Robust non-linear regression analysis: A greedy approach employing kernels and application to image denoising

George Papageorgiou[*], Pantelis Bouboulis[†]and Sergios Theodoridis[‡]

January 5, 2016

Abstract

We consider the task of robust non-linear estimation in the presence of both bounded noise and outliers. Assuming that the unknown non-linear function belongs to a Reproducing Kernel Hilbert Space (RKHS), our goal is to accurately estimate the coefficients of the kernel regression matrix. Due to the existence of outliers, common techniques such as the Kernel Ridge Regression (KRR), or the Support Vector Regression (SVR) turn out to be inadequate. Instead, we employ sparse modeling arguments to model and estimate the outliers, adopting a greedy approach. In particular, the proposed robust scheme, i.e., Kernel Greedy Algorithm for Robust Denoising (KGARD), is a modification of the classical Orthogonal Matching Pursuit (OMP) algorithm. In a nutshell, the proposed scheme alternates between a KRR task and an OMP-like selection step. Convergence properties as well as theoretical results concerning the identification of the outliers are provided. Moreover, KGARD is compared against other cutting edge methods (using toy examples) to demonstrate its performance and verify the aforementioned theoretical results. Finally, the proposed robust estimation framework is applied to the task of image denoising, showing that it can enhance the denoising process significantly, when outliers are present.

1 Introduction

The problem of function estimation has attracted significant attention in the machine learning community over the past decades. In this paper, we target the specific task of regression, which is typically described as follows: given a data set of the form $\mathcal{D} = \{(y_i, \boldsymbol{x}_i)\}_{i=1}^N$, we aim to estimate the input-output relation between $\boldsymbol{x}_i$ and y_i, i.e., a function f, such that $f(\boldsymbol{x}_i)$ is "close" to y_i, for all i. This is usually achieved by employing a *loss function*, i.e., a function $C(\boldsymbol{x}_i, y_i, f(\boldsymbol{x}_i))$, that measures the deviation between the observed values, y_i, and the predicted values, $f(\boldsymbol{x}_i)$, and minimizing the so called *Empirical Risk*, i.e. $\sum_{i=1}^N C(\boldsymbol{x}_i, y_i, f(\boldsymbol{x}_i))$. For example, in the least squares regression, one adopts the squared error, i.e., $C(\boldsymbol{x}_i, y_i, f(\boldsymbol{x}_i)) := (y_i - f(\boldsymbol{x}_i))^2$ and minimizes a quadratic function.

Naturally, the choice for f strongly depends on the underlying true model. For example, assuming that our observations are generated via $y_i = \boldsymbol{x}_i^T \underline{\boldsymbol{\theta}} + v_i,\ i = 1, ..., N$, where v_i's are random noise variables or $\boldsymbol{y} = \boldsymbol{X}\underline{\boldsymbol{\theta}} + \boldsymbol{v}$ more compactly, it is reasonable to adopt a linear input-output relation $f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\theta}$ aiming to estimate $\underline{\boldsymbol{\theta}} \in \mathbb{R}^M$. This is the case of linear ridge regression, where the goal is to minimize $L_2(\boldsymbol{\theta}) := \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_2^2,\ \lambda \geq 0$. In the particular case where $N < M$, additional sparsity constraints on $\boldsymbol{\theta}$ lead to the case of sparse modelling, which has gained in popularity in the recent years. For example, one might choose to find the sparsest $\boldsymbol{\theta}$ that keeps the squared error low (i.e., minimize $\|\boldsymbol{\theta}\|_0$ subject to the constraint $\|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|_2^2 \leq \varepsilon$) or modify the ridge regression task to include an ℓ_1 norm regularization term (i.e., minimize the cost function $L_1(\boldsymbol{\theta}) := \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_1$).

[*] geopapag@di.uoa.gr

[†] panbouboulis@gmail.com

[‡] stheodor@di.uoa.gr

In this paper, we assume that f belongs to a RKHS. These are inner product function spaces, in which every function is reproduced by an associated (space defining) kernel; that is for every $\boldsymbol{x} \in \mathcal{X}$, there exists $\kappa(\cdot, \boldsymbol{x}) \in \mathcal{H}$, such that $f(\boldsymbol{x}) = \langle f, \kappa(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}}$. This is the case that has been addressed (amongst others) by two very popular and well-established methods which are commonly referred to as the *Kernel Ridge Regression* (KRR) and the *Support Vector Regression* (SVR).

Another important issue that determines the quality of the estimation is the underlying noise model. Undeniably, the aforementioned least squares methods are optimum for the linear regression task (in the maximum likelihood sense), in the presence of *white Gaussian noise* [1]. However, this is not the case when *outliers* are present or when the noise distribution exhibits long tails; this task is widely known as *robust* linear regression and has been thoroughly studied over the years.

Initially, it should be pointed out that, in order to be able to handle this task, even in the linear case, a few assumptions are necessary. First of all, the regression matrix should be full rank. This is a condition required even for the non-robust case. In fact, if we let $\text{rank}(\boldsymbol{X}) = r < M$, then the solution to the LS task is not unique [2, 3]. Hence, for the linear case, we should impose $N > M$. The proposed methods for solving the task of linear robust estimation are divided into two categories. The first one includes methods that penalize large residuals, while the second, comprises methods where explicit outlier modelling is employed. The latter approach is quite recent, whereas the first one has been developed during the 70's. The primary representative of the first approach is the family of weighted least squares (WLS) methods which uses weights (weighting coefficients) in order to penalize large residuals. This method is also known as M-estimator, with a large number of variations established over the years [4, 5, 6, 7, 8, 9, 10]. The second approach employs sparsity arguments, while assuming that the noise vector $\boldsymbol{v}$ is decomposed into two parts, a dense vector $\boldsymbol{\eta}$ of inlier noise and a sparse outlier one $\boldsymbol{u}$, i.e., $\boldsymbol{v} = \boldsymbol{u} + \boldsymbol{\eta}$. As a result, sparsity constraints are only applied to vector $\boldsymbol{u}$. Such methods are based on the so-called ℓ_1 minimization techniques (LASSO formulation task) solvable by a variety of methods such as the alternating direction method of multipliers (ADMM) [11, 12], sparse Bayesian learning techniques [13, 14, 1] and greedy selection methods [3].

Although the problem of robust regression has been extensively studied for the linear case and many algorithms have been established, the more general non-linear case employing kernels has been addressed only recently [15, 16]. In this case, y_i is assumed to be generated by

$$y_i = \underline{f}(\boldsymbol{x}_i) + v_i, \ \ i = 1, ..., N, \tag{1}$$

where v_i are random noise variables which may contain outliers. The present paper focuses on this task in the special case where the unknown function, $\underline{f}$, is assumed to lie in an RKHS, $\mathcal{H}$. It should be noted that both SVR and KRR can be employed to address this problem, but the presence of outliers reduces significantly their performance due to over-fitting, [17, 18]. Of course, in SVR this effect is not as dominant as in typical KRR, due to the ℓ_1 loss that it is employed. Hence, a more specific treatment is required in order to establish a robust estimator for the KRR task.

Our proposed method adopts a model of the form $y = f(\boldsymbol{x})$, where $f \in \mathcal{H}$ and, also, a decomposition of the noise into two parts, the inlier vector $\boldsymbol{\eta}$ and the sparse outlier vector $\boldsymbol{u}$, in a way similar to the aforementioned robust linear regression methods. Then, it employs a two step algorithmic procedure attempting to estimate both the outliers and also the original function $\underline{f}$. The algorithm alternates between a greedy method based on the popular *Orthogonal Matching Pursuit* (OMP) [19, 20, 21], that selects the dominant outlier in each step, and a kernel ridge regression task to update the estimate of $\underline{f}$. Results regarding convergence as well as theoretical properties of the recovery of the outlier's support are also provided. Moreover, comparisons against the previously published approaches based on the Bayesian framework and on the minimization of the ℓ_1-norm for the sparse outlier vector are performed.

The rest of the paper is organized as follows. In section 2, the basic properties of RKHS and *greedy methods*, under the sparsity-aware learning umbrella, are presented, and in section 3, the problem is formulated and comparable state-of-the-art methods are reviewed. Next, in section

4 the proposed scheme is introduced and described in detail. Section 5 provides the theoretical results, regarding the convergence of the scheme as well as the identification of the outliers. In section 6, extended tests of the proposed scheme against other cutting edge methods are performed. There, the efficiency of the proposed method in terms of both the achieved mean squared error as well as the complexity is investigated. Finally, in section 7, the method is applied to the task of robust denoising for images, in order to remove the mix of impulsive and Gaussian noise from images.

Notation: Throughout this work, capital calligraphic letters are employed to denote sets, e.g., $\mathcal{S}$, where $\mathcal{S}^c$ denotes the complement of $\mathcal{S}$. Small letters denote scalars, e.g., ε, while bold capital letters denote matrices, e.g., $\boldsymbol{X}$, bold lowercase letters are reserved for vectors, e.g., $\boldsymbol{\theta}$ (each vector is regarded as a column vector) and the symbol $\cdot^T$ denotes the transpose of the respective matrix/vector. Also diag($\boldsymbol{a}$), where $\boldsymbol{a}$ is a vector, denotes the respective square diagonal matrix[1], while supp($\boldsymbol{a}$) denotes the support set of the vector $\boldsymbol{a}$. The $j-$th column of matrix $\boldsymbol{X}$ is denoted by $\boldsymbol{x}_j$ and the element of the $i-$th row and $j-$th column of matrix $\boldsymbol{X}$ by x_{ij}. Moreover, the $i-$th element of vector $\boldsymbol{\theta}$ is denoted by θ_i. An arithmetic index in parenthesis, i.e., (k), $k = 0, 1, \dots$ is reserved to declare an iterative (algorithmic) process, e.g., on matrix $\boldsymbol{X}$ and vector $\boldsymbol{r}$ the iteratively generated matrix and vector are denoted by $\boldsymbol{X}_{(k)}$ and $\boldsymbol{r}_{(k)}$, respectively. Following this rationale, $r_{(k),i}$ is reserved for the $i-$th element of the iteratively generated vector $\boldsymbol{r}_{(k)}$. The notation $\boldsymbol{X}_{\mathcal{S}}$ denotes the matrix $\boldsymbol{X}$ restricted over the set $\mathcal{S}$, i.e., the matrix that comprises the columns of $\boldsymbol{X}$, whose indices belong to the ordered index set $\mathcal{S} = \{j_1 < \cdots < j_s\}$. Accordingly, the notation $\boldsymbol{u}_{\mathcal{S}}$ denotes the elements of vector $\boldsymbol{u}$, restricted over the set $\mathcal{S} \subseteq \text{supp}(\boldsymbol{u})$. Finally, the identity matrix of dimension N will be denoted as $\boldsymbol{I}_N$ where $\boldsymbol{e}_j$ is its $j-$th column vector, the zero matrix of dimension $N \times N$, as $\boldsymbol{O}_N$, the vector of zero elements of appropriate dimension as $\boldsymbol{0}$ and the columns of matrix $\boldsymbol{I}_N$ restricted over the set $\mathcal{S}$, as $\boldsymbol{I}_{\mathcal{S}}$.

2 Preliminaries

2.1 Reproducing Kernel Hilbert Spaces

In this section, an overview of some of the basic properties of the RKHS is provided [22, 23, 24, 25].

