessandro Buccini[a], Lothar Reichel[b]

[a] *Department of Mathematics and Computer Science, University of Cagliari, Cagliari, 09124, Italy.*
[b] *Department of Mathematical Sciences, Kent State University, Kent, OH 44242, USA.*

Abstract

Discrete ill-posed problems arise in many areas of science and engineering. Their solutions, if they exist, are very sensitive to perturbations in the data. Regularization aims to reduce this sensitivity. Many regularization methods replace the original problem by a minimization problem with a fidelity term and a regularization term. Recently, the use of a p-norm to measure the fidelity term and a q-norm to measure the regularization term has received considerable attention. The relative importance of these terms is determined by a regularization parameter. When the perturbation in the available data is made up of impulse noise and a sparse solution is desired, it often is beneficial to let $0 < p, q < 1$. Then the p- and q-norms are not norms. The choice of a suitable regularization parameter is crucial for the quality of the computed solution. It therefore is important to develop methods for determining this parameter automatically, without user-interaction. However, the latter has so far not received much attention when the data is contaminated by impulse noise. This paper discusses two approaches based on cross validation for determining the regularization parameter in this situation. Computed examples that illustrate the performance of these approaches when applied to the restoration of impulse noise contaminated images are presented.

Keywords: ℓ^p-ℓ^q minimization, ill-posed problem, iterative method
2010 MSC: 65F10, 65R32, 90C26

Dedicated to Fiorella Sgallari on the occasion of her 65th birthday.

Email addresses: alessandro.buccini@unica.it (Alessandro Buccini), reichel@math.kent.edu (Lothar Reichel)

1. Introduction

We consider the computation of an approximate solution of problems of the form

$$A\mathbf{x} + \boldsymbol{\delta} = \mathbf{b}^\delta, \tag{1}$$

where $A \in \mathbb{R}^{m\times n}$ is a large matrix, whose singular values decrease to zero gradually with no significant gap, the vector $\mathbf{b}^\delta \in \mathbb{R}^m$ represents measured error-contaminated data, and $\boldsymbol{\delta} \in \mathbb{R}^m$ denotes the unknown error. We will sometimes refer to $\boldsymbol{\delta}$ as noise. The quotient of the largest and smallest singular values of A is known as the condition number of A. Due to the decrease of the singular values to zero, this condition number is large. This makes it difficult to compute a meaningful approximate solution of (1); see below.

Problems of the kind (1) are commonly referred to as *discrete ill-posed problems*. They typically arise from the discretization of ill-posed problems, such as Fredholm integral equations of the first kind with a smooth kernel; see, e.g., [12, 16, 17] for discussions on ill-posed and discrete ill-posed problems.

Let $\mathbf{b} = [b_i] \in \mathbb{R}^m$ denote the unknown error-free vector associated with $\mathbf{b}^\delta = [b_i^\delta]$. Thus, $\mathbf{b}^\delta = \mathbf{b} + \boldsymbol{\delta}$. In the present paper, we will assume that the error $\boldsymbol{\delta}$ in $\mathbf{b}^\delta$ is made up of impulse noise, possibly together with Gaussiam noise. Impulse noise affects only a certain percentage of the entries of $\mathbf{b}$ and leaves the other entries unchanged. Specifically,

$$b_i^\delta = \begin{cases} d_i & \text{with probability } \sigma, \\ b_i & \text{with probability } 1-\sigma, \end{cases} \tag{2}$$

where the d_i are identically and uniformly distributed random numbers in an interval $[d_{\min}, d_{\max}]$, which is the dynamic range of b_i. If $d_i \in \{d_{\min}, d_{\max}\}$, i.e., in case all d_i attain their maximum or minimum achievable values, impulse noise is commonly referred to as salt-and-pepper noise. Impulse noise simulates the effect of broken sensors on the measuring device. The application of primary interest to us is the restoration of blurred and noise-contaminated images, however, the techniques described also can be used for other applications.

Since $\boldsymbol{\delta}$ is not known, a naïve approach to determine an approximation of the solution of (1) is to solve the least-squares problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}^\delta\|_2. \tag{3}$$

However, due to the large condition number of the matrix A and the error $\boldsymbol{\delta}$ in $\mathbf{b}^\delta$, the solution $A^\dagger\mathbf{b}^\delta$ of (3), where $A^\dagger$ denotes the Moore–Penrose pseudoinverse of A, generally does not furnish a meaningful approximation of the desired vector

$$\widehat{\mathbf{x}} := A^\dagger\mathbf{b}. \tag{4}$$

To achieve an accurate approximation of $\widehat{\mathbf{x}}$, the least-squares problem (3) is replaced by a minimization problem, whose solution is less sensitive to the error $\boldsymbol{\delta}$ in $\mathbf{b}^\delta$ than the solution of (3). This replacement is known as *regularization*. A regularization technique that recently has received considerable attention, see,

e.g., [4, 5, 8, 13, 19, 20, 25] and references therein, is to replace the least-squares problem (3) by an ℓ^p-ℓ^q minimization problem of the form

$$\mathbf{x}^* := \arg\min_{\mathbf{x}\in\mathbb{R}^n} \left\{ \frac{1}{p} \left\| A\mathbf{x} - \mathbf{b}^\delta \right\|_p^p + \frac{\mu}{q} \left\| L\mathbf{x} \right\|_q^q \right\}, \tag{5}$$

where the regularization matrix $L \in \mathbb{R}^{\ell\times n}$ is such that

$$\mathcal{N}(A) \cap \mathcal{N}(L) = \{\mathbf{0}\}. \tag{6}$$

Here $\mathcal{N}(M)$ denotes the null space of the matrix M, and

$$\|\mathbf{z}\|_s := \left(\sum_{j=1}^{r} |z_j|^s \right)^{1/s}, \quad \mathbf{z} = [z_1, z_2, \ldots, z_r]^T \in \mathbb{R}^r.$$

We will refer to $\|\mathbf{z}\|_s$ as the s-norm of $\mathbf{z}$ also for $0 < s < 1$, even though the mapping $\mathbf{z} \to \|\mathbf{z}\|_s$ is not a norm for these s-values. We are interested in solving (5) for $0 < p, q < 1$. Note that if either $p < 1$ or $q < 1$, then the functional (5) generally is non-convex. When $p = q = 2$, the minimization problem (5) reduces to Tikhonov regularization [15, 17, 24].

The first term in (5) is known as the *fidelity term* and the second term as the *regularization term*. The matrix L in the latter term often is chosen to be a discretized differential operator. We are primarily concerned with the restoration of images that have been contaminated by impulse noise and possibly also by Gaussian noise. Let the image be made up of $m \times m$ pixels. We let the ntries of the vector $\mathbf{x} \in \mathbb{R}^n$ represent the pixels enumerated column-wise. The matrix $A \in \mathbb{R}^{n\times n}$ represents a blurring operator and we will let L be a scaled discretized Laplacian in two space-dimension, i.e.,

$$L := L_2 \otimes I + L_2 \otimes I, \tag{7}$$

where the symmetric tridiagonal matrix

$$L_2 = \begin{bmatrix} 1 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 1 \end{bmatrix} \in \mathbb{R}^{m\times m}$$

is a scaled discrete Laplacian with Neumann boundary conditions in one space-dimension, I denotes the identity matrix of order m, and $\otimes$ stands for the Kronecker product. Because A is a blurring matrix and L is a discrete differential operator, the condition (6) generally is satisfied. The regularization parameter $\mu > 0$ balances the relative influence of these terms on the solution.

We briefly comment on the choice of q. In many situations it is known that the desired solution (4) is sparse in some basis. To enhance sparsity, we may

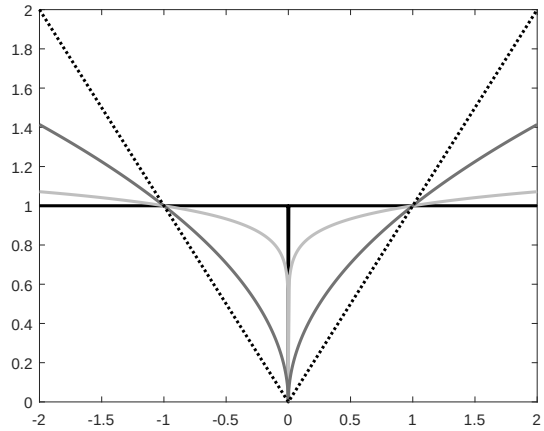


Figure 1: Comparison of different ℓ^q-norms. The solid black graph represents the ℓ^0-norm, the dotted black graph shows the ℓ^1-norm, the dark gray solid graph displays the $\ell^{0.5}$-norm, and the light gray solid graph depicts the $\ell^{0.1}$-norm.

consider using a regularization term with the ℓ^0-norm, which counts the number of non-vanishing entries. However, the minimization problem so obtained is very difficult to solve. Therefore, it is common to approximate the ℓ^0-norm by the ℓ^1-norm. The main advantage of using this approximation is that the ℓ^1-norm is convex. This makes the computation of a solution easier. However, ℓ^q-norms with $0 < q < 1$ are better approximations of the ℓ^0-norm. In particular, the smaller q, the better the approximation; see Figure 1 for an illustration. The main drawback of using $0 < q < 1$ is that the resulting minimization problem (5) is not convex; see Lanza et al. [26] for a recent discussion on the choice of q in the context of image restoration.

We turn to the choice of p. The value of p should depend on the type of noise in the data $\mathbf{b}^\delta$. For white Gaussian noise, $p = 2$ is appropriate and a method for determining μ for this kind of noise, based on the discrepancy principle is described in [2]. This method requires that a fairly accurate estimate of the norm of the noise be available and allows $0 < q < 1$. However, for impulse noise, $p = 2$ usually produces restorations of poor quality. It has been shown, see, e.g., [19, 25], that choosing $0 < p < 1$ in the fidelity term leads to accurate restorations in the case of salt-and-pepper noise.

A popular approach to solving the minimization problem (5) is to approximate the ℓ^p- and ℓ^q-norms by weighted ℓ^2-norms. The iterative refinement of these approximations leads to a solution process known as the iteratively reweighted norm (IRN) method. This method proceeds by solving a sequence of weighted least-squares problems until an accurate approximate solution of (5) has been found. Several implementations are available; see, e.g., [8, 10, 14, 25, 29, 32]. Applications of the IRN method to minimization problems (5) with p or q smaller than unity are described in [19, 25]. It is shown in [19] that

the solutions of the sequence of weighted least-squares problems converge to a stationary point of the functional (5).

We use the IRN-method FMM-GKS described in [19] for solving (5), and will refer to it simply as the MM-GKS method. Each iteration with MM-GKS can be divided into two steps: The first step majorizes the functional to be minimized in (5) by a quadratic functional that is tangent to the functional at the current approximation. Then, in the second step, the unique minimizer of the majorant is computed and used as the new iterate.

None of the works on solution methods for (5) mentioned discuss how a suitable value of the regularization parameter μ can be determined automatically, i.e., without user interaction. The value of μ affects the quality of the computed solution, and it is important to develop techniques for determining a suitable value. It is the purpose of the present paper to describe two algorithms for determining μ. Both algorithms are based on cross-validation (CV); see, e.g., Stone [31] for a discussion on cross-validation. Here we only note that CV is a so-called heuristic parameter choice rule and, therefore, may fail for certain data; see, Engl et al. [12] and Kindermann [21, 22], as well as [28], for discussions on and illustrations of heuristic methods. In numerous numerical experiments with applications to image restorations, some of which are reported in Section 4, we have never observed CV to fail to determine a useful value of the regularization parameter.

In our first algorithm for determining a suitable value of μ, we apply the CV technique to the data, i.e., to the vector $\mathbf{b}^\delta$. CV splits the data into two complementary sets: the training set and the testing set. The first set is used for solving the problem with different regularization parameters. Then the second set is used to validate the regularization parameter. CV selects the parameter μ that minimizes the difference between the reconstructed data set and the testing set.

Our second algorithm for determining a suitable value of μ uses a modified version of the CV approach outlined above. Instead of seeking to reconstruct the right-hand side, the algorithm applies CV to the computed solutions. To the best of our knowledge, this modified CV approach, henceforth referred to as MCV, has not been considered before. We will show that both CV and MCV determine regularization parameters that yield restorations of good quality and that, typically, MCV determines regularization parameters that give restorations of higher quality than CV.

Several other methods have been developed for the restoration of images that are corrupted by blur and impulse noise. In particular, two-phase strategies have been shown to yield accurate restorations; see, e.g., [3, 6, 7, 30]. These methods first identify pixels that are contaminated by impulse noise by means of a median-type filter. Subsequently, these pixels are removed from the computations and the noise-free problem so obtained is solved by a variational method. In the current literature on two-phase methods, the functional to be minimized in the second phase is usually convex and little attention is given to the selection of the regularization parameter. In fact, only Sciacchitano et al. [30] propose a two-phase method that does not require a user to specify a regularization

parameter. None of the methods mentioned is designed to remove mixed noise, i.e., noise that is made up of both impulse noise and Gaussian noise. One of the main advantages of the method of this paper is that, as we will show in Section 4, it is able to restore images that have been contaminated by mixed noise in a satisfactory manner. This type of noise is of considerable interest, and the choice of a suitable value of the regularization parameter is important.

This paper is organized as follows: Section 2 outlines the IRN method described in [19] for the solution of (5). Our algorithms for determining the regularization parameter μ are described in Section 3 and a few numerical example are presented in Section 4. Finally, Section 5 contains concluding remarks.

2. A majorization-minimization method

This section briefly describes the method proposed in [19]. This is an alternating direction method of multipliers (ADMM); see, e.g., Beck [1, Chapter 15] for a general discussion of this kind of methods: In one "direction" a minimization problem is solved by computing an approximate solution of the associated normal equations (eq. (10) below). This requires significant computations. In the other "direction" the weights (see eq. (9)) are updated. This is very inexpensive. The method is well suited for the solution of problems of the form (5), because only the solution in one "direction" requires significant computations; see below or [19, 25]. We remark that there also are other methods available for the minimization of (5), in particular for the situation when $p, q \geq 1$; see, e.g., [11, 13, 14]. In the present paper, we are primarily concerned with the case when both $p, q < 1$ in (5), though the method described also can be applied when $1 \leq p < 2$ or $1 \leq q < 2$. The following description is very similar to the one provided in [2, 19]. We present it here for the convenience of the reader.

Assume that $0 < s \leq 1$, because the smoothing to be described is not required for $s > 1$. In all computed examples of Section 4, we have $0 < s < 1$. Introduce a smoothed version of the function $\mathbf{x} \to \|\mathbf{x}\|_s^s$ as follows. Consider the function $\Phi_s : \mathbb{R} \to \mathbb{R}$ defined by

$$\Phi_s(t) = |t|^s.$$

If $0 < s \leq 1$, then Φ_s is not differentiable at 0. Therefore, we define the smoothed version of Φ_s as

$$\Phi_{s,\varepsilon}(t) = \left(\sqrt{t^2 + \varepsilon^2}\right)^s, \tag{8}$$

where $\varepsilon > 0$ is a small constant. Clearly, $\Phi_{s,\varepsilon}(t)$ is everywhere differentiable. A smoothed version of $\|\mathbf{x}\|_s^s$ for $\mathbf{x} = [x_1, \ldots, x_n]^t \in \mathbb{R}^n$ is given by the right-hand side of

$$\|\mathbf{x}\|_s^s \approx \sum_{i=1}^{n} \Phi_{s,\varepsilon}(x_i).$$

Throughout this paper, the superscript ^{t} denotes transposition.

Introduce the smoothed version of the functional that is minimized in (5),

$$\mathcal{J}_\varepsilon(\mathbf{x}) := \frac{1}{p}\sum_{i=1}^{m} \Phi_{p,\varepsilon}((A\mathbf{x}-\mathbf{b}^\delta)_i) + \frac{\mu}{q}\sum_{i=1}^{\ell} \Phi_{q,\varepsilon}((L\mathbf{x})_i).$$

The smoothed minimization problem associated with (5) reads

$$\mathbf{x}^* := \arg\min_{\mathbf{x}\in\mathbb{R}^n} \mathcal{J}_\varepsilon(\mathbf{x}).$$

The method described in [19] for computing a stationary point of $\mathcal{J}_\varepsilon$ is a majorization-minimization method. It determines a sequence of iterates $\mathbf{x}^{(k)}$, $k = 1, 2, \dots$, that converge to a stationary point of $\mathcal{J}_\varepsilon$. The method requires the gradient $\nabla\mathcal{J}_\varepsilon$ to exist. This is secured by the smoothing described.

At each step the functional $\mathcal{J}_\varepsilon$, for $k = 1, 2, \dots$, is majorized by a quadratic functional $\mathbf{x} \to \mathcal{Q}(\mathbf{x}, \mathbf{x}^{(k)})$ that is tangent to $\mathcal{J}_\varepsilon$ at $\mathbf{x}^{(k)}$. The next iterate $\mathbf{x}^{(k+1)}$ is the unique minimizer of $\mathbf{x} \to \mathcal{Q}(\mathbf{x}, \mathbf{x}^{(k)})$. We outline this method in the remainder of this section.

Definition 1. *Consider the differentiable function $\mathcal{J}_\varepsilon(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$. We say that the function $\mathbf{x} \to \mathcal{Q}(\mathbf{x}, \mathbf{y}) : \mathbb{R}^n \to \mathbb{R}$ is a* quadratic tangent majorant *for $\mathcal{J}_\varepsilon(\mathbf{x})$ at $\mathbf{y}$ if the following conditions hold:*

- *$\mathcal{Q}(\mathbf{x}, \mathbf{y})$ is quadratic;*
- *$\mathcal{Q}(\mathbf{x}, \mathbf{y}) \geq \mathcal{J}_\varepsilon(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^n$;*
- *$\mathcal{Q}(\mathbf{y}, \mathbf{y}) = \mathcal{J}_\varepsilon(\mathbf{y})$ and $\nabla\mathcal{Q}(\mathbf{y}, \mathbf{y}) = \nabla\mathcal{J}_\varepsilon(\mathbf{y})$.*

2.1. Majorization step

We outline the construction of a quadratic tangent majorant at the point $\mathbf{x}^{(k)}$. Two approaches are described in [19], one yields a majorant with fixed aperture and the other one a majorant with the largest aperture possible. The second kind of majorant approximates the function $\mathcal{J}_\varepsilon$ better than the first kind, but its computation is more demanding. In the following, we will only consider the majorant with fixed aperture, and we will apply this method in our numerical examples reported in Section 4. We note, however, that we could equally well have applied the method that determines the largest aperture possible in the computed examples. A comparison of these methods is reported in [19].