A RKHS [26], is a Hilbert space $\mathcal{H}$ over a field $\mathbb{F}$ for which there exists a positive definite function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{F}$ with the following two important properties:

- For every $\boldsymbol{x} \in \mathcal{X}$, $\kappa(\cdot, \boldsymbol{x})$ belongs to $\mathcal{H}$ and
- κ has the so called *reproducing property*, i.e., $f(\boldsymbol{x}) = \langle f, \kappa(\cdot, \boldsymbol{x})\rangle_{\mathcal{H}}$, for all $f \in \mathcal{H}$, in particular $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \langle \kappa(\cdot, \boldsymbol{y}), \kappa(\cdot, \boldsymbol{x})\rangle_{\mathcal{H}}$.

The *Gram* matrix $\boldsymbol{K}$ corresponding to the kernel κ, i.e., the matrix with elements $\kappa_{ij} := \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$, is positive definite for any selection of finite number of points $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N$, $N \in \mathbb{N}^*$. The following popular theorem establishes that although a RKHS may have infinite dimension, the solution of any regularized risk regression optimization task lies in the span of N specific kernels.

Theorem 1 (Representer Theorem)**.** *Denote by $\Omega : [0, +\infty) \to \mathbb{R}$ a strictly monotonic increasing function, by $\mathcal{X}$ a nonempty set and by $L : \mathcal{X} \times \mathbb{R}^2 \to \mathbb{R} \cup \{\infty\}$ an arbitrary loss function. Then each minimizer $f \in \mathcal{H}$ of the regularized minimization problem:*

$$\min_f \left\{ \sum_{i=1}^N L\left(\boldsymbol{x}_i, y_i, f(\boldsymbol{x}_i)\right) + \Omega\left(||f||_{\mathcal{H}}\right) \right\},$$

admits a representation of the form $f = \sum_{j=1}^N \alpha_j \kappa(\cdot, \boldsymbol{x}_j)$.

[1] This matrix has the vector's coefficients on its *diagonal*, while all other entries are equal to zero.

In many applications a *bias* term, c, is often included to the aforementioned expansion; in other words, we assume that f admits the following representation:

$$f = \sum_{j=1}^{N} \alpha_j \kappa(\cdot, \boldsymbol{x}_j) + c. \tag{2}$$

The use of the bias factor is theoretically justified by the Semi-parametric Representer Theorem [22, 1].

Although there are many kernels to choose from, in this paper the experiments are focused on the real *Gaussian radial basis function* (RBF), i.e., $\kappa_\sigma(\boldsymbol{x}, \boldsymbol{x}') := \exp\left(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2/\sigma^2\right)$, defined for $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^M$, where σ, is a free positive parameter that defines the shape of the kernel function. In the following, κ is adopted to denote the Gaussian RBF. An important property of this kernel is presented in the following theorem.

Theorem 2 (Full Rank of Gaussian RBF Gram Matrix)**.** *Suppose that* $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N \subset \mathcal{X} \subseteq \mathbb{R}^\nu$ *are distinct points and* $\sigma > 0$*. The matrix* $\boldsymbol{K}$ *given by*

$$\kappa_{ij} := exp(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}{\sigma^2})$$

has full rank.

The significance of the theorem is that the points $\kappa(\cdot, \boldsymbol{x}_1), \kappa(\cdot, \boldsymbol{x}_2), ..., \kappa(\cdot, \boldsymbol{x}_N) \in \mathcal{H}$ are linearly independent, i.e. span the N-dimensional subspace of $\mathcal{H}$ [22].

2.2 Sparse Modeling and the Orthogonal Matching Pursuit (OMP)

Undeniably, sparsity-aware learning techniques is one among the most prominent fields in the area of machine learning and signal processing and has gained considerable attention over the last decade. Sparsity is the feature of a model that is represented via an economic form, i.e., one which contains as many zero elements as possible. The goal of obtaining sparse/economic representations [27, 1], can be reached via various seemingly distinct directions, such as convex optimization (based on ℓ_1 minimization), *greedy selection* techniques and non-convex optimization. Amongst those, the present paper is focused on the greedy approach.

The core, for the greedy family of methods, is the Orthogonal Matching Pursuit (OMP) algorithm [19, 20, 21]. The scheme is based on the classic Matching Pursuit method, which was proposed for signal compression [28, 29, 30], although the method was already familiar to statisticians. Apart from OMP, other variants also exist [31, 32, 33, 34, 35], although they fall out of the scope of this paper.

Let $\boldsymbol{A} \in \mathbb{R}^{N \times M}$, with $N < M$, be full-rank matrix. Since $\boldsymbol{A}$ is an overcomplete dictionary, the linear system of equations $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}$ has infinitely many solutions. The fact that no unique solution exists, is reinforced when our observations are corrupted by additive noise $\boldsymbol{\eta}$, i.e., assuming that $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\eta}$. Thus, only additional constraints, imposed to the unknown vector, restrict the set of all possible solutions. Hence, if we wish to acquire a sparse solution/estimate, the ℓ_0 (pseudo)-norm should be employed. The OMP scheme attempts to solve the following minimization problem:

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_0 \text{ s.t. } \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2 \leq \epsilon, \tag{3}$$

a task known as sparse denoising. Although (3) is a NP-Hard combinatorial problem, thus not solvable in polynomial time, under certain assumptions [27, 1], the OMP algorithm (as other methods too) succeeds to provide sufficiently good and simultaneously sparse solutions. The algorithm sequentially selects columns of $\boldsymbol{A}$ that correspond to non-zero elements of the sparse vector $\boldsymbol{x}$. The scheme is presented in Algorithm 1.

Steps 6 and 8 are of great importance, since the proposed non-linear method, i.e., KGARD, builds upon similar arguments. At the selection step (step 6), the method identifies the column

Algorithm 1 Orthogonal Matching Pursuit: OMP

```
 1: procedure OMP($\boldsymbol{A}, \boldsymbol{b}, \epsilon$)
 2:     $\hat{\boldsymbol{x}} \leftarrow \boldsymbol{0}$
 3:     $\boldsymbol{r} \leftarrow \boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}} = \boldsymbol{b}$
 4:     $\mathcal{S} \leftarrow \emptyset$
 5:     while $\|\boldsymbol{r}\|_2 > \epsilon$ do
 6:         $j_k := \arg\max_j \frac{|\langle \boldsymbol{r}, \boldsymbol{a}_j \rangle|}{\|\boldsymbol{a}_j\|_2^2}$                ▷ Selection step.
 7:         $\mathcal{S} \leftarrow \mathcal{S} \cup \{j_k\}$
 8:         $\hat{\boldsymbol{x}} := \arg\min_{\boldsymbol{x}} \|\boldsymbol{b} - \boldsymbol{A}_{\mathcal{S}}\boldsymbol{x}\|_2^2$      ▷ Least Squares solution step.
 9:         $\boldsymbol{r} \leftarrow \boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}}$
10:     Output: a sparse vector $\hat{\boldsymbol{x}}$.
```

of matrix $\boldsymbol{A}$, which is more correlated with the residual up to this point. Then, the support set of the active columns $\mathcal{S}$ (associated with indices of previously selected columns) is augmented by the newly selected column and the Least Squares minimization task is performed (step 8), restricted over the subspace that originates from the columns $\boldsymbol{a}_j$ of $\boldsymbol{A}$ that belong to the set $\mathcal{S}$, i.e., columns of $\boldsymbol{A}_{\mathcal{S}}$. At this point, we should emphasize, that once a column is selected at a particular step, it cannot be selected again (in future steps), since its inner product with any future residual, i.e., $\langle \boldsymbol{r}, \boldsymbol{a}_j \rangle$, is zero.

3 Problem Formulation and Related Works

3.1 Robust Ridge Regression in RKHS

Assume that a specific RKHS $\mathcal{H}$ and the data set $\mathcal{D} = \{(y_i, \boldsymbol{x}_i)\}_{i=1}^N$ are given, so that each measurement y_i is produced from $\boldsymbol{x}_i$, via

$$y_i = \underline{f}(\boldsymbol{x}_i) + \underline{u}_i + \eta_i, \; i = 1, \dots, N \tag{4}$$

where $\underline{f} \in \mathcal{H}$, $\underline{u}_i$ represents a possible outlier and η_i a bounded noise component. In a more compact form, this can be cast as $\boldsymbol{y} = \underline{\boldsymbol{f}} + \underline{\boldsymbol{u}} + \boldsymbol{\eta}$, where $\underline{\boldsymbol{f}}$ is the vector containing the values $\underline{f}(\boldsymbol{x}_i)$ for all $i = 1, \dots, N$. As $\underline{\boldsymbol{u}}$ represents the vector of the (unknown) outliers, it is reasonable to assume that this is a sparse vector. Our goal is to estimate the input-output relation $\underline{f}$ from the noisy observations of the data set $\mathcal{D}$. This can be interpreted as the task of of simultaneously estimating both a sparse vector $\boldsymbol{u}$ and as well as a function $f \in \mathcal{H}$, that maintains a low squared error for $L(\mathcal{D}, f, \boldsymbol{u}) = \sum_{i=1}^N (y_i - f(\boldsymbol{x}_i) - u_i)^2$. Moreover, motivated by the representer theorem, we adopt the representation in (2), as a means to represent the solution for f. Under these assumptions, equation (4) could be expressed in a compact form as

$$\boldsymbol{y} = \boldsymbol{K}\underline{\boldsymbol{\alpha}} + \underline{c}\mathbf{1} + \underline{\boldsymbol{u}} + \boldsymbol{\eta} = \boldsymbol{X}_{(0)} \begin{pmatrix} \underline{\boldsymbol{\alpha}} \\ \underline{c} \end{pmatrix} + \boldsymbol{v}, \tag{5}$$

where $\boldsymbol{K}$ is the kernel matrix, $\boldsymbol{X}_{(0)} = [\boldsymbol{K} \; \mathbf{1}]$ and $\boldsymbol{v} = \underline{\boldsymbol{u}} + \boldsymbol{\eta}$ is the total noise vector (inlier and outlier). Accordingly, the squared error can be written, in terms of the corresponding estimates, as

$$L(\mathcal{D}, \boldsymbol{\alpha}, c, \boldsymbol{u}) = \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{a} - c\mathbf{1} - \boldsymbol{u}\|_2^2,$$

and the respective task can be cast as:

$$\begin{aligned} \min_{\boldsymbol{u}, \boldsymbol{a} \in \mathbb{R}^N, c \in \mathbb{R}} \quad & \|\boldsymbol{u}\|_0 \\ \text{subject to} \quad & \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{a} - c\mathbf{1} - \boldsymbol{u}\|_2^2 \leq \varepsilon_1, \\ & \left\| \begin{pmatrix} \boldsymbol{a} \\ c \end{pmatrix} \right\|_2^2 \leq \varepsilon_2, \end{aligned} \tag{6}$$

for some $\varepsilon_1, \varepsilon_2 > 0$, where we have also included a constraint that keeps the norm of the vector of the kernel expansion coefficients low (as an attempt to guard against overfitting). This can be equivalently expressed using regularization terms as follows:

$$\begin{array}{cc} \min\limits_{\boldsymbol{u},\boldsymbol{a}\in\mathbb{R}^N, c\in\mathbb{R}} & \|\boldsymbol{u}\|_0 \\ \text{subject to} & \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{a} - c\boldsymbol{1} - \boldsymbol{u}\|_2^2 + \lambda\|\boldsymbol{a}\|_2^2 + \lambda c^2 \leq \varepsilon, \end{array} \tag{7}$$

for some predefined parameters $\varepsilon, \lambda > 0$. An alternative regularization strategy, which is common in the respective literature (based on KRR), is to include the norm of f, i.e., $\|f\|_{\mathcal{H}}^2 = \boldsymbol{a}^T\boldsymbol{K}\boldsymbol{a}$, instead of the norm of the coefficients' vector, leading to the following task:

$$\begin{array}{cc} \min\limits_{\boldsymbol{u},\boldsymbol{a}\in\mathbb{R}^N, c\in\mathbb{R}} & \|\boldsymbol{u}\|_0 \\ \text{subject to} & \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{a} - c\boldsymbol{1} - \boldsymbol{u}\|_2^2 + \lambda\boldsymbol{a}^T\boldsymbol{K}\boldsymbol{a} \leq \varepsilon. \end{array} \tag{8}$$