Let

$$\begin{aligned}\mathbf{v}^{(k)} &:= A\mathbf{x}^{(k)} - \mathbf{b}^\delta,\\ \mathbf{u}^{(k)} &:= L\mathbf{x}^{(k)},\end{aligned}$$

and introduce the vectors

$$\begin{aligned}\boldsymbol{\omega}_{\text{fid}}^{(k)} &:= \mathbf{v}^{(k)}\left(1 - \left(\frac{(\mathbf{v}^{(k)})^2 + \varepsilon^2}{\varepsilon^2}\right)^{p/2-1}\right),\\ \boldsymbol{\omega}_{\text{reg}}^{(k)} &:= \mathbf{u}^{(k)}\left(1 - \left(\frac{(\mathbf{u}^{(k)})^2 + \varepsilon^2}{\varepsilon^2}\right)^{q/2-1}\right),\end{aligned} \tag{9}$$

where all operations are element-wise. It is shown in [19] that the function

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

with c a suitable constant independent of $\mathbf{x}$, is a quadratic tangent majorant for $\mathcal{J}_\varepsilon$ at $\mathbf{x}^{(k)}$.

2.2. Minimization step

Given $\mathbf{x}^{(k)}$, the next iterate $\mathbf{x}^{(k+1)}$ is the minimizer of $\mathbf{x} \to \mathcal{Q}(\mathbf{x}, \mathbf{x}^{(k)})$. Since $\mathcal{Q}$ is quadratic, $\mathbf{x}^{(k+1)}$ can be computed by determining the zero of the gradient, i.e., by solving the linear system of equations

$$(A^t A + \eta L^t L)\mathbf{x}^{(k+1)} = A^t(\mathbf{b}^{\delta} + \boldsymbol{\omega}_{\text{fid}}^{(k)}) + \eta L^t \boldsymbol{\omega}_{\text{reg}}^{(k)}, \quad \eta := \mu\frac{\varepsilon^{q-2}}{\varepsilon^{p-2}}. \tag{10}$$

The matrix on the left-hand side is non-singular for any $\mu > 0$ due to the requirement (6). Therefore $\mathbf{x}^{(k+1)}$ is the unique minimizer of $\mathcal{Q}(\mathbf{x}, \mathbf{x}^{(k)})$.

An approximate solution of (10) can be computed efficiently by seeking a solution in a low-dimensional subspace. Let the columns of $V_k \in \mathbb{R}^{n\times d_k}$, with $1 \le d_k \ll n$, form an orthonormal basis for the subspace in which we determine the next approximate solution $\mathbf{x}^{(k+1)}$ of (10). We compute $\mathbf{x}^{(k+1)}$ by solving the minimization problem

$$\mathbf{y}^{(k+1)} := \arg\min_{\mathbf{y}\in\mathbb{R}^{d_k}} \left\| \begin{bmatrix} AV_k \\ \eta^{1/2} L V_k \end{bmatrix} \mathbf{y} - \begin{bmatrix} \mathbf{b}^{\delta} + \boldsymbol{\omega}_{\text{fid}}^{(k)} \\ \eta^{1/2}\boldsymbol{\omega}_{\text{reg}}^{(k)} \end{bmatrix} \right\|_2^2 \tag{11}$$

and letting

$$\mathbf{x}^{(k+1)} := V_k \mathbf{y}^{(k+1)}. \tag{12}$$

Introduce the QR factorizations

$$\begin{array}{rclll} AV_k &=& Q_A R_A & \text{with} & Q_A \in \mathbb{R}^{m\times d}, \quad R_A \in \mathbb{R}^{d\times d}, \\ LV_k &=& Q_L R_L & \text{with} & Q_L \in \mathbb{R}^{\ell\times d}, \quad R_L \in \mathbb{R}^{d\times d}. \end{array} \tag{13}$$

Thus, the matrices Q_A and Q_L have orthonormal columns, and the matrices R_A and R_L are upper triangular. Inserting the factorizations (13) into (11) yields

$$\mathbf{y}^{(k+1)} := \arg\min_{\mathbf{y}\in\mathbb{R}^{d_k}} \left\| \begin{bmatrix} R_A \\ \eta^{1/2} R_L \end{bmatrix} \mathbf{y} - \begin{bmatrix} Q_A^t(\mathbf{b}^{\delta} + \boldsymbol{\omega}_{\text{fid}}^{(k)}) \\ \eta^{1/2} Q_L^t \boldsymbol{\omega}_{\text{reg}}^{(k)} \end{bmatrix} \right\|_2^2,$$

and substituting (12) into (10) gives the residual vector

$$\mathbf{r} := A^t(AV_k\mathbf{y}^{(k+1)} - \mathbf{b}^{\delta} - \boldsymbol{\omega}_{\text{fid}}^{(k)}) + \eta L^t(LV_k\mathbf{y}^{(k+1)} - \boldsymbol{\omega}_{\text{reg}}^{(k)}). \tag{14}$$

We expand the solution subspace by including the scaled residual vector $\mathbf{v}_{\rm new} = \mathbf{r}/\|\mathbf{r}\|$ in the solution subspace. This vector is orthogonal to the columns of the matrix V_k, and we define the new matrix $V_{k+1} = [V_k, \mathbf{v}_{\rm new}] \in \mathbb{R}^{n\times d_{k+1}}$, $d_{k+1} = d_k + 1$, whose columns form an orthonormal basis for the expanded solution subspace. The so determined solution subspaces are referred to as *generalized Krylov subspaces*. A related application of generalized Krylov subspaces is described in [24].

In addition to storing the matrix V_{k+1}, we also store the matrices

$$AV_{k+1} = [AV_k, A\mathbf{v}_{\rm new}], \quad LV_{k+1} = [LV_k, L\mathbf{v}_{\rm new}].$$

The QR factorizations of these matrices are computed by updating the QR factorizations (13) according to

$$AV_{k+1} = [AV_k, A\mathbf{v}_{\rm new}] = [Q_A, \tilde{\mathbf{q}}_A] \begin{bmatrix} R_A & \mathbf{r}_A \\ \mathbf{0}^t & \tau_A \end{bmatrix},$$
$$LV_{k+1} = [LV_k, L\mathbf{v}_{\rm new}] = [Q_L, \tilde{\mathbf{q}}_L] \begin{bmatrix} R_L & \mathbf{r}_L \\ \mathbf{0}^t & \tau_L \end{bmatrix},$$

where

$$\begin{array}{rclcrcl} \mathbf{r}_A &=& Q_A^t(A\mathbf{v}_{\rm new}), & & \mathbf{q}_A &=& A\mathbf{v}_{\rm new} - Q_A\mathbf{r}_A, \\ \tau_A &=& \|\mathbf{q}_A\|_2\,, & & \tilde{\mathbf{q}}_A &=& \mathbf{q}_A/\tau_A, \\ \mathbf{r}_L &=& Q_L^t(L\mathbf{v}_{\rm new}), & & \mathbf{q}_L &=& L\mathbf{v}_{\rm new} - Q_L\mathbf{r}_L, \\ \tau_L &=& \|\mathbf{q}_L\|_2\,, & & \tilde{\mathbf{q}}_L &=& \mathbf{q}_L/\tau_L; \end{array}$$

see Daniel et al. [9] for details.

Algorithm 1 summarizes the computations. To initiate the computations a user chooses a k_0-dimensional solution subspace of $\mathbb{R}^n$. The columns of the matrix V_0 form an orthonormal basis for this subspace. Thus, $d_k = k_0 + k$. In the computations reported in Section 4, we let $k_0 = 10$ and let the columns of V_0 form an orthonormal basis for the Krylov subspace

$$\mathcal{K}_{10}(A^tA, A^t\mathbf{b}^\delta) = \text{span}\{A^t\mathbf{b}^\delta, (A^tA)A^t\mathbf{b}^\delta, \ldots, (A^tA)^9A^t\mathbf{b}^\delta\}.$$

This subspace is determined by carrying out 10 steps of the Golub–Kahan bidiagonalization algorithm applied to A with initial vector $\mathbf{b}^\delta$; see, e.g., [23]. Algorithm 1 summarizes the computations.

Algorithm 1 (The MM-GKS method)**.** *Let $0 < p, q \le 2$ and $\mu > 0$. Consider $A \in \mathbb{R}^{m\times n}$ and $L \in \mathbb{R}^{\ell\times n}$ such that* (6) *holds. Fix $\varepsilon > 0$ and $k_0 > 0$, and choose*

the initial vector $\mathbf{x}_0$*;*

Generate the initial subspace basis: $V_0 \in \mathbb{R}^{n\times k_0}$ *such that* $V_0^t V_0 = I$*;*
Compute and store AV_0 *and* LV_0*;*
Compute the QR factorizations $AV_0 = Q_A R_A$ *and* $LV_0 = Q_L R_L$*;*
$\eta = \mu \frac{\varepsilon^{q-2}}{\varepsilon^{p-2}}$*;* $\mathbf{y}^{(0)} = V_0^t \mathbf{x}^{(0)}$*;*
for $k = 0, 1, \ldots$ **do**
$\mathbf{v}^{(k)} = A\mathbf{x}^{(k)} - \mathbf{b}^{\delta}$*;*
$\mathbf{u}^{(k)} = LV_k \mathbf{y}^{(k)}$*;*
$\boldsymbol{\omega}_{\text{fid}}^{(k)} = \mathbf{v}^{(k)} \left(1 - \left(\frac{(\mathbf{v}^{(k)})^2+\varepsilon^2}{\varepsilon^2}\right)^{p/2-1}\right)$*;*
$\boldsymbol{\omega}_{\text{reg}}^{(k)} = \mathbf{u}^{(k)} \left(1 - \left(\frac{(\mathbf{u}^{(k)})^2+\varepsilon^2}{\varepsilon^2}\right)^{q/2-1}\right)$*;*
$\mathbf{y}^{(k+1)} = (R_A^t R_A + \eta R_L^t R_L)^{-1}(R_A^t Q_A^t(\mathbf{b}^{\delta} + \boldsymbol{\omega}_{\text{fid}}^{(k)}) + \eta R_L^t Q_L^t \boldsymbol{\omega}_{\text{reg}}^{(k)})$*;*
$\mathbf{r} = A^t(AV_k\mathbf{y}^{(k+1)} - \mathbf{b}^{\delta} - \boldsymbol{\omega}_{\text{fid}}^{(k)}) + \eta L^t(LV_k\mathbf{y}^{(k+1)} - \boldsymbol{\omega}_{\text{reg}}^{(k)})$*;*
$\mathbf{v}_{\text{new}} = \mathbf{r}/\|\mathbf{r}\|_2$*;* $V_{k+1} = [V_k, \mathbf{v}_{\text{new}}]$*;*
Update the QR factorizations $AV_{k+1} = Q_A R_A$ *and* $LV_{k+1} = Q_L R_L$*;*
$\mathbf{x}^{(k+1)} = V_k \mathbf{y}^{(k+1)}$*;*
end

Let $\mathbf{x}^*$ be the approximated solution computed by Algorithm 1 we will write

$$\mathbf{x}^* = \text{MM-GKS}(A, \mathbf{b}^{\delta}, p, q, \mu, \varepsilon, \mathbf{x}_0, V_0),$$

where we expressly state that the initial approximate solution is $\mathbf{x}_0$, and that the initial subspace basis is V_0.

We summarize properties of the approximate solutions determined by the algorithm. The range of a matrix M is denoted by $\mathcal{R}(M)$.

Proposition 2. *We have* $\mathbf{x}^{(0)} \in \mathcal{R}(A^T)$ *and* $\mathbf{x}^{(k)} \in \mathcal{R}(A^T) \cup \mathcal{R}(L)$ *for* $k = 1, 2, \ldots$.

Proof. The statement about $\mathbf{x}^{(0)}$ follows from the definition of the vector. The property of $\mathbf{x}^{(k)}$ is a consequence of (14) and the fact that the matrix L, defined by (7), is symmetric. □

In our applications of the minimization problem (5), the matrix-vector product $A\mathbf{x}$ models blurring of $\mathbf{x}$. Therefore, $A\mathbf{x}$ represents the discretization of a function that typically is quite smooth. This also holds for $A^T\mathbf{x}$. The vector $\mathbf{x}^{(0)}$ in Algorithm 1 therefore, generally, represents the discretization of a smooth function and is poorly suited to model a piece-wise smooth image. The vectors $\mathbf{x}^{(k)}$, $k = 1, 2, \ldots$, generated by the algorithm generally have a component in $\mathcal{R}(L)$. Since L is a discrete differential operator, the vector $L\mathbf{x}$ generally represents the discretization of a function that is less smooth than the function whose discretization gives $\mathbf{x}$. The component in $\mathcal{R}(L)$ of the vectors $\mathbf{x}^{(k)}$, $k \geq 1$, makes these vectors useful for approximating piece-wise smooth functions. Thus, having a regularization term defined by a discrete differential operator in (5) is essential for the good performance of Algorithm 1.

The main computational effort of Algorithm 1 is the evaluation of matrix-vector products with the matrices A, L, and their transposes. Since the matrices AV_k and LV_k are stored, each iteration requires the evaluation of the matrix-vector products $A\mathbf{v}_{k+1}$ and $L\mathbf{v}_{k+1}$, which are needed for updating AV_k and LV_k, and of their QR factorizations. The evaluation of a matrix-vector product with each one of the matrices A^t and L^t is required when computing the residual vector (14).

The following result for the approximate solutions $\mathbf{x}^{(k)}$ computed by Algorithm 1 is shown in [19].

Theorem 3. *Let* (6) *hold. Then for any initial approximate solution* $\mathbf{x}^{(0)} \in \mathbb{R}^n$, *the sequence* $\{\mathbf{x}^{(k)}\}_k$ *converges to a stationary point of* $\mathcal{J}_\varepsilon(\mathbf{x})$. *Thus,*

(i) $\lim_{k\to\infty} \left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\|_2 = 0$,

(ii) $\lim_{k\to\infty} \nabla\mathcal{J}_\varepsilon(\mathbf{x}^{(k)}) = \mathbf{0}$.

3. Determining the regularization parameter

This section describes two CV methods for determining the regularization parameter μ. The first method applies CV to the data vector and the second one to the computed solution. For notational simplicity, we describe these two methods when applied to a Tikhonov regularization problem in standard form,

$$\min_{\mathbf{x}\in\mathbb{R}^n} \{\|A\mathbf{x} - \mathbf{b}^\delta\|_2^2 + \mu\|\mathbf{x}\|_2^2\}, \tag{15}$$

where $A \in \mathbb{R}^{m\times n}$, $\mathbf{b}^\delta \in \mathbb{R}^m$, and $\mu > 0$ is a regularization parameter. We assume that A is very ill-conditioned, that $\mathbf{b}^\delta$ is contaminated by error, and that we are interested in the solution (4).

3.1. Cross validation

The CV method partitions the right-hand side $\mathbf{b}^\delta$ into two complementary subsets: the training set and the testing set. The training set is used for solving the problem (15) (with the rows of the testing set removed) for different regularization parameters, and the testing set is used to validate the computed solution and select a suitable regularization parameter. The computations with the CV method proceeds as follows. Without loss of generality, we may assume that the testing set consists of the first d elements of $\mathbf{b}^\delta$. Let $\widetilde{\mathbf{b}}^\delta \in \mathbb{R}^{m-d}$ and $\widetilde{A} \in \mathbb{R}^{(m-d)\times n}$ denote the restrictions of $\mathbf{b}^\delta = [b_j^\delta]$ and $A = [A_{ij}]$, respectively, in (15) to the training set, i.e.,

$$\widetilde{\mathbf{b}}^\delta = [b_{d+1}^\delta, b_{d+2}^\delta, \ldots, b_m^\delta]^t,$$

$$\widetilde{A} = \begin{bmatrix} A_{d+1,1} & A_{d+1,2} & \ldots & A_{d+1,n} \\ A_{d+2,1} & A_{d+2,2} & \ldots & A_{d+2,n} \\ \vdots & \vdots & \ldots & \vdots \\ A_{m,1} & A_{m,2} & \ldots & A_{m,n} \end{bmatrix}.$$

Let $\{\mu_j\}_{j=1}^l$ denote a set of positive regularization parameters. For each $j = 1, 2, \ldots, l$, we solve the Tikhonov regularization problem obtained by replacing A by $\widetilde{A}$, $\mathbf{b}^\delta$ by $\widetilde{\mathbf{b}}^\delta$, and μ by μ_j in (15). Denote the computed solutions by $\mathbf{x}_{\mu_j}$, $j = 1, 2, \ldots, l$. We validate these solutions by using the testing set. Thus, for each $\mathbf{x}_{\mu_j}$, we compute the residual norms

$$r_j = \sqrt{\sum_{i=1}^{d} \left(\left(A\mathbf{x}_{\mu_j} \right)_i - b_i^\delta \right)^2}, \quad j = 1, 2, \ldots, l.$$

The norm r_j measure how well the computed solution $\mathbf{x}_{\mu_j}$ is able to predict the testing data (which were not included when computing $\mathbf{x}_{\mu_j}$). Let μ_k be such that $r_k \leq r_j$ for $j = 1, 2, \ldots, l$. We then select $\mu = \mu_k$ for the solution of (15). The heuristics behind this approach for determining μ is that an accurate approximation $\mathbf{x}_\mu$ of $\widehat{\mathbf{x}}$ should be able to predict the testing data accurately.