3.2 Convex Relaxation: Refined Alternating Directions Method of Multipliers (RAM)

It is evident that the problems (7), (8) constitute non-convex optimization tasks. In order to achieve stable solutions and mobilize the rich toolbox of convex optimization, many authors prefer to consider an alternative convex task, which is closely related to the original minimization problem, using a convex relaxation technique. This can be achieved by substituting the ℓ_0 "norm" of the sparse outlier vector $\boldsymbol{u}$ with the closest convex norm, i.e., the ℓ_1 norm. Thus, problem (8) can be cast as:

$$\begin{array}{c} \min\limits_{\boldsymbol{u}, f\in\mathcal{H}} ||\boldsymbol{u}||_1 \\ \text{s.t.} \left\{ \sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i) - u_i)^2 + \lambda||f||_{\mathcal{H}}^2 \right\} \leq \varepsilon, \end{array} \tag{9}$$

for $\varepsilon, \lambda > 0$. Considering the linear representation $f = \sum_{j=1}^{N} \alpha_j \kappa(\cdot, \boldsymbol{x}_j)$ (no bias factor c was used in [15]), the constraint task (9) is equivalent to

$$\min_{\boldsymbol{\alpha},\boldsymbol{u}} \left\{ \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{u}\|_2^2 + \lambda\boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{\alpha} + \mu||\boldsymbol{u}||_1 \right\}, \tag{10}$$

for values of $\mu > 0$ depending on values of $\varepsilon > 0$, where $\boldsymbol{\alpha}$ is the vector of the kernel's coefficients and $\boldsymbol{u}$ is the outlier vector [15]. The respective convex minimization form is known as the (generalized) LASSO task [36, 37], which is solvable by a large variety of methods, e.g., using the Alternating Direction Method of Multipliers (ADMM) or its efficient implementation, i.e., the so-called *AM solver* [15]. This scheme is further improved, by using a non-convex relaxation technique that attempts to solve

$$\min_{\boldsymbol{\alpha},\boldsymbol{u}} \left\{ \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{u}\|_2^2 + \lambda\boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{\alpha} + \mu \sum_{i=1}^{N} \log\left(|u_i| + \delta\right) \right\},$$

for $\delta > 0$ sufficiently small in order to avoid numerical instability. Since the additional regularization term is now concave, the overall problem is non-convex. However, it can be replaced with the local linear approximation of the logarithmic function via the use of the reweighted ℓ_1-norm technique [38], leading to

$$[\hat{\boldsymbol{a}}_{(k)}, \hat{\boldsymbol{u}}_{(k)}] \quad := \quad \arg\min_{\boldsymbol{\alpha},\boldsymbol{u}} \left\{ \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{u}\|_2^2 + \lambda\boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{\alpha} + \mu \sum_{i=1}^{N} w_{(k),i}|u_i| \right\}, \tag{11}$$

where the coordinates of $\boldsymbol{w}_{(k)}$ are given by

$$w_{(k),i} \quad := \quad \left(\left|u_{(k-1),i}\right| + \delta\right)^{-1}, \ i = 1, ..., N. \tag{12}$$

Algorithm 2 (Weighted) Alternating directions solver: WAM

```
procedure WAM($\boldsymbol{K}$, $\boldsymbol{y}$, $\mu$, $\lambda$, $\boldsymbol{w}$)
    $\hat{\boldsymbol{u}}_{(0)} \leftarrow \boldsymbol{0}$
    for $k = 1, 2, ...$ do
        $\hat{\boldsymbol{\alpha}}_{(k)} \leftarrow [\boldsymbol{K} + \lambda \boldsymbol{I}_N]^{-1} \left(\boldsymbol{y} - \hat{\boldsymbol{u}}_{(k-1)}\right)$
        $\boldsymbol{r}_{(k)} \leftarrow \boldsymbol{y} - \boldsymbol{K}\hat{\boldsymbol{\alpha}}_{(k)},\ \hat{u}_{(k),i} \leftarrow S\left(r_{(k),i}, \frac{w_i \mu}{2}\right), i = 1, ...N$
    Output: $\hat{\boldsymbol{\alpha}}_{(k)}$ and $\hat{\boldsymbol{u}}_{(k)}$ after $k$ iterations.
```

Algorithm 3 Refined AM solver: RAM

```
procedure RAM($\boldsymbol{K}$, $\boldsymbol{y}$, $\mu$, $\lambda$, $\delta$)
    $\left[\hat{\boldsymbol{\alpha}}_{(0)}, \hat{\boldsymbol{u}}_{(0)}\right] \leftarrow$ WAM($\boldsymbol{K}, \boldsymbol{y}, \mu, \lambda, \boldsymbol{1}$)
    for $k = 1, 2, ...$ do
        $w_{(k),i} = (|\hat{u}_{(k-1),i}| + \delta)^{-1},\ i = 1, ..., N,$
        $\left[\hat{\boldsymbol{\alpha}}_{(k)}, \hat{\boldsymbol{u}}_{(k)}\right] \leftarrow$ WAM($\boldsymbol{K}, \boldsymbol{y}, \mu, \lambda, \boldsymbol{w}_{(k)}$)
    Output: $\hat{\boldsymbol{\alpha}}_{(k)}$ and $\hat{\boldsymbol{u}}_{(k)}$ after $k$ iterations.
```

The Refined AM solver scheme is summarized in Algorithms 3 and 2, where S denotes the soft-thresholding operator $S(z, \gamma) := sign(z) \cdot \max(0, |z| - \gamma)$. It should also be noted that, the original AM solver (an improved implementation of ADMM), could be obtained from Algorithm 2, for weights equal to one, i.e., setting $\boldsymbol{w} = \boldsymbol{1}$; the WAM solver is a more general scheme. Finally, the scheme could be implemented more efficiently, by applying the Cholesky factorization (with cost $O(N^2)$ after the factorization) instead of an inversion, since matrix $[\boldsymbol{K} + \lambda \boldsymbol{I}_N]$ remains unchanged. The aforementioned refinement step was shown to greatly improve the performance of the original AM solver [15]. Moreover, in practice, more than 2 iterations do not offer significant improvements on its performance. Furthermore, we should emphasize that the optimum parameters (λ_*, μ_*) to be used with RAM (in terms of MSE), are not identical to the parameters of AM (λ, μ) in (10) (WAM with $\boldsymbol{w} = \boldsymbol{1}$). Thus, for $\mu_* > \mu$ (fluctuation of the step size), the convergence speed of the RAM scheme is also improved. Finally, theoretical properties of the method indicate that for small values of $\delta > 0$, the method attempts to approximate the ℓ_0 norm of the sparse outlier vector $\boldsymbol{u}$.

3.3 Sparse Bayesian Learning: Robust Relevance Vector Machine (RB-RVM)

Another kernel-based related method is via the Bayesian approach. The Sparse Bayesian learning scheme, i.e., the Robust Bayesian-RVM (RB-RVM), is an RVM modified scheme that employees the use of hyper-parameters to impose sparsity on the outlier estimates. [16, 39, 1].

Assuming that f admits the linear representation in (2), the authors suggest the reformulation of (5), in the form $\boldsymbol{y} = \boldsymbol{X}\underset{\sim}{\boldsymbol{z}} + \boldsymbol{\eta}$, where $\boldsymbol{X} = [\boldsymbol{K}\ \boldsymbol{1}\ \boldsymbol{I}_N]$, $\underset{\sim}{\boldsymbol{z}} = (\underline{\boldsymbol{\theta}}^T, \underline{\boldsymbol{u}}^T)^T$ and $\underline{\boldsymbol{\theta}} = (\underline{\boldsymbol{\alpha}}^T, \underline{c})^T$, which is the vector of the coefficients to be estimated. Then, the joint posterior distribution of $\underline{\boldsymbol{\theta}}$ and $\underline{\boldsymbol{u}}$ (assumed independent) is estimated via:

$$p(\underline{\boldsymbol{\theta}}, \underline{\boldsymbol{u}}|\boldsymbol{y}) = \frac{p(\underline{\boldsymbol{\theta}})p(\underline{\boldsymbol{u}})p(\boldsymbol{y}|\underline{\boldsymbol{\theta}}, \underline{\boldsymbol{u}})}{p(\boldsymbol{y})},$$

where $p(\boldsymbol{y}|\underline{\boldsymbol{\theta}}, \underline{\boldsymbol{u}}) = \mathcal{N}(\boldsymbol{X}\underset{\sim}{\boldsymbol{z}}, \sigma^2 \boldsymbol{I}_N)$ and assuming that the elements of inlier noise vector $\boldsymbol{\eta}$ belong to a Gaussian distribution with variance σ^2. Next, priors which 'promote sparsity' are assigned to the vectors $\underline{\boldsymbol{\theta}}$ and $\underline{\boldsymbol{u}}$. To this end,

$$p(\boldsymbol{s}|\boldsymbol{h}) = \prod_{i=0}^{N} \mathcal{N}(s_i|0, h^{-1})$$

holds for both vectors $\underline{\boldsymbol{\theta}}$ and $\underline{\boldsymbol{u}}$, with hyper-parameters $\boldsymbol{\beta} = [\beta_{,} \dots, \beta_{N+1}]^T$ and $\boldsymbol{\delta} = [\delta_1, \dots, \delta_N]^T$, respectively, where each of the hyper-parameters follows a uniform distribution. The maximization of $p(\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{\delta}, \sigma^2)$ is performed by an EM algorithm, the parameters $\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\delta}}$ and $\hat{\sigma}^2$ are estimated and then used for computing the posterior covariance and mean given by

$$\boldsymbol{R} = \left(\sigma^{-2}\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{A}\right)^{-1} \text{ and } \boldsymbol{m} = \sigma^{-2}\boldsymbol{R}\boldsymbol{X}^T\boldsymbol{y},$$

where $\boldsymbol{A} = diag\left(\hat{\beta}_1, ..., \hat{\beta}_{N+1}, \hat{\delta}_1, ..., \hat{\delta}_N\right)$. Finally, prediction is accomplished, using the covariance and mean of the posterior distribution, for the parameter part $\underline{\boldsymbol{\theta}}$ of $\underline{\boldsymbol{z}}$, i.e., $R_{\underline{\boldsymbol{\theta}}} = \boldsymbol{R}_{\{1:N+1,1:N+1\}}$ and $\boldsymbol{m}_{\underline{\boldsymbol{\theta}}} = \boldsymbol{m}_{\{1:N+1\}}$. The difference of the scheme, related to the classic RVM formulation, is that instead of inferring just the parameter vector $\underline{\boldsymbol{\theta}}$ to the RVM algorithm, it infers the joint parameter-outlier vector $\underline{\boldsymbol{z}}$. This is accomplished by replacing the matrix $[\boldsymbol{K}\ \boldsymbol{1}]$ with the matrix $[\boldsymbol{K}\ \boldsymbol{1}\ \boldsymbol{I}_N]$.