To reduce variability, we apply CV for several different partitionings and average the regularization parameter values determined for each partitioning. In detail, let K be a not too large positive integer and carry out K CV steps. At step $1 \leq k \leq K$, we consider a randomly selected set of d components of the vector $\mathbf{b}^\delta$ as testing data, while the other $m-d$ components are used as training data. Each step provides a regularization parameter $\mu^{(k)}$ for $k = 1, 2, \ldots, K$. These parameters may differ. The regularization parameter μ to be used for the solution of (15) is the average of the parameter values $\mu^{(k)}$, i.e.,

$$\mu = \frac{1}{K} \sum_{k=1}^{K} \mu^{(k)}.$$

The process is summarized in the following algorithm.

Algorithm 2 (Cross Validation)**.** *Let $A \in \mathbb{R}^{m \times n}$, $d < m$, and let $K > 0$ be a positive integer. Consider the solution of* (15) *and let $\{\mu_j\}_{j=1}^l$ be a set of*

positive regularization parameters.

for $k = 1, 2, \ldots, K$ **do**

Construct a set $I^{(k)}$ *of* d *distinct random integers between* 1 *and* n*;*

Let $\widetilde{A}$ *and* $\widetilde{\mathbf{b}}^{\delta}$ *denote the matrix and vector, respectively, obtained by removing the rows with indices in* $I^{(k)}$ *from* A *and* $\mathbf{b}^{\delta}$*;*

for $j = 1, 2, \ldots, l$ **do**

$\mathbf{x}_{\mu_j}^{(k)} =$ *MM-GKS*$(\widetilde{A}, \widetilde{\mathbf{b}}^{\delta}, p, q, \mu, \varepsilon, \mathbf{x}_{k,j}^{\text{init}}, V_{k,j}^{\text{init}})$,

i.e., $\mathbf{x}_{\mu_j}^{(k)}$ *denotes the regularized solution of the system* (15) *with the matrix* A *replaced by* $\widetilde{A}$*,* $\mathbf{b}^{\delta}$ *replaced by* $\widetilde{\mathbf{b}}^{\delta}$*, and* μ *replaced by* μ_j*;*

Compute $r_j^{(k)} = \sqrt{\sum_{i \in I^{(k)}} \left(\left(A\mathbf{x}_{\mu_j}^{(k)} \right)_i - b_i^{\delta} \right)^2}$*;*

end

Let $j^* = \arg\min_{1 \le j \le l} \{ r_j^{(k)} \}$*;*

Let $\mu^{(k)} = \mu_{j^*}$*;*

end

Compute $\mu = \frac{1}{K} \sum_{k=1}^{K} \mu^{(k)}$*;*

3.2. Modified cross-validation

The standard CV technique compares predictions of the right-hand side determined for different parameters μ_j. We would like to exploit a similar idea, but instead compare predictions of computed solutions.

Let I_1 and I_2 denote two distinct sets of d distinct random integers between 1 and n. Similarly to standard CV, let $\widetilde{A}_i$ and $\widetilde{\mathbf{b}}_i^{\delta}$ denote versions of the matrix and right-hand side of (15), respectively, in which the rows of $\mathbf{b}^{\delta}$ and A with index in I_i have been removed, for $i = 1, 2$.

Let $\{\mu_j\}_{j=1}^{l}$ be a set of positive regularization parameters. For $i = 1, 2$, let $\mathbf{x}_{\mu_j}^{(i)}$ denote the solution of the Tikhonov regularization problem (15) with A replaced by $\widetilde{A}_i$, $\mathbf{b}^{\delta}$ replaced by $\widetilde{\mathbf{b}}_i^{\delta}$, and μ replaced by μ_j, i.e.,

$$\mathbf{x}_{\mu_j}^{(i)} = \arg\min_{\mathbf{x} \in \mathbb{R}^n} \left\{ \left\| \widetilde{A}_i \mathbf{x} - \widetilde{\mathbf{b}}_i^{\delta} \right\|_2^2 + \mu_j \left\| \mathbf{x} \right\|_2^2 \right\}, \quad i = 1, 2.$$

Compute the quantities

$$\Delta x_j = \left\| \mathbf{x}_{\mu_j}^{(1)} - \mathbf{x}_{\mu_j}^{(2)} \right\|_2, \qquad j = 1, 2, \ldots, l,$$

and let μ_k minimize Δx_j over $j = 1, 2, \ldots, l$. Thus, this application of CV considers the computed approximations of $\widehat{\mathbf{x}}$ instead of the residual vectors. We refer to this method as modified CV (MCV). To reduce variability, we apply MCV for several index sets I_1 and I_2. The following algorithm describes MCV.

Algorithm 3 (Modified Cross Validation)**.** *Let $A \in \mathbb{R}^{m\times n}$, $d < m$, and let $K > 0$ be an integer. Consider the solution of* (15) *and let $\{\mu_j\}_{j=1}^{l}$ be a set of positive regularization parameters. Let $K \in \mathbb{N}$.*

for $k = 1, 2, \ldots, K$ **do**
  Construct two distinct sets $I_1^{(k)}$ and $I_2^{(k)}$ of d distinct random integers between 1 and n;
  Let, for $i = 1, 2$, $\widetilde{A}_i$ and $\widetilde{\mathbf{b}}_i^{\delta}$ denote the matrix and vector, respectively, obtained by removing the rows with indices in $I_i^{(k)}$ from A and $\mathbf{b}^{\delta}$;
  for $j = 1, 2, \ldots, l$ **do**
    for $i = 1, 2$ **do**
      $\mathbf{x}_{\mu_j}^{(k,i)} = \text{MM-GKS}(\widetilde{A}_i, \widetilde{\mathbf{b}}_i^{\delta}, p, q, \mu, \varepsilon, \mathbf{x}_{k,j,i}^{\text{init}}, V_{k,j,i}^{\text{init}})$,
      i.e., $\mathbf{x}_{\mu_j}^{(k,i)}$ denotes the regularized solution of the system (15) *with the matrix A replaced by $\widetilde{A_i}$, $\mathbf{b}^{\delta}$ replaced by $\widetilde{\mathbf{b}}_i^{\delta}$, and μ replaced by μ_j;*
    end
    Compute $\Delta x_j^{(k)} = \left\| \mathbf{x}_{\mu_j}^{(k,1)} - \mathbf{x}_{\mu_j}^{(k,2)} \right\|_2$;
  end
  Let $j^* = \arg\min_{1\le j\le l} \left\{ \Delta x_j^{(k)} \right\}$;
  Let $\mu^{(k)} = \mu_{j^*}$;
end
Compute $\mu = \frac{1}{K} \sum_{k=1}^{K} \mu^{(k)}$;

3.3. Implementation details

The execution of Algorithms 2 and 3 requires that computations with Algorithm 1 be carried out multiple times. These executions can be carried out in parallel, but nevertheless may be fairly expensive. This subsection describes some implementation details and discusses certain implementation choices.

In our implementation for all k, j, and i in both the CV and MCV methods, we set the initial approximate solution to $\widetilde{A}_i^t \widetilde{\mathbf{b}}_i^{\delta}$, i.e., for the CV algorithm we set

$$\mathbf{x}_{k,j}^{\text{init}} = \widetilde{A}^t \widetilde{\mathbf{b}}^{\delta}, \quad j = 1 \ldots, l \text{ and } k = 1, \ldots, K,$$

while for the MCV algorithm we set

$$\mathbf{x}_{k,j,i}^{\text{init}} = \widetilde{A}_i^t \widetilde{\mathbf{b}}_i^{\delta}, \quad j = 1, \ldots, l,\ k = 1, \ldots, K, \text{ and } i = 1, 2.$$

Moreover, we set the initial subspace basis as $\widetilde{A}_i^t \widetilde{\mathbf{b}}_i^{\delta} / \left\| \widetilde{A}_i^t \widetilde{\mathbf{b}}_i^{\delta} \right\|_2$, i.e., for the CV algorithm we set

$$V_{k,j}^{\text{init}} = \widetilde{A}^t \widetilde{\mathbf{b}}^{\delta}, \quad j = 1 \ldots, l \text{ and } k = 1, \ldots, K,$$

while for the MCV algorithm we set

$$V_{k,j,i}^{\text{init}} = \widetilde{A}_i^t \widetilde{\mathbf{b}}_i^\delta / \left\| \widetilde{A}_i^t \widetilde{\mathbf{b}}_i^\delta \right\|_2, \quad j = 1, \ldots, l, \ k = 1, \ldots, K, \text{ and } i = 1, 2.$$

This choices allow all the runs of the MM-GKS method to be independent and, therefore, in parallel.

The first aspect which we would like to discuss is the choice of the initial approximate solution. We remark that we also considered other initial approximate solutions. We also implemented the options:

- For a fixed k, we set the initial approximation for the computation of $\mathbf{x}_{\mu_j}^{(k)}$ to $\mathbf{x}_{\mu_{j-1}}^{(k)}$, i.e. for the CV algorithm

$$\mathbf{x}_{k,j}^{\text{init}} = \begin{cases} \widetilde{A}^t \widetilde{\mathbf{b}}^\delta & j = 1, \\ \mathbf{x}_{\mu_{j-1}}^{(k)} & j > 1, \end{cases} \qquad k = 1 \ldots, K,$$

  and for the MCV algorithm

$$\mathbf{x}_{k,j,i}^{\text{init}} = \begin{cases} \widetilde{A}_i^t \widetilde{\mathbf{b}}_i^\delta & j = 1, \\ \mathbf{x}_{\mu_{j-1}}^{(k,i)} & j > 1, \end{cases} \qquad k = 1 \ldots, K \text{ and } i = 1, 2.$$

- For a fixed j, we set the initial approximate solution for the computation of $\mathbf{x}_{\mu_j}^{(k)}$ to $\mathbf{x}_{\mu_j}^{(k-1)}$, i.e. for the CV algorithm

$$\mathbf{x}_{k,j}^{\text{init}} = \begin{cases} \widetilde{A}^t \widetilde{\mathbf{b}}^\delta & k = 1, \\ \mathbf{x}_{\mu_j}^{(k-1)} & k > 1, \end{cases} \qquad j = 1 \ldots, l,$$

  and for the MCV algorithm

$$\mathbf{x}_{k,j,i}^{\text{init}} = \begin{cases} \widetilde{A}_i^t \widetilde{\mathbf{b}}_i^\delta & k = 1, \\ \mathbf{x}_{\mu_j}^{(k-1,i)} & k > 1, \end{cases} \qquad j = 1 \ldots, l \text{ and } i = 1, 2.$$

Both these initial approximate solutions lead to poor numerical results. These choices do not allow enough variability in the computed solutions. The comparison of the computed solutions therefore is not meaningful. In the first case, μ_1 is always chosen as regularization parameter, while in the second case we obtain that $\mu^{(1)} = \mu^{(2)} = \ldots = \mu^{(K)}$, i.e., all the sweeps over all the parameters μ_j provide the same result.

Another point that is worthwhile to mention is the selection of the solution subspace. Algorithms 2 and 3 determine different solution subspaces for each value of j, k, and i. These subspaces are constructed as described in Algorithm 1 and, thus, depend on the restricted operator $\widetilde{A}_i$ and the associated right-hand side $\widetilde{\mathbf{b}}_i^\delta$, as well as on the parameter μ_j. This construction, coupled with the update of the QR factorizations in Algorithm 1, can become expensive if many iterations have to be carried out. To speed up the computations with Algorithms 2 and 3, we considered two possible approaches to the selection of the solution subspaces. For ease of notation, we will only discuss cross validation, however, the results obtained with modified cross validation are similar.

- For a fixed j, let $V^{(j)}$ denote the subspace constructed by the MM-GKS method for $\mu = \mu_j$ and $k = 1$. We fix the solution subspace for $\mu = \mu_j$ and $k > 1$ as $V^{(j)}$, i.e., for each j we construct a solution subspace only for $k = 1$, and we "recycle" this subspace for the subsequent ks.
- We construct the subspace $\mathcal{K}_r(AA^t, A^t\mathbf{b}^\delta)$ with r sufficiently large, and use it as solution subspace for all j and k.

Both these approaches reduce the computational effort required by the CV method. However, none of them give satisfactory numerical results. In particular, the first choice of subspaces do not provide necessary additional information as k changes. In other words, $\mathbf{x}_{\mu_j}^{(k)} \approx \mathbf{x}_{\mu_j}^{(1)}$ for all k. This is due to the fact that the constructed subspace $V^{(j)}$ is of fairly small dimension. Thus, even if we slightly change the operator and right-hand side, as we do when we change k, the computed solution in such a small space is very close to the one obtained for $k = 1$. This implies that $\mu^{(1)} = \mu^{(2)} = \ldots = \mu^{(K)}$. We turn to the second approach, and first discuss the choice of r. Let $\mathbf{x}^*$ denote an approximate solution computed by Algorithm 1. By construction, we know that $\mathbf{x}^*$ belongs to a subspace of fairly small dimension, which we denote by $\hat{k}$. Then $\hat{k}$ is the sum of k_0, the dimension of the initial subspace, and the number of iterations carried out. However, looking at the magnitude of the coefficients of $\mathbf{x}^*$ in the basis of the subspace constructed by the MM-GKS method, we observe that, generally, only the first $\tilde{k} < 50$ components are significant; the other components are of very small magnitude. It is therefore reasonable to assume that, by choosing $r > 2\tilde{k}$, the space $\mathcal{K}_r(AA^t, A^t\mathbf{b}^\delta)$ should contain an accurate approximation of $\mathbf{x}^*$. We therefore set $r = 100$. Moreover, if $r > 100$ then the computational advantage that we obtain by fixing the space a priori would be less significant, since we still need to compute the QR factorizations of $\tilde{A}V_r$ for each k, where V_r a matrix whose columns form an orthonormal basis for $\mathcal{K}_r(AA^t, A^t\mathbf{b}^\delta)$ determined, e.g., by Golub–Kahan bidiagonalization. Our numerical experiments show that the MM-GKS method defined in this manner is not able to compute accurate reconstructions in $\mathcal{K}_r(AA^t, A^t\mathbf{b}^\delta)$ for reasonable values of r. This is possibly due to the fact that when we compute $\mathbf{x}_{\mu_j}^{(k)}$ we are not solving the system $A\mathbf{x} = \mathbf{b}^\delta$, but $\widetilde{A}\mathbf{x} = \widetilde{\mathbf{b}}^\delta$ and, thus, we are considering a subspace associated with a different problem.

We conclude from the above observations that, for both the cross validation and modified cross validation methods, it is of vital importance that all the runs of the methods are independent of each other.

Finally, we would like to point out that in the computations reported in the next section, we let $K = l = 10$ and $d = \left\lceil \frac{n}{200} \right\rceil$ in both Algorithm 2 and 3. The μ_j are equispaced in a logarithmic scale. These parameter choices are is not critical for the performance of the CV and MCV methods. Other values of K and l can be used without changing the results significantly.

4. Numerical examples

This section presents some numerical examples. All computations are carried out in MATLAB 2016a with about 15 significant decimal digits running on a laptop computer with a quad core CPU Intel Core i7-6700HQ @ 2.60GHz processor and 16 GB of RAM. We compare results obtained by computing approximate solutions with Algorithm 1 with the regularization parameter determined by Algorithm 2 or Algorithm 3, and will refer to these methods as MM-GKS-CV and MM-GKS-MCV, respectively. These methods are applied to image restoration problems and compared to restored images computed by Algorithm 1 with the optimal regularization parameter, i.e., the regularization parameter which yields a restored image that is closest to the desired solution $\widehat{\mathbf{x}}$. We refer to the latter restoration as the optimal restoration. The matrix A represents the blurring operator and the vector $\mathbf{b}^\delta$ represents the available blur- and noise-contaminated image. The entries of $\mathbf{b}^\delta$ are pixel values.

The error $\boldsymbol{\delta}$ in $\mathbf{b}^\delta$ models different types of noise. In the first examples, we consider salt-and-pepper noise of different percentages. Subsequently, we consider a mixture of salt-and-pepper noise and white Gaussian noise, as well as a mixture of impulse noise and white Gaussian noise. To model salt-and-pepper noise a certain percentage of randomly chosen entries of $\mathbf{b}$ are set to 0 (which corresponds to black) or to 255 (which is the maximum value attainable and corresponds to white). Impulse noise is modeled by letting a certain percentage of randomly chosen entries of $\mathbf{b}$ be randomly chosen uniformly distributed integers in the interval $[0, 255]$; cf. (2). We refer to this percentage as the *noise level* and denote it by σ. White Gaussian noise is modeled by pseudo-random numbers with zero mean and specified variance. We refer in this case to the ratio $\eta = \frac{\|\boldsymbol{\delta}\|_2}{\|\mathbf{b}\|_2}$ as the noise level.

Differentials of many natural images are known to be sparse. Therefore, we would like to compute a solution with this property. We use the second derivative operator in tensor form as regularization matrix L with the same boundary conditions as the matrix A; see [18] for more information about boundary conditions.

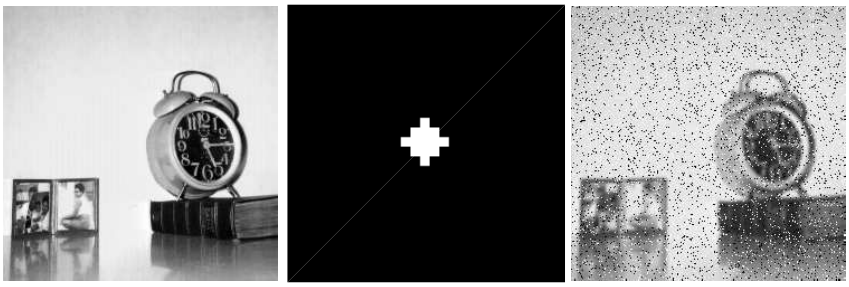
We let $\varepsilon = 1$ in (8). This value is small compared to the average size of the elements of the vector $\mathbf{b}^\delta$. The iterations with the algorithms are terminated as soon as two consecutive iterates are sufficiently close, i.e., as soon as

$$\frac{\left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k+1)}\right\|_2}{\left\|\mathbf{x}^{(k)}\right\|_2} \leq 10^{-4}.$$

The quality of the restored images is measured by the Peak Signal to Noise Ratio (PSNR), which is defined as

$$\mathrm{PSNR}(\mathbf{x}) = 20 \log_{10} \left(\frac{255\, n}{\|\mathbf{x} - \widehat{\mathbf{x}}\|_2} \right),$$

where $\widehat{\mathbf{x}}$ denotes the desired solution (4), n is the number of pixels in the image, and 255 is the largest possible pixel value.