4 Kernel Greedy Algorithm for Robust Denoising (KGARD)

4.1 Motivation and Proposed Scheme

Our proposed scheme, alternates between a regularized Least Squares step and an OMP selection step based on the residual. It is well known that raw residuals can fail to detect outliers at leverage points; this is also known as swamping and masking of the outliers, [4]. In [40], for the linear case of the robust regression task, it is shown that the method successfully identifies all possible outliers under sufficient conditions (bounds). Our analysis there is based on the smallest principal angle between the regression subspace and all possible outlier subspaces. In practise, this condition eliminates occurrences of leverage points. However, such a condition cannot be applied to the non-linear case, which is our focus in the current manuscript. To this end the following discussion is of interest.

The i-th residual for the (non regularized) Least Squares step according to the model in (5) and regardless of the statistics of the joint noise vector $\boldsymbol{v}$, is written as

$$r_i = (1 - h_{ii})y_i - \sum_{\substack{j=1 \\ j\neq i}}^{N} h_{ij}y_j, \tag{13}$$

where $\boldsymbol{H} = \boldsymbol{X}_{(0)}(\boldsymbol{X}_{(0)}^T\boldsymbol{X}_{(0)})^{-1}\boldsymbol{X}_{(0)}^T$ is the hat matrix, [4]. From (13), it is evident that the diagonal of the hat matrix (with values between $1/N$ and 1) contains extremely useful information. More importantly, it characterizes whether or not an outlier in the observations is detectable via the Least Squares solution residual. If h_{ii} tends to 1, the evaluation can be misleading. To this end, in the the chapter dealing with the asymptotics of robust regression estimates in [4], the diagonal elements of the hat matrix $\boldsymbol{H}$ are assumed to be uniformly small, i.e., $\max_{1\leq i\leq N} h_{ii} = h << 1$. Furthermore, by applying the *SVD* decomposition of matrix $\boldsymbol{X}_{(0)}$, i.e., $\boldsymbol{X}_{(0)} = \boldsymbol{Q}\boldsymbol{S}\boldsymbol{V}^T$, we notice that $\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{Q}^T$. Thus, it seems natural that such an assumption could also be adopted for the non-linear case. However, since at the first step (and also at every next step) of the non-linear task a regularized Least Squares task is solved the $\boldsymbol{H}$ matrix is replaced by

$$\tilde{\boldsymbol{H}} = \boldsymbol{X}_{(0)}(\boldsymbol{X}_{(0)}^T\boldsymbol{X}_{(0)} + \lambda\boldsymbol{I}_{N+1})^{-1}\boldsymbol{X}_{(0)}^T = \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T, \tag{14}$$

where $\boldsymbol{G}$ is a diagonal matrix with elements $g_{ii} = \frac{\sigma_i^2}{\sigma_i^2+\lambda}$, $\lambda > 0$ is the regularization parameter and σ_i the $i-$th singular value of the matrix $\boldsymbol{X}_{(0)}$. Thus, this leads to an expression similar to (13), simply by replacing matrix $\boldsymbol{H}$, by $\tilde{\boldsymbol{H}}$. Hence, it is a matter of simple manipulations to establish from (14), for the diagonal elements of the new hat matrix, that it satisfies

$$\tilde{h}_{ii} = \frac{\sigma_i^2}{\sigma_i^2 + \lambda}h_{ii}. \tag{15}$$

Equation (15) is of great importance, since it guarantees that $\tilde{h}_{ii} < h_{ii}$ for any $\lambda > 0$. Furthermore, as $\lambda \to 0$ the Least Squares solution residual tends to become more sensitive to the correct detection of an outlier, while as $\lambda \to \infty$ then $\tilde{h}_{ii} \to 0$ and thus occurrences of leverage points tend to disappear. In simple words, the regularization performed on the specific task guards the method against occurrences of leverage points. Of course, this fact alone does not guarantee that one could safely detect an outlier via the residual. This is due to the following two reasons: a) the outliers values could be too small (engaging with the inlier noise) or b) the fraction of outliers contaminating the data could be enormously large; in such cases the summation term in (13) could easily be the dominant one. Based on the previous discussion, for the rest of the paper, we adopt the assumptions that the outliers are relatively few (the vector $\boldsymbol{u}$ is sparse) and also that the outlier values are (relatively) large. From a practical point of view, the latter assumption is natural, since we want to detect relatively large values of outlier noise. The first assumption is, also, in line with the use of the greedy approach. It is well established by now that greedy techniques work well for relatively small sparsity levels. These assumptions are also verified by the obtained experimental results.

In the following, we build upon the two formulations (7) and (8), that attempt (and indeed succeed) to solve the robust least squares task. Obviously, the difference lies solely on the regularization term. In the first approach, the regularization is performed using the ℓ_2-norm of the unkown kernel parameters (which is a standard regularization technique in linear methods). In contrast, in the alternative formulation we perform the regularization via the $\mathcal{H}$-norm of f. The reason for this modification was the improved performance obtained in practice via the first approach, as it will be demonstrated next.

Since both tasks in (7) and (8) are known to be NP-hard, a straight-forward computation of a solution seems impossible. However, under certain assumptions, greedy-based techniques often manage to provide accurate solutions to ℓ_0-norm minimization tasks, which are also guaranteed to be close to the optimal solution. The proposed KGARD algorithm, which is based on a modification of the popular Orthogonal Matching Pursuit (OMP), has been adapted to both formulations, i.e., (7) and (8).

First, one should notice that, the quadratic inequality constraint could also be written in a more compact form as follows:

$$J(\boldsymbol{z}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{z}\|_2^2 + \lambda \boldsymbol{z}^T \boldsymbol{B}\boldsymbol{z} \leq \varepsilon, \tag{16}$$

where

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{K} & \boldsymbol{1} & \boldsymbol{I}_N \end{bmatrix}, \ \boldsymbol{z} = \begin{pmatrix} \boldsymbol{\alpha} \\ c \\ \boldsymbol{u} \end{pmatrix}, \tag{17}$$

and for the choice of matrix $\boldsymbol{B}$ either one of the following matrices could be used,

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{I}_N & \boldsymbol{0} & \boldsymbol{O}_N \\ \boldsymbol{0}^T & 1 & \boldsymbol{0}^T \\ \boldsymbol{O}_N & \boldsymbol{0} & \boldsymbol{O}_N \end{bmatrix} \text{ or } \begin{bmatrix} \boldsymbol{K} & \boldsymbol{0} & \boldsymbol{O}_N \\ \boldsymbol{0}^T & 0 & \boldsymbol{0}^T \\ \boldsymbol{O}_N & \boldsymbol{0} & \boldsymbol{O}_N \end{bmatrix}, \tag{18}$$

depending on whether the model (7) or (8) is adopted, respectively.

The proposed method, as presented in Algorithm 4, attempts to solve the task (7) or (8), via a sparse greedy-based approach. The algorithm alternates between an LS task and a column selection step, that enlarges the solution subspace at each step, in order to minimize the residual error. The scheme shares resemblances to the OMP algorithm. Its main differences, are: (a) the solution of a regularized LS task at each iteration (instead of a simple LS task), i.e.,

$$\min_{\boldsymbol{z}} J_k(\boldsymbol{z}) = \min_{\boldsymbol{z}} \left\{ \|\boldsymbol{y} - \boldsymbol{X}_{\mathcal{S}_k}\boldsymbol{z}\|_2^2 + \lambda \boldsymbol{z}^T \boldsymbol{B}_{\mathcal{S}_k}\boldsymbol{z} \right\}, \tag{19}$$

and (b) the use of a specific initialization on the solution and the residual. These differences lead to a completely distinctive performance analysis for the method. The scheme is specified best, with the use of subsets, corresponding to a set of *active* and *inactive columns*, for any given

Algorithm 4 Kernel Greedy Algorithm for Robust Denoising: KGARD

```
procedure KGARD($\boldsymbol{K}$, $\boldsymbol{y}$, $\lambda$, $\epsilon$)
    $k \leftarrow 0$
    $\mathcal{S}_0 \leftarrow \{1, 2, ..., N+1\}$, $\mathcal{S}_0^c \leftarrow \{N+2, ..., 2N+1\}$
    $\hat{\boldsymbol{z}}_{(0)} \leftarrow \left(\boldsymbol{X}_{\mathcal{S}_0}^T \boldsymbol{X}_{\mathcal{S}_0} + \lambda \boldsymbol{B}_{\mathcal{S}_0}\right)^{-1} \boldsymbol{X}_{\mathcal{S}_0}^T \boldsymbol{y}$        ▷ Initial reg. least squares solution step.
    $\boldsymbol{r}_{(0)} \leftarrow \boldsymbol{y} - \boldsymbol{X}_{\mathcal{S}_0} \hat{\boldsymbol{z}}_{(0)}$
    while $\|\boldsymbol{r}_{(k-1)}\|_2 > \epsilon$ do
        $k \leftarrow k + 1$
        $j_k \leftarrow \arg\max_{j \in \tilde{\mathcal{S}}_k^c} |r_{(k-1),j}|$        ▷ Selection step ($\tilde{\mathcal{S}}_k^c$ is defined in (20)).
        $\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{j_k + N + 1\}$, $\mathcal{S}_k^c \leftarrow \mathcal{S}_k^c - \{j_k + N + 1\}$
        $\hat{\boldsymbol{z}}_{(k)} \leftarrow \left(\boldsymbol{X}_{\mathcal{S}_k}^T \boldsymbol{X}_{\mathcal{S}_k} + \lambda \boldsymbol{B}_{\mathcal{S}_k}\right)^{-1} \boldsymbol{X}_{\mathcal{S}_k}^T \boldsymbol{y}$        ▷ Reg. Least squares solution step.
        $\boldsymbol{r}_{(k)} \leftarrow \boldsymbol{y} - \boldsymbol{X}_{\mathcal{S}_k} \hat{\boldsymbol{z}}_{(k)}$
    Output: $\hat{\boldsymbol{z}}_{(k)} = \left(\hat{\boldsymbol{\alpha}}_{(k)}^T, \hat{c}_{(k)}, \hat{\boldsymbol{u}}_{(k)}^T\right)^T$ after $k$ iterations.
```

matrix. In particular, the $2N+1$ column vectors of the matrices $\boldsymbol{X}$ and $\boldsymbol{B}$ are divided into two complementary subsets: the active set, $\mathcal{S}_k$, which contains the indices of the active columns of the matrix at step k, and the inactive set, $\mathcal{S}_k^c$, which contains the remaining ones, i.e., those that do not participate in the representation. Thus, $\boldsymbol{X}_{\mathcal{S}_k}$ and $\boldsymbol{B}_{\mathcal{S}_k}$ denote the column vectors of matrices $\boldsymbol{X}$ and $\boldsymbol{B}$, respectively, restricted over the subset $\mathcal{S}_k$. Moreover, we define the set of indices

$$\tilde{\mathcal{S}}_k^c := \{i - N - 1 |\ i \in \mathcal{S}_k^c\}, \tag{20}$$

which is very helpfull for the description of the proposed method. While the set $\mathcal{S}_k^c$ refers to the columns of the augmented matrix $\boldsymbol{X}$, the set $\tilde{\mathcal{S}}_k^c$ refers to the columns of the identity matrix (the last part of matrix $\boldsymbol{X}$), i.e., matrix $\boldsymbol{I}_N$. In other words, $\tilde{\mathcal{S}}_k^c$ originates by subtracting the value $N+1$ from each one of the elements of $\mathcal{S}_k^c$. Initially, only the first $N+1$ columns of matrices $\boldsymbol{X}$ and $\boldsymbol{B}$, have been activated. Thus, $k = 0$, leads to the initialization of the active set $\mathcal{S}_0 = \{1, 2, \ldots, N+1\}$ with the corresponding matrices:

$$\boldsymbol{X}_{\mathcal{S}_0} = [\boldsymbol{K}\ \boldsymbol{1}],$$

and

$$\boldsymbol{B}_{\mathcal{S}_0} = \boldsymbol{I}_{N+1} \text{ or } \begin{bmatrix} \boldsymbol{K} & \boldsymbol{0} \\ \boldsymbol{0}^T & 0 \end{bmatrix},$$