(a) (b) (c)

Figure 2: Clock test image: (a) true image (246×246 pixels), (b) PSF (29×29 pixels), (c) blurred and noisy image (salt-and-pepper noise: $\sigma = 0.1$).

Clock.. In our first example, we consider the image in Figure 2(a) and blur it with the PSF shown in Figure 2(b), which models out-of-focus blur. To model salt-and-pepper noise, we set 10% of the pixels to either 0 or 255; see Figure 2(c) for the blurred and noisy image. This image is represented by the vector $\mathbf{b}^{\delta}$. We set $p = 0.8$ and $q = 0.1$. Since this is a generic image, we impose *reflexive* boundary conditions; see [27] for a discussion on these boundary conditions.

Figure 3 displays the restored images obtained with the MM-GKS-CV and MM-GKS-MCV methods, and by selecting the optimal μ, i.e., the one that maximizes the PSNR, by trial and error. By visual inspection, we can see that the two proposed approaches for determining the regularization parameter and computing a restoration provide restorations of high quality that are close to the one obtained with the optimal μ-value. This is confirmed by the PSNR-values reported in Table 1. The table shows that MCV (Algorithm 3) yields a more accurate restoration than CV (Algorithm 2).

Finally, Figure 4 displays the PSNR-value as a function of the regularization parameter μ. The figure shows the PSNR-values associated with regularization parameter values determined by the CV and MCV methods, as well as the PSNR-value that corresponds to the optimal μ-value. We can observe that the μ-value determined by MCV is closer to the optimal one than the value provided by CV.

Cameraman.. We use the image in Figure 5(a) and blur it with the non-symmetric Gaussian PSF shown in Figure 5(b), and add 20% salt-and-pepper noise; Figure 5(c) shows the blurred and noisy image. Similarly as in the previous example, we set $p = 0.8$ and $q = 0.1$, and impose reflexive boundary conditions.

Figure 6 shows the restoration obtained when μ is determined by the CV and MCV methods, or is chosen to minimize the PSNR. Table 1 reports the PSNR-values of these restorations. We note that the CV algorithm yields a regularization parameter μ that gives a worse restoration than the regularization parameter computed by MCV. The latter restoration is close in quality to the one achieved with the optimal μ.

(a) (b) (c)

Figure 3: Clock restorations: (a) Optimal μ, (b) CV, (c) MCV.

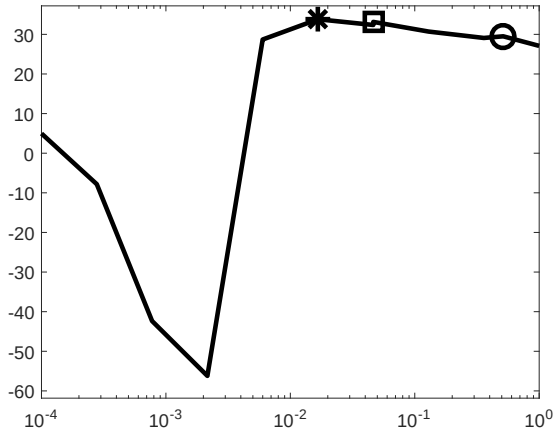
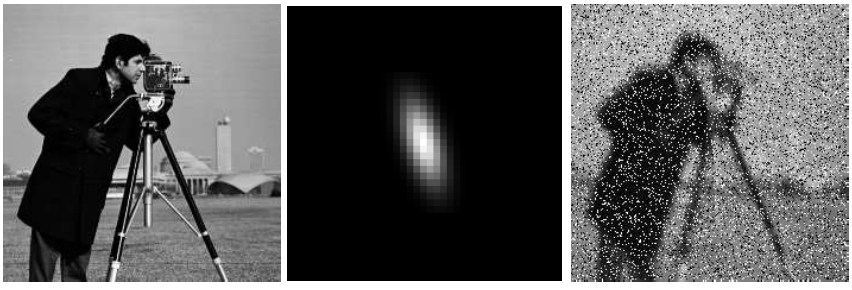


Figure 4: Clock test image: PSNR-values (vertical axis) for different μ-values (horizontal axis). The optimal value of μ is identified by a star, the μ computed by the CV algorithm is identified by a circle, and the μ determined by the MCV algorithm is identified by a square.

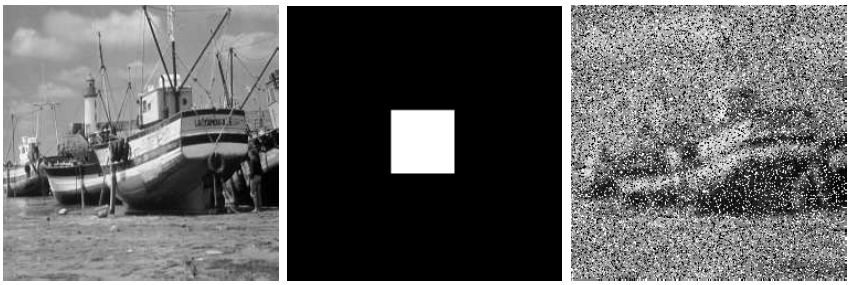


(a) (b) (c)

Figure 5: Cameraman test image: (a) true image (234×234 pixels), (b) PSF (41×41 pixels), (c) blurred and noisy image (salt-and-pepper noise: $\sigma = 0.2$).

(a) (b) (c)

Figure 6: Cameraman restorations: (a) Optimal μ, (b) CV, (c) MCV.

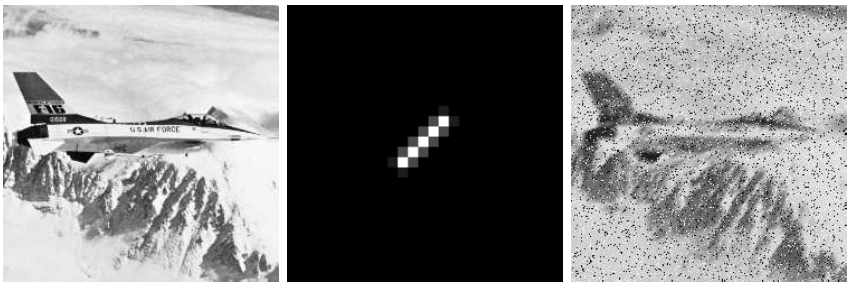


(a) (b) (c)

Figure 7: Boat test image: (a) true image (250×250 pixels), (b) PSF (26×26 pixels), (c) blurred and noisy image (salt-and-pepper noise: $\sigma = 0.3$).

(a) (b) (c)

Figure 8: Boat restorations: (a) Optimal μ, (b) CV, (c) MCV.



(a) (b) (c)

Figure 9: Jet plane test image: (a) true image (247×247 pixels), (b) PSF (27×27 pixels), (c) blurred and noisy image (salt-and-pepper noise: $\sigma = 0.1$, Gaussian noise: $\epsilon = 0.01$).

Boat.. In this example we consider the boat image in Figure 7(a). We blur it with the average PSF shown in Figure 7(b) and add 30% salt-and-pepper noise; the contaminated image is shown in Figure 7(c). We set $p = 0.8$ and $q = 0.5$, and use reflexive boundary conditions.

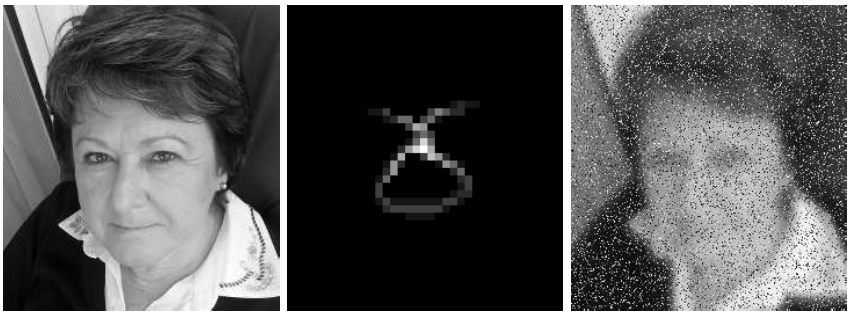
As in the previous examples, we show restorations obtained for three values of the regularization parameter furnished by CV, MCV, and by maximizing the PSNR-value; see Figure 8. The PSNR-values are reported in Table 1. By comparing the restorations, we observe that, similarly as in the previous examples, the MCV method yields a restoration of higher quality than the CV method. Moreover, in this example the parameter μ determined by MCV is very close to the optimal value. The MCV restoration can be seen to have much sharper edges than the CV restoration.