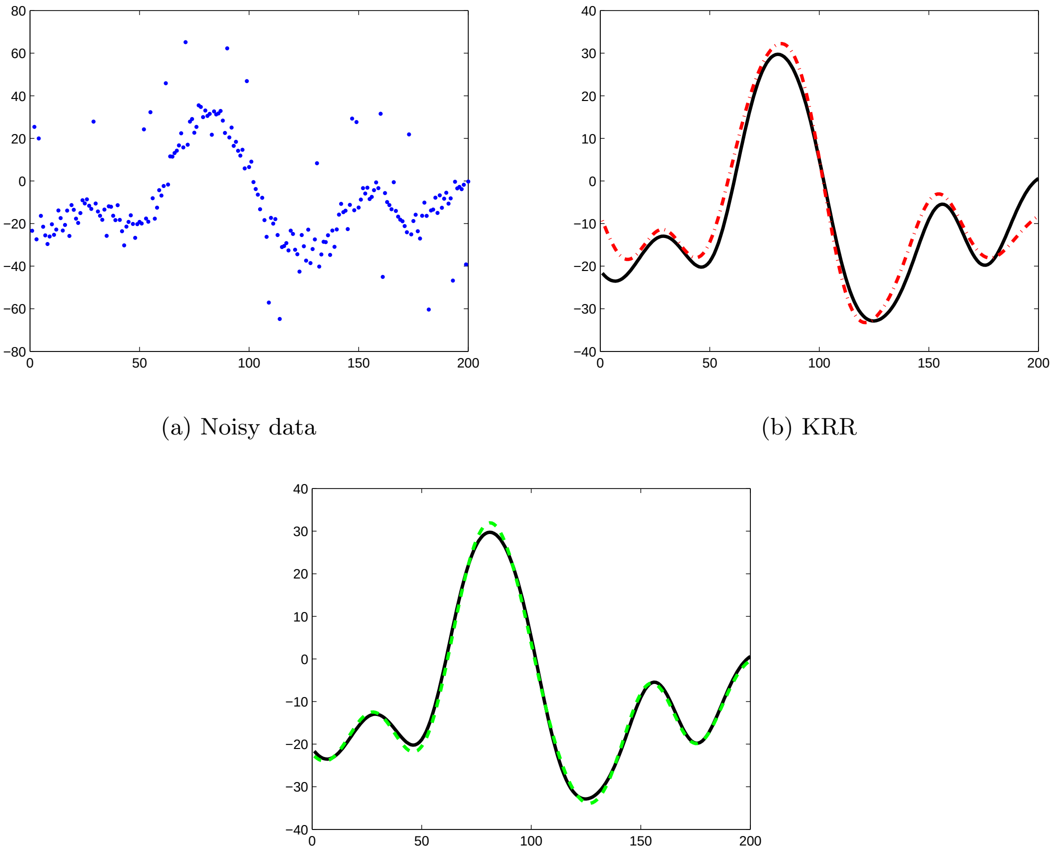
depending on the model selection, i.e., (7) or (8) respectively. Hence, the solution to the initial LS problem, is given by

$$\hat{\boldsymbol{z}}_{(0)} \ := \ \arg\min_{\boldsymbol{z}}\{J_0(\boldsymbol{z})\} \ = \ \left(\boldsymbol{X}_{\mathcal{S}_0}^T \boldsymbol{X}_{\mathcal{S}_0} + \lambda \boldsymbol{B}_{\mathcal{S}_0}\right)^{-1} \boldsymbol{X}_{\mathcal{S}_0}^T \boldsymbol{y}.$$

Next, the method computes the residual $\boldsymbol{r}_{(0)} = \boldsymbol{y} - \boldsymbol{X}_{\mathcal{S}_0} \hat{\boldsymbol{z}}_{(0)}$ and identifies an outlier[2], as the largest value of the residual vector. The corresponding index, say $j_1 \in \tilde{\mathcal{S}}_k^c$, is added into the set of active columns, i.e., $\mathcal{S}_1 = \mathcal{S}_0 \cup \{j_k + N + 1\}$. Thus, the matrix $\boldsymbol{X}_{\mathcal{S}_0}$ is augmented by a column, drawn from matrix $\boldsymbol{I}_N$, forming matrix $\boldsymbol{X}_{\mathcal{S}_1}$. Accordingly, the matrix $\boldsymbol{B}_{\mathcal{S}_0}$ is augmented by a zero row and a zero column, forming $\boldsymbol{B}_{\mathcal{S}_1}$. The new LS task is solved again (using matrices $\boldsymbol{X}_{\mathcal{S}_1}$, $\boldsymbol{B}_{\mathcal{S}_1}$) and a new residual $\boldsymbol{r}_{(1)}$ is computed. The process is repeated, until the residual drops below a predefined threshold.

The gains of the robust estimation (with KGARD) over the standard KRR task are demonstrated in the following pilot experiment. We consider our input data as 400 equidistant points over the interval $[0, 1)$ and generate our uncorrupted data with a non-linear function $\underline{f} \in \mathcal{H}$ as a (sparse) linear combination of Gaussian kernels with $\sigma = 0.1$ centered at a small number (i.e., $8 - 35$) of those points (randomly selected). Next, the data is separated into two subsets, the

[2] If outliers are not present the algorithm terminates and no outlier estimate exists in the solution $\hat{\boldsymbol{z}}_{(0)}$.



(a) Noisy data

(b) KRR

(c) KGARD

Figure 1: The significance of robust estimation: (a) Data corrupted by both inlier and 10% of outlier noise. (b) The black and the red dashed lines, correspond to the uncorrupted data and the non-robust estimation performed, respectively, over the training set with $MSE_{tr} = 10.79$. (c) The black and the green dashed lines, correspond to the uncorrupted data and the robust estimation performed with KGARD, respectively, over the training set with $MSE_{tr} = 1.21$.

training and the validation (testing) subset. The training subset consists of the 200 odd indexed points of the entire set (first, third, e.t.c.) and the validation subset includes the remaining ones (even indices). The noisy data, emerge from (4), where η corresponds to white Gaussian noise with $\sigma = 4$ and $\underline{u}_i$ to outlier values ± 40 at a fraction of 10% (uniformly distributed over the support set). Finally, the MSE is measured over 1000 “Monte Carlo” runs (independent experiments) for both the training and the validation set (more details on the experiment can be found in section 6.2).

In Figure 1(a), we have plotted the noisy data (blue dots) of the training set (for a specific simulation), which is generated via (4). The red dashed line in Figure 1(b) corresponds to the estimation performed by a simple KRR task; the disadvantage of not performing robust estimation is clear. On the other hand, in Figure 1(c), the advantages of the robust estimation performed with KGARD are depicted (for either choice of matrix $\boldsymbol{B}$); it is evident that, the estimation has not been affected by the presence of outliers. Although, both approaches, (7) and (8), are suitable for dealing with the sparse minimization task, in practise the selection of (7) proves to be better choice. In order to justify our claim, we performed the following evaluation. The MSE attained via the $\mathcal{H}$-norm regularization, is $MSE = 1.35$ for both the training and the validation set. However, when performing the estimation with the ℓ_2-norm regularization, the respective value for both the training and the validation set is *reduced* to $MSE = 1.21$; that is, the performance is improved by 10.4%. To this end, in the future, the model (7) is adopted and thus the respective $\boldsymbol{B}$ matrix is used. Finally, it should be noted that all λ and ϵ parameters have been optimized accordingly.

Remark 1. *In order to simplify the notation, in the next sections, we adopt* $\boldsymbol{X}_{(k)}$ *and* $\boldsymbol{B}_{(k)}$ *to refer to the matrices* $\boldsymbol{X}_{\mathcal{S}_k}$ *and* $\boldsymbol{B}_{\mathcal{S}_k}$ *at the* k *step.*

Remark 2. *Once a column has been selected at the* k *step, it cannot be selected again in any subsequent step, since the corresponding residual coordinate is zero. In other words, the algorithm always selects a column from the last part of* $\boldsymbol{X}$*, i.e., matrix* $\boldsymbol{I}_N$*, that is not included in* $\mathcal{S}_k$*.*

4.2 Efficient Implementations

Since the outliers often comprise a small fraction of the data set, i.e., $k << N$, this leads to a fast implementation time for OMP-like schemes such as KGARD. Initially, the inversion of matrix $\boldsymbol{X}_{(0)}^T\boldsymbol{X}_{(0)} + \lambda\boldsymbol{B}_{(0)}$ plus the multiplication of $\boldsymbol{X}_{(0)}^T\boldsymbol{y}$, requires $\mathrm{O}\left((N+1)^3\right)$ flops. At each one of the subsequent steps, the required complexity is $\mathrm{O}\left((N+k+1)^3\right)$, while the total cost for the method is $\mathrm{O}\left((N+1)^3(k+1) + (5/2)N^2k^2 + (4/3)Nk^3 + k^4/4\right)$, which is acceptable, since $k << N$ is assumed. However, the complexity of the method could be further reduced, since a large part of the inverted matrix remains unchanged. To this end, several methods could be employed, e.g., [41].

The first technique, which has been applied to the proposed scheme, is the *matrix inversion lemma* (MIL). The initial computational cost requirement is cubic, due to the inversion of the matrix

$$\boldsymbol{M}_{(0)} := \boldsymbol{X}_{(0)}^T\boldsymbol{X}_{(0)} + \lambda\boldsymbol{B}_{(0)} = \begin{bmatrix} \boldsymbol{K}^T\boldsymbol{K} + \lambda\boldsymbol{I}_N & \boldsymbol{K}^T\boldsymbol{1} \\ \boldsymbol{1}^T\boldsymbol{K} & N+\lambda \end{bmatrix}. \tag{21}$$

In the subsequent steps, the column vector $\boldsymbol{e}_{j_k}$ is selected from matrix $\boldsymbol{I}_N$ and the new matrix needs to be inverted. However, with the application of MIL, the inversion at each step is avoided. Instead, we compute $\boldsymbol{M}_{(k)}$ and its inverse recursively, i.e.,

$$\boldsymbol{M}_{(k)} := \begin{bmatrix} \boldsymbol{M}_{(k-1)} & \boldsymbol{X}_{(k-1)}^T\boldsymbol{e}_{j_k} \\ \boldsymbol{e}_{j_k}^T\boldsymbol{X}_{(k-1)} & 1 \end{bmatrix}.$$

Thus, the required operations are $\mathrm{O}\left((N+1+k)^2\right)$ per iteration, with a total cost of

$$\mathrm{O}\left(2N^3 + 2kN^2 + k^3/3 + (3/2)k^2N\right).$$

An alternative technique, which is even more efficient and is basically used throughout this paper, is the *Cholesky decomposition* for matrix $\boldsymbol{M}_{(k)}$. This is summarized in the following steps:

- *Replace* the initial reg. least squares solution step 4 of algorithm 4, with:

  Factorization step: $\boldsymbol{M}_{(0)} = \boldsymbol{L}_{(0)}\boldsymbol{L}_{(0)}^T$

  Solve $\boldsymbol{L}_{(0)}\boldsymbol{L}_{(0)}^T\boldsymbol{z} = \boldsymbol{X}_{(0)}^T\boldsymbol{y}$ using:
  - forward substitution $\boldsymbol{L}_{(0)}\boldsymbol{q} = \boldsymbol{X}_{(0)}^T\boldsymbol{y}$
  - backward substitution $\boldsymbol{L}_{(0)}^T\boldsymbol{z} = \boldsymbol{q}$

  Complexity: $\mathrm{O}\left((N+1)^3/3 + (N+1)^2\right)$

  and the regularized Least squares solution step 10 of algorithm 4, with:

  Compute $\boldsymbol{d}$ such that: $\boldsymbol{L}_{(k-1)}\boldsymbol{d} = \boldsymbol{X}_{(k-1)}^T\boldsymbol{e}_{j_k}$

  Compute: $b = \sqrt{1 - ||\boldsymbol{d}||_2^2}$

  Matrix Update: $\boldsymbol{L}_{(k)} = \begin{bmatrix} \boldsymbol{L}_{(k-1)} & \boldsymbol{0} \\ \boldsymbol{d}^T & b \end{bmatrix}$

Solve $\boldsymbol{L}_{(k)}\boldsymbol{L}_{(k)}^T\boldsymbol{z} = \boldsymbol{X}_{(k)}^T\boldsymbol{y}$ using:

- forward substitution $\boldsymbol{L}_{(k)}\boldsymbol{p} = \boldsymbol{X}_{(k)}^T\boldsymbol{y}$
- backward substitution $\boldsymbol{L}_{(k)}^T\boldsymbol{z} = \boldsymbol{p}$

Complexity: $\mathrm{O}\left((9/2)N^2 + 5Nk + (3/2)k^2\right)$ per iteration.

Employing the Cholseky decomposition plus the update step leads to a reduction of the total computational cost to $\mathrm{O}\left((N+1)^3/3 + (N+1)^2 + k^3/2 + (5/2)Nk^2\right)$, which is the faster implementation for this problem (recall that $k << N$).

4.3 Further Improvements on KGARD's Performance

In order to simplify the theoretical analysis and reduce the corresponding equations, the proposed algorithm employs the same regularization parameter for all kernel coefficients. However, one may employ a more general scheme as follows:

$$\begin{array}{cc} \min\limits_{\boldsymbol{u},\boldsymbol{a}\in\mathbb{R}^N, c\in\mathbb{R}} & \|\boldsymbol{u}\|_0 \\ \text{subject to} & \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{a} - c\boldsymbol{1} - \boldsymbol{u}\|_2^2 + \|\boldsymbol{\Psi}\boldsymbol{a}\|_2^2 + \lambda c^2 \leq \varepsilon, \end{array} \tag{22}$$

where $\boldsymbol{\Psi}$ is a more general regularization matrix (Tikhonov matrix). For example, as the accuracy of kernel based methods usually drops near the border of the input domain, it is reasonable to increase the regularization effect at these points. This can be easily implemented by employing a diagonal matrix with positive elements on the diagonal and increase the regularization factors that correspond to the points near the border. This is demonstrated in the experimental section 6.

5 Theoretical Analysis

In the current section, we study the theoretical properties of the proposed robust kernel regression method, i.e., KGARD. Firstly, we establish that the method always converges in finite time and that the reconstruction error of the method is *strictly decreasing*. Next, we provide the necessary conditions so that the proposed method succeeds in *identifying first* the locations of all the outliers, for the case where only outliers exist in the noise. The derived theoretical conditions for the second part (i.e., the outlier identification) are rather tight. However, as demonstrated in the experiments, the method achieves to recover the correct support of the sparse outlier vector in many cases where the theoretical result doesn't hold. This leads to the conclusion that the provided conditions can be loosen up significantly in the future. Moreover, in practice, where inlier noise also exists, the method succeeds to correctly identify the majority of the outliers. The reason that, the analysis is carried out for the case where inlier noise is not present, is due to the fact that the analysis gets highly involved. The absence of the inlier noise makes the analysis easier and it highlights some theoretical aspects on why the method works. It must be emphasized that, such a theoretical analysis is carried out for the first time and it is absent in the previously published works.

5.1 Convergence Analysis

Our main focus in this section is to examine some important properties of the proposed algorithmic scheme. Firstly, we discuss the convergence of the algorithm. In particular, it is easy to check that the proposed algorithm will always converge in finite time. Indeed, assuming the worst case scenario, where the algorithm continues until all columns of $\boldsymbol{I}_N$ are selected, we can easily see that the norm of the residual vector will eventually drop below ϵ. Of course, this is something that occurs in the case where the parameter ϵ is set extremely low. As a consequence, the procedure will continue and model all noise samples (even those originating from a Gaussian source) as impulses, filling up the vector $\boldsymbol{u}$ and producing a residual vector equal to $\boldsymbol{0}$. Obviously, if ϵ is carefully tuned and the outliers are sufficiently sparse, the algorithm will stop well before that. Hence, sensible tuning of ϵ should be applied.

Moreover, note that, for all $\varepsilon \geq 0$, there exists $\boldsymbol{z}$ such that $J_k(\boldsymbol{z}) \leq \varepsilon$. This implies that the feasible set of (7) is always nonempty[3]. It is straightforward to prove that the set of *normal equations*, obtained from the minimization of (19), at step k, is

$$(\boldsymbol{X}_{(k)}^T \boldsymbol{X}_{(k)} + \lambda \boldsymbol{B}_{(k)})\boldsymbol{z} = \boldsymbol{X}_{(k)}^T \boldsymbol{y},$$

where $(\boldsymbol{X}_{(k)}^T \boldsymbol{X}_{(k)} + \lambda \boldsymbol{B}_{(k)})$ is invertible, i.e., (19) has a unique minimum, for all k.

Lemma 1. *The matrix $\boldsymbol{M}_{(k)} := \boldsymbol{X}_{(k)}^T \boldsymbol{X}_{(k)} + \lambda \boldsymbol{B}_{(k)}$ is (strictly) positive definite, hence invertible.*

Proof. Consider a non-zero vector $\boldsymbol{z} \in \mathbb{R}^{2N+1}$ so that $\boldsymbol{z} = \left(\boldsymbol{\alpha}^T, \beta, \boldsymbol{\gamma}^T\right)^T$ is decomposed such that $\boldsymbol{\alpha} \in \mathbb{R}^N$, $\beta \in \mathbb{R}$ and $\gamma \in \mathbb{R}^k$. Then it is easy to show that

$$\boldsymbol{z}^T \boldsymbol{M}_{(k)} \boldsymbol{z} = \|\boldsymbol{K}\boldsymbol{\alpha} + \beta \boldsymbol{1} + \boldsymbol{I}_{\mathcal{S}_k}\boldsymbol{\gamma}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_2^2 + \lambda\beta^2 > 0,$$

which implies that $\boldsymbol{M}_{(k)}$ is a (strictly) positive definite matrix. □

Alternatively, one could express (19) as follows[4]:

$$\min_{\boldsymbol{z}} \quad J_k(\boldsymbol{z}) = \left\| \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{0} \end{pmatrix} - \boldsymbol{D}_{(k)}\boldsymbol{z} \right\|_2^2, \tag{23}$$

where $\boldsymbol{D}_{(k)} = \begin{bmatrix} \boldsymbol{X}_{(k)} \\ \sqrt{\lambda}\boldsymbol{B}_{(k)} \end{bmatrix}$. Problem (23) has a unique solution, if and only if the nullspaces of $\boldsymbol{X}_{(k)}$ and $\boldsymbol{B}_{(k)}$ intersect only trivially, i.e., $\mathcal{N}(\boldsymbol{X}_{(k)}) \cap \mathcal{N}(\boldsymbol{B}_{(k)}) = \{0\}$ [42, 43]. Hence, $\boldsymbol{M}_{(k)}$ is (strictly) positive definite, as the columns of $\boldsymbol{D}_{(k)}$ are linearly independent, and the minimizer $\boldsymbol{z}_* \in \mathbb{R}^{2N+1}$ of (19), is unique [2].

Furthermore, similarly to the discussion in Section 3, an equivalent formulation for (19) is

$$\min_{\boldsymbol{z}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{z}||_2^2, \text{ s.t. } ||\boldsymbol{B}\boldsymbol{z}||_2 \leq \delta, \tag{24}$$

for some $\delta > 0$. In (24), the regularization term is replaced by a quadratic constraint. Equivalence between (19) and (24) has been well studied and established [44]. The reason for resorting to the latter formulation is to be used for the proof of the following lemma.

Lemma 2. *The norm of the residual for KGARD is strictly decreasing.*

Proof. Recall that during initialization, KGARD sets $\mathcal{S}_0$ to include only the first $N+1$ columns of matrices $\boldsymbol{X}$, $\boldsymbol{B}$ and let $\hat{\boldsymbol{z}}_{(0)}$ denote the initial solution of (24) and $\boldsymbol{r}_{(0)} = \boldsymbol{y} - \boldsymbol{X}_{(0)}\hat{\boldsymbol{z}}_{(0)}$ be the initial residual. Since our goal is to remove the unknown/unwanted additive noise, it is expected that $\boldsymbol{y} \notin \mathcal{R}(\boldsymbol{X}_{(0)})$ (the range of the matrix), regardless of the statistics of the additive noise (Gaussian, impulse or both). Suppose, now, that the ℓ_2 norm of the residual $\boldsymbol{r}_{(0)}$ is below our threshold parameter ϵ. In this case, the method is forced to stop; either no outlying values are identified or the threshold parameter is tuned extremely high. Nevertheless, the case of greater importance, is when outliers are present; the algorithm continues expanding the set of active columns with columns from the identity matrix and thus a sparse outlier vector is generated.

At each subsequent iteration, k, the algorithm selects an index from the set $\mathcal{S}_{k-1}^c$ of *inactive* columns from matrix $\boldsymbol{X}$. Then, $\mathcal{S}_{k-1}$ is enlarged by the selected index (say j_k) and the matrix $\boldsymbol{X}_{(k-1)}$ is augmented by the column vector $\boldsymbol{e}_{j_k}$ forming $\mathcal{S}_k$ and $\boldsymbol{X}_{(k)}$, respectively. Finally, the solution $\hat{\boldsymbol{z}}_{(k)} \in \mathbb{R}^{N+k+1}$ and the residual $\boldsymbol{r}_{(k)} = \boldsymbol{y} - \boldsymbol{X}_{(k)}\hat{\boldsymbol{z}}_{(k)}$ are computed, respectively, by

[3] For example, if we select $\boldsymbol{z} = \left(\boldsymbol{0}^T, 0, \boldsymbol{y}^T\right)^T$, then $J_k(\boldsymbol{z}) = 0$.