Jet plane.. We consider the image in Figure 9(a). It is blurred with the motion PSF in Figure 9(b), and we add a mixture of salt-and-pepper and Gaussian noise. Specifically, we add 10% of salt-and-pepper noise and 1% of white Gaussian noise. Thus, if $\boldsymbol{\delta}_G$ represents the Gaussian noise, then

$$\eta = \frac{\|\boldsymbol{\delta}_G\|_2}{\|\mathbf{b}\|_2} = 0.01.$$

(a) (b) (c)

Figure 10: Jet plane restorations: (a) Optimal μ, (b) CV, (c) MCV.



(a) (b) (c)

Figure 11: Fiorella test image: (a) true image (264×238 pixels), (b) PSF (41×37 pixels), (c) blurred and noisy image (impulse noise: $\sigma = 0.2$, Gaussian noise: $\epsilon = 0.01$).

Similarly as in the other examples, we impose reflexive boundary conditions and set $p = 0.8$ and $q = 0.1$.

Figure 10 displays the restorations obtained with the MM-GKS-CV and MM-GKS-MCV methods, as well as the restoration obtained with the optimal regularization parameter. Visual inspection of these restorations shows that ℓ_p-ℓ_q minimization is able to determine accurate restorations in the presence of mixed noise. Moreover, we note that both the CV and MCV algorithms are able to determine suitable regularization parameters. These observations are confirmed by the PSNR-values reported in Table 1. The PSNR-values show that, even though the MCV restoration is affected by slight ringing, it is more accurate than the CV restoration.

Fiorella.. In this example we blur the image in Figure 11(a) with the non-symmetric PSF in Figure 11(b), and add a mixture of impulse noise and white Gaussian noise, such that the impulse noise corrupts 20% of the pixels and the norm of the Gaussian noise is 1%. Similarly as above, we impose reflexive boundary conditions and set $p = 0.8$ and $q = 0.1$.

Figure 12 depicts the restorations obtained with all the considered methods and Table 1 displays PSNR-values for the restorations. Visual inspection of

(a) (b) (c)

Figure 12: Fiorella restorations: (a) Optimal μ, (b) CV, (c) MCV.

Example	Optimal μ	CV μ	MCV μ
Clock	33.8103	29.5063	33.1401
Cameraman	24.1153	20.7145	23.9731
Boat	26.1539	25.273	26.1539
Jet plane	26.5764	25.4870	25.7399
Fiorella	27.1715	26.8313	25.1991

Table 1: PSNR values obtained with the two proposed algorithms and by choosing the optimal μ value.

the restorations and comparison of the PSNR-values show that for this example the CV method is able to determine a better regularization parameter than the MCV method. However, both the CV and MCV methods provide accurate restorations in the very difficult scenario of mixed impulse and Gaussian noise.

5. Conclusion and extension

The image restoration methods described in [19, 25] require a user to provide a suitable value of the regularization parameter. This paper develops two approaches based on cross-validation for determining such a value. This enhances the usefulness of the methods in [19, 25].

Computed examples show the two methods for determining the regularization parameter, and in particular the modified cross-validation algorithm (Algorithm 3), to provide good approximations of the optimal parameter μ. We remark that these methods do not require additional knowledge about the image and, thus, are completely automatic. They therefore can be applied to real-world problems.

Similarly as the two-phase methods described in [3, 6, 7, 30], it may be attractive to preprocess images that are contaminated by impulse noise by a median filter. This would result in a fully automatic two-phase method. We are presently investigating the properties of this kind of method.

Acknowledgments

We would like to thank Omar de la Cruz Cabrera for discussions. The work of the first author is partially supported by a grant from the GNCS group of INdAM, while the work of the second author is supported in part by NSF grants DMS-1720259 and DMS-1729509.

References

[1] A. Beck, First-Order Methods in Optimization, SIAM, Philadelphia, 2017.

[2] A. Buccini and L. Reichel, An ℓ_2-ℓ_q regularization method for large discrete ill-posed problems, J. Sci. Comput., in press.

[3] J.-F. Cai, R. H. Chan, and M. Nikolova, Fast two-phase image deblurring under impulse noise, J. Math. Imaging Vis., 36 (2010), pp. 46–53.

[4] J.-F. Cai, R. H. Chan, L. Shen, and Z. Shen, Simultaneously inpainting in image and transformed domains, Numer. Math., 112 (2009), pp. 509–533.

[5] J.-F. Cai, R. H. Chan, and Z. Shen, A framelet-based image inpainting algorithm, Appl. Comput. Harmonic Anal., 24 (2008), pp. 131–149.

[6] R. H. Chan, H. Chung-Wa, and M. Nikolova, Salt-and-pepper noise removal by median-type noise detectors and detail-preserving regularization, IEEE Trans. Image Process., 14 (2005), pp. 1479–1485.

[7] R. H. Chan, Y. Dong, and M. Hintermüller, An efficient two-phase L^1-TV method for restoring blurred images with impulse noise, IEEE Trans. Image Process., 19 (2010), pp. 1731–1739.

[8] R. H. Chan and H. X. Liang, Half-quadratic algorithm for ℓ_p-ℓ_q problems with applications to TV-ℓ_1 image restoration and compressive sensing, in Proceedings of Efficient Algorithms for Global Optimization Methods in Computer Vision, Lecture Notes in Comput. Sci. # 8293, Springer, Berlin, 2014, pp. 78–103.

[9] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart, Reorthogonalization and stable algorithms for updating the Gram–Schmidt QR factorization, Math. Comp., 30 (1976), pp. 772–795.

[10] M. Donatelli, T. Huckle, M. Mazza, and D. Sesana, Image deblurring by sparsity constraint on the Fourier coefficients, Numer. Algorithms, 72 (2016), pp. 341–361.

[11] A. El Mouatasim and M. Wakrim, Control subgradient algorithm for image ℓ_1 regularization, Signal, Image and Video Processing (SIVIP), 9 (Supplement 1), (2015), pp. 275–283.

[12] H. W. Engl, M. Hanke, and A. Neubauer, Regularization of Inverse Problems, Kluwer, Dordrecht, 1996.

[13] C. Estatico, S. Gratton, F. Lenti, and D. Titley-Peloquin, A conjugate gradient like method for p-norm minimization in functional spaces, Numer. Math., 137 (2017), pp. 895–922.

[14] S. Gazzola and J. G. Nagy, Generalized Arnoldi–Tikhonov method for sparse reconstruction, SIAM J. Sci. Comput., 36 (2014), pp. B225–B247.

[15] S. Gazzola, P. Novati, and M. R. Russo, On Krylov projection methods and Tikhonov regularization, Electron. Trans. Numer. Anal., 44 (2015), pp. 83–123.

[16] M. Hanke and P. C. Hansen, Regularization methods for large-scale problems, Surv. Math. Ind., 3 (1993), pp. 253–315.

[17] P. C. Hansen, Rank Deficient and Discrete Ill-Posed Problems, SIAM, Philadelphia, 1998.

[18] P. C. Hansen, J. G. Nagy, and D. P. O'Leary, Deblurring Images: Matrices, Spectra, and Filtering, SIAM, Philadelphia, 2006.

[19] 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.

[20] J. Huang, M. Donatelli, and R. H. Chan, Nonstationary iterated thresholding algorithms for image deblurring, Inverse Probl. Imaging, 7 (2013), pp. 717–736.

[21] S. Kindermann, Discretization independent convergence rates for noise level-free parameter choice rules for the regularization of ill-conditioned problems, Electron. Trans. Numer. Anal., 38 (2011), pp. 233–257.

[22] S. Kindermann, Convergence analysis of minimization-based noise level-free parameter choice rules for linear ill-posed problems, Electron. Trans. Numer. Anal., 40 (2013), pp. 58–81.

[23] G. H. Golub and C. F. Van Loan, Matrix Computations, Johns Hopkins University Press, 4th ed., Baltimore, 2013.

[24] J. Lampe, L. Reichel, and H. Voss, Large-scale Tikhonov regularization via reduction by orthogonal projection, Linear Algebra Appl., 436 (2012), pp. 2845–2865.

[25] 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.

[26] A. Lanza, S. Morigi, and F. Sgallari, Constrained TV_p-ℓ_2 model for image restoration, J. Sci. Comput., 68 (2016), pp. 64–91.

[27] M. K. Ng, R. H. Chan, and W.-C. Tang, A fast algorithm for deblurring models with Neumann boundary conditions, SIAM J. Sci. Comput, 21 (1999), pp. 851–866.

[28] L. Reichel and G. Rodriguez, Old and new parameter choice rules for discrete ill-posed problems, Numer. Algorithms, 63 (2013), pp. 65–87.

[29] P. Rodríguez and B. Wohlberg, Efficient minimization method for a generalized total variation functional, IEEE Trans. Image Process., 18 (2009), pp. 322–332.

[30] F. Sciacchitano, Y. Dong, and M. S. Andersen, Total Variation based parameter-free model for impulse noise removal, Numer. Math. Theor. Meth. Appl., 10 (2017), pp. 186–204.

[31] M. Stone, Cross-validatory choice and assessment of statistical prediction, J. Royal Stat. Soc., series B, 36 (1977), pp. 111–147.

[32] R. Wolke and H. Schwetlick, Iteratively reweighted least squares: algorithms, convergence analysis, and numerical comparisons, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 907–921.