[4] Notice that $\boldsymbol{B}$ is a projection matrix (this holds only for the regularization performed with the ℓ_2 norm).

solving (24) (step 10 of the algorithm). At $k+1$ step, the process is repeated and the matrices are augmented. At this stage we have,

$$\boldsymbol{X}_{(k+1)} = [\boldsymbol{X}_{(0)}\ \boldsymbol{e}_{j_1}\ \cdots\ \boldsymbol{e}_{j_k}\ \boldsymbol{e}_{j_{k+1}}] = [\boldsymbol{X}_{(k)}\ \boldsymbol{e}_{j_{k+1}}].$$

Now, let $\hat{\boldsymbol{z}}_{(k+1)} \in \mathbb{R}^{N+k+2}$ be the unique minimizer of $L_{k+1}(\boldsymbol{z}) = ||\boldsymbol{y} - \boldsymbol{X}_{(k+1)}\boldsymbol{z}||_2^2$ subject to the constraint $\|\boldsymbol{B}_{(k+1)}\boldsymbol{z}\|_2 \leq \delta$, i.e., the minimization of (24) at the $k+1$ step. Also let $\mathsf{z}_{(k+1)} = (\hat{\boldsymbol{z}}_{(k)}^T, r_{(k),j_{k+1}})^T$. Observe that $\mathsf{z}_{(k+1)}$ belongs to the feasible set defined by the inequality constraint of (24) at the current step[5], and hence $L_{k+1}(\hat{\boldsymbol{z}}_{(k+1)}) \leq L_{k+1}(\mathsf{z}_{(k+1)})$. Moreover, we have that

$$\begin{aligned} L_{k+1}(\mathsf{z}_{(k+1)}) &= \left\| \boldsymbol{y} - \begin{bmatrix} \boldsymbol{X}_{(k)} & \boldsymbol{e}_{j_{k+1}} \end{bmatrix} \cdot \begin{pmatrix} \hat{\boldsymbol{z}}_{(k)} \\ r_{(k),j_{k+1}} \end{pmatrix} \right\|_2^2 \\ &= \left\| \boldsymbol{y} \ - \ \boldsymbol{X}_{(k)}\hat{\boldsymbol{z}}_{(k)} \ - \ r_{(k),j_{k+1}}\boldsymbol{e}_{j_{k+1}} \ \right\|_2^2 \\ &= \left\| \boldsymbol{r}_{(k)} \ - \ r_{(k),j_{k+1}}\boldsymbol{e}_{j_{k+1}} \right\|_2^2 < \left\| \boldsymbol{r}_{(k)} \right\|_2^2, \end{aligned} \tag{25}$$

where the last strict inequality is due to the fact that $|r_{(k),j_{k+1}}| > 0$ (if $r_{(k),j_{k+1}} = 0$, then $\boldsymbol{r}_{(k)}$ is a zero vector, since its maximum value is 0 and the algorithm should have been terminated at iteration k). Thus, we conclude that

$$\|\boldsymbol{r}_{(k+1)}\|_2^2 = L_{k+1}(\hat{\boldsymbol{z}}_{(k+1)}) \leq L_{k+1}(\mathsf{z}_{(k+1)}) < \|\boldsymbol{r}_{(k)}\|_2^2,$$

which proves the claim. □

Remark 3. *It should be noted that, despite the fact that the error is strictly decreasing, there are no theoretical guarantees on the accuracy of the approximation. In other words, we cannot be sure whether our solution is good or bad, even for the case where only outliers exist in the noise. However, in practise, extended experimentation have shown that the approximation is fairly good, even in the existence of both inlier and outlier noise.*

5.2 Identification of the Outliers for the Noiseless Case

The following theorem establishes a bound on the largest singular value of matrix $\boldsymbol{X}_0$, which guarantees that the method first identifies the correct locations of all the outliers, for the case where only outliers exist in the noise. However, since the ϵ parameter controls the number of iterations, for which the method identifies an outlier, it is not guaranteed that it will stop, once all the outliers are identified, unless the correct value is somehow given. Thus, it is possible that a few other locations, that do not correspond to outliers, are also identified. Notable is also the fact that, such a result has never been established before by other comparative methods.

Theorem 3. *Let $\boldsymbol{K}$ be a full rank, square, real valued matrix. Suppose, that*

$$\boldsymbol{y} = [\boldsymbol{K}\ \boldsymbol{1}] \underbrace{\begin{pmatrix} \underline{\boldsymbol{\alpha}} \\ \underline{c} \end{pmatrix}}_{\underline{\boldsymbol{\theta}}} + \underline{\boldsymbol{u}},$$

where $\underline{\boldsymbol{u}}$ is a sparse (outlier) vector. KGARD is guaranteed to identify first the correct locations of all the outliers[6]*, if the maximum singular value of matrix $\boldsymbol{X}_{(0)} := [\boldsymbol{K}\ \boldsymbol{1}]$, satisfies:*

$$\sigma_M(\boldsymbol{X}_{(0)}) < \gamma\sqrt{\lambda}, \tag{26}$$

where

$$\gamma = \sqrt{\frac{\min|\underline{u}| - \sqrt{2\lambda}||\underline{\boldsymbol{\theta}}||_2}{2||\underline{\boldsymbol{u}}||_2 - \min|\underline{u}| + \sqrt{2\lambda}||\underline{\boldsymbol{\theta}}||_2}}, \tag{27}$$

[5] Geometrically the feasible set remains the same, while matrix $\boldsymbol{B}$ is augmented by zero elements at each step.

[6] However, the theorem does not guarantee that only the locations of the outliers will be identified. If the value of ϵ is too small, then KGARD will next identify locations that do not correspond to true outlier indices.

$\min|\underline{u}|$ *is the smallest absolute value of the sparse vector over the non-zero coordinates and* $\lambda > 0$ *is a sufficiently large*[7] *regularization parameter for KGARD.*

The proof is presented in the Appendix section.

Careful tuning of the ϵ parameter seems to play an important role regarding the performance of KGARD. This is the user-defined parameter, that controls the number of iterations for the method (thus the convergence speed) and also the sparsity for the outlier estimate vector. Assuming that the ϵ value is set to a relatively small value, the algorithm will first select the correct locations of the outliers and then continue until all columns of $\boldsymbol{I}_N$ are selected (in such case $k = N$ is the maximum number of iterations for KGARD). Consequently, we can easily see that the norm of the residual vector will eventually drop below $\epsilon > 0$ (and if all columns are selected $\boldsymbol{r}_{(k)} = \boldsymbol{0}$). Simply stated, the procedure will continue and model other samples (not originating from an outlier noise source) as outliers filling up the outlier estimate vector $\hat{\boldsymbol{u}}$, which will no longer be sparse. On the other hand, if ϵ is set to relatively large values, the algorithm will stop within a few only iterations, which leads to the identification of only a few of the true outliers in the dataset. Hence, sensible tuning of ϵ should be applied. Finally, it should be stated that the algorithm is not very sensitive to the choice of ϵ, i.e., small changes in its value do not affect much the sparsity level of the outlier estimate.

Remark 4. *The theorem does not guarantee that only the locations of the true outliers will be identified. If the value of is too small, then KGARD once it identifies the location of the true outliers, it will next identify locations that do not correspond to outlier indices.*

6 Experiments

For the entire section of experiments, the Gaussian (RBF) kernel is employed and all results are averaged over 1000 "Monte Carlo" runs (independent simulations). At each experiment, the parameters are optimized (via cross-validation) and the respective parameter values are given (for each method), so that results are reproducible. The specific (MATLAB) code can be found in `http://bouboulis.mysch.gr/kernels.html`.

6.1 Recovery of the Sparse Outlier Vector's Support

In the current section, our main concern is to test on the validity of the condition (26) in practise. To this end, we have performed the following experiment, for the case where only outliers exist in the noise.

We consider $N = 100$ equidistant points over the interval $[0, 1]$ and generate the output data via $\underline{f}(x_i) = \sum_{j=1}^{N} \underline{\alpha}_j \kappa(x_i, x_j)$, where κ is the Gaussian kernel with $\sigma = 0.1$ and the vector of coefficients $\underline{\boldsymbol{\alpha}} = [\underline{\alpha}_1, \ldots, \underline{\alpha}_N]$ is a sparse vector with the number of non-zero coordinates ranging between 2 and 23 and their values drawn from $\mathcal{N}(0, 0.5^2)$. Since no inlier noise exists, our corrupted data is given from (4) for $\eta_i = 0$ and outlier values $\pm\underline{u}$. Moreover, since the condition (26) is valid for fixed values of the parameters involved, we have measured the ability of KGARD to recover the support of the sparse outlier vector, i.e., $\mathcal{T} = \text{supp}(\underline{\boldsymbol{u}})$, while varying the values of the outliers. In Figure 2, the ability of KGARD to identify the exact sparse outlier vector support is demonstrated, for a fraction of outliers at 10%. On the vertical axis we have measured the percentage of correct and wrong indices recovered, while varying the value u of the outliers. In parallel, the bar chart demonstrates the validity of the introduced condition (26). It is clear that, if the condition holds, KGARD identifies the correct support of the sparse outlier vector successfully. However, even if the condition is rarely satisfied, e.g., for $\underline{u} = 100$, the method still manages to identify the correct support. This fact leads to the conclusion that the condition imposed by (26) is rather strict. This is in line with most sparse modeling related conditions, which, in practice, fall short in predicting the exact recovery conditions.

[7] Since the regularization parameter is defined by the user, we assume that such a value can be achieved, so that the γ parameter makes sense. More details can be found in the proof at the appendix section.

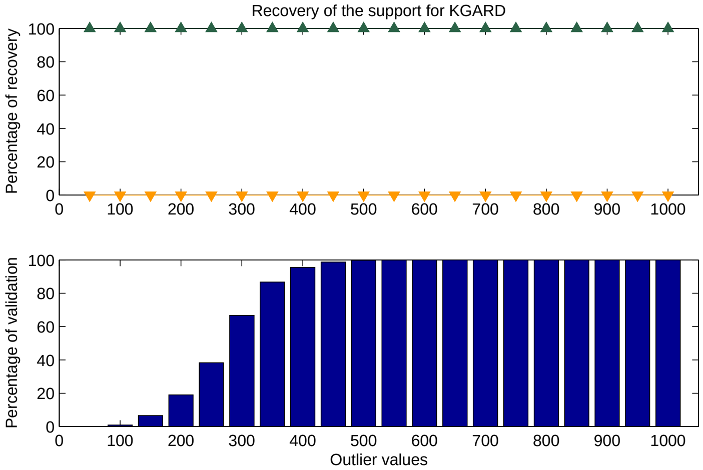


Figure 2: Percentage of the correct (green pointing up) and wrong (orange pointing down) indices that KGARD has identified, while varying the values $\pm \underline{u}$ of the outliers at the fixed fraction of 10%. Although the condition (26) is valid only for values greater than ±600 (and with high probability valid for values 400-599), the support of the sparse outlier vector has been correctly identified for much smaller values of outlier noise, too.

Outlier fraction	Correct support	Wrong support	Outlier value u
5 %	100 %	0 %	450
10 %	100 %	0 %	600
15 %	100 %	0 %	650
20 %	100 %	0 %	700
25 %	100 %	0 %	750
30 %	100 %	0 %	950

Table 1: Percentage of correct and wrong indices identified for all outlier values $\underline{u}$ ranging from 50 to 1000. The correct support corresponds to true outliers (indices in $\mathcal{T}$), while the wrong one corresponds to points which are wrongly classified as outliers (thus do not belong to $\mathcal{T}$). In the final column the minimum value $\underline{u}$ of outliers for which the support recovery condition is valid, is listed.

Finally, in Table 1, the previous experiment has been performed for various fractions of outliers. In the second and third column, we have listed the percentage of correct and wrong indices (truly) identified by the method, for all values of outliers ranging from 50 to 1000. Moreover, in the final column, the minimum value of outliers, which renders the condition valid, is shown. For example, in the second row and for 10% of outliers, the condition is valid only for values greater than 600 (last column of Table 1). However, the method manages to correctly identify the support (one-to-one index - columns two and three), not only for values $\underline{u}$ greater than 600, but for all outlier values, i.e, from the minimum value of 50 to the maximum value of 1000. It should also be noted that, experiments have been performed with the use of various non-linear functions (not only linear combinations of kernels) and results were similar to the ones presented here.

6.2 Evaluation of the Method: Mean-Square-Error (MSE)

In the current section, the previously established methods that deal with the non-linear robust estimation with kernels, i.e., the Bayesian approach RB-RVM and the weighted ℓ_1-norm approximation method (RAM), are compared against KGARD in terms of the mean-square-error (MSE) performance. Additionally, the evaluation is enhanced with a list of the percentage of the correct and wrong indices that each method has identified, for all methods except for the Bayesian

Method	MSE_{tr}	MSE_{val}	Cor. supp	Wr. supp	MIT (sec)	Inlier - Outlier
RB-RVM	0.0850	0.0851	-	-	0.298	20 dB - 5%
RAM ($\lambda = 0.07, \mu = 2.5$)	0.0344	0.0345	100 %	0.2 %	0.005	20 dB - 5%
KGARD ($\lambda = 0.2, \varepsilon = 10$)	**0.0285**	**0.285**	100 %	0 %	0.004	20 dB - 5%
RB-RVM	0.0911	0.0912	-	-	0.298	20 dB - 10%
RAM ($\lambda = 0.07, \mu = 2.5$)	0.0371	0.0372	100 %	0.1 %	0.007	20 dB - 10%
KGARD ($\lambda = 0.2, \varepsilon = 10$)	**0.0305**	**0.0305**	100 %	0 %	0.008	20 dB - 10%
RB-RVM	0.0992	0.0994	-	-	0.299	20 dB - 15%
RAM ($\lambda = 0.07, \mu = 2$)	0.0393	0.0393	100 %	0.6 %	0.008	20 dB - 15%
KGARD ($\lambda = 0.3, \varepsilon = 10$)	**0.0330**	**0.0330**	100 %	0 %	0.012	20 dB - 15%
RB-RVM	0.1189	0.1184	-	-	0.305	20 dB - 20%
RAM ($\lambda = 0.07, \mu = 2$)	**0.0421**	**0.0422**	100 %	0.4 %	0.010	20 dB - 20%
KGARD ($\lambda = 1, \varepsilon = 10$)	0.0626	0.0626	100 %	0 %	0.017	20 dB - 20%
RB-RVM	0.3630	0.3631	-	-	0.327	15 dB - 5%
RAM ($\lambda = 0.15, \mu = 5$)	0.1035	0.1036	100%	0.7 %	0.005	15 dB - 5%
KGARD ($\lambda = 0.3, \varepsilon = 15$)	**0.0862**	**0.0862**	100 %	0.1 %	0.005	15 dB - 5%
RB-RVM	0.3828	0.3830	-	-	0.319	15 dB - 10%
RAM ($\lambda = 0.15, \mu = 5$)	0.1117	0.1118	100%	0.4 %	0.006	15 dB - 10%
KGARD ($\lambda = 0.3, \varepsilon = 15$)	**0.0925**	**0.0925**	100 %	0 %	0.008	15 dB - 10%
RB-RVM	0.4165	0.4166	-	-	0.317	15 dB - 15%
RAM ($\lambda = 0.15, \mu = 5$)	0.1186	0.1186	100%	0.3 %	0.007	15 dB - 15%
KGARD ($\lambda = 0.3, \varepsilon = 15$)	**0.1001**	**0.1003**	100 %	0 %	0.012	15 dB - 15%
RB-RVM	0.4793	0.4798	-	-	0.312	15 dB - 20%
RAM ($\lambda = 0.15, \mu = 4$)	**0.1281**	**0.1282**	100%	1.4 %	0.008	15 dB - 20%
KGARD ($\lambda = 0.7, \varepsilon = 15$)	0.1340	0.1349	100 %	0 %	0.016	15 dB - 20%

Table 2: Computed MSE for $\underline{f}(x) = 20sinc(2\pi x)$ over the training and validation set, percentage of correct and wrong support recovered and mean implementation time (MIT), for each level of inlier noise and fraction of outliers.

approach (not provided directly by the RB-RVM method). Moreover, the *mean implementation time* (MIT) is measured for each experiment. Finally, following section 4.3, for the first two experiments we have increased the regularization value λ of KGARD near the edge points/borders, as a means to improve the performance. In particular, at the 5 first and 5 last points (borders), the regularizer is automatically multiplied by the factor of 5, with respect to the predefined value λ which is used on the interior points. The experiments are described in more detail next.

- For the first experiment, we have selected the *sinc* function, which is a popular one in machine learning. We consider 398 equidistant points over the interval $[-0.99, 1)$ for the input values and generated the uncorrupted output values via $\underline{f}(x_i) = 20sinc(2\pi x_i)$. Next, the set of points is split into two subsets, the training and the validation subset. The training subset, with points denoted by (y_i, x_i), consists of the $N = 199$ odd indexed points (first, third, e.t.c.), while the validation subset comprises the remaining points (denoted as (y_i', x_i')). The original data of the training set, is then contaminated by noise, as (4) suggests. The inlier part is considered to be random Gaussian noise of appropriate variance (measured in dB), while the outlier part consists of various fractions of outliers, with constant values ±15, distributed uniformly over the support set. Finally, the kernel parameter σ has been set equal to $\sigma = 0.15$.

  Table 2 depicts the performance of each method, where the best results are marked in **bold**. In terms of the computed MSE, it is clear that KGARD attains a lower MSE for both the training and the validation error for all fractions of outliers, except for the fraction of 20%. This fact is also aligned with the theoretical properties of the sparse greedy methods, since their performance boosts as the sparsity level of the approximation is low. On the other hand, the RAM solver seems more suitable for larger fractions of outliers. Moreover, the computational cost is comparable for both methods (RAM and KGARD), for small fractions of outliers. Regarding the identification of the sparse outlier vector support, although both methods correctly identify the indices that belong to the sparse outlier vector's support, i.e.,

Method	MSE_{tr}	MSE_{val}	Cor. supp	Wr. supp	MIT (sec)	Outliers
RB-RVM	3.3405	3.3436	-	-	0.309	5%
RAM ($\lambda = 0.15, \mu = 33$)	1.2459	1.2473	100%	0 %	0.005	5%
KGARD ($\lambda = 0.3, \varepsilon = 57$)	**1.1567**	**1.1580**	99.8 %	1.2 %	0.004	5%
RB-RVM	3.6111	3.6176	-	-	0.308	10%
RAM ($\lambda = 0.15, \mu = 31$)	1.3085	1.3100	100%	0.1 %	0.005	10%
KGARD ($\lambda = 0.3, \varepsilon = 55$)	**1.2110**	**1.2120**	99.9 %	0.9 %	0.008	10%
RB-RVM	3.7902	3.7950	-	-	0.308	15%
RAM ($\lambda = 0.15, \mu = 28$)	1.3945	1.3972	100%	0.2 %	0.006	15%
KGARD ($\lambda = 0.3, \varepsilon = 53$)	**1.2922**	**1.2942**	100 %	0.8 %	0.012	15%
RB-RVM	4.0685	4.0705	-	-	0.307	20%
RAM ($\lambda = 0.15, \mu = 24$)	**1.5110**	**1.5109**	100%	0.8 %	0.007	20%
KGARD ($\lambda = 0.3, \varepsilon = 52$)	1.5173	1.5262	99.9 %	0.4 %	0.016	20%

Table 3: Performance evaluation for each method, for the case where the input data lies on the 1-dimensional space and the output $\underline{f} \in \mathcal{H}$ is considered as a linear combination of a few kernels. The inlier noise is considered random Gaussian with $\sigma = 4$ and for various fractions of outliers, the training and validation MSE, the percentage of correct support recovered and the mean implementation time (MIT), is listed.

$\mathcal{T} = \text{supp}(\underline{\boldsymbol{u}})$, RAM (wrongly) identifies more indices as outliers than KGARD.

- For the second experiment, the performance for each method is evaluated for the following set-up. The input data consists of 400 equidistant points over the interval $[0, 1)$ and the uncorrupted observations are generated via $\underline{f}(x_i) = \sum_{j=1}^{400} \underline{\alpha}_j \kappa(x_j, x_i)$ with the Gaussian (RBF) kernel with parameter $\sigma = 0.1$, employing a sparse coefficient vector $\underline{\boldsymbol{\alpha}}$, with the number of non-zero values ranging between $4\% - 18\%$ and their values randomly drawn from the Gaussian distribution $\mathcal{N}(0, 20^2)$. In the sequel, the set of points is split into two subsets, the training and the validation subset. Similar to the first experiment, the training subset consists of the $N = 200$ odd indexed points (first, third, e.t.c.), while the remaining (even indices) correspond to the validation/test subset. The uncorrupted observations of the training set are generated via (4) and contaminated by random Gaussian with variance 4. Various fractions of outliers have been used (distributed uniformly over the training points) with values ±40.

  In Table 3, the performance for each method is shown. Once again, KGARD attains the lowest MSE for all fractions of outliers up to 15%. It is readily seen that, this holds despite the fact that the support of the sparse outlier vector is not fully recovered (due to the existence of heavy inlier noise). Also, for the case where 20% of outlier values are present, the MSE for RAM is lower than KGARD's, for both the training and the validation set.

- For the final pilot experiment, KGARD's performance is tested for the case where the input data lies on a two-dimensional subspace. To this end, we consider 31 points in $[0, 1]$ and separate these points, to form the training set, which comprises 16 odd indices and the rest 15, forming the validation set. Next, the 31^2 points are distributed over a squared lattice in plane $[0, 1] \times [0, 1]$, where each uncorrupted measurement is generated by $\underline{f}(\boldsymbol{x}_i) = \sum_{j=1}^{31^2} \underline{\alpha}_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$, ($\sigma = 0.2$) and a sparse coefficient vector $\underline{\boldsymbol{\alpha}} = [\underline{\alpha}_1, \ldots, \underline{\alpha}_{31}]$ with non-zero values ranging between $4\% - 17.5\%$ and their values randomly drawn from $\mathcal{N}(0, 25.6^2)$. Thus, the training subset, consists of $N = 16^2$ points, while the remaining 15^2 correspond to the validation/test subset. According to equation (4), the original observations of the training set are corrupted by inlier noise originating from $\mathcal{N}(0, 3^2)$ and outlier values ±40. The results are given in Table 4 for various fractions of outliers, with the best values of the MSE marked in **bold**. It is evident that, for the 2-dimensional non-linear denoising task, KGARD's performance outperforms its competitors (in terms of MSE), for all fractions of the outliers.

Finally, it should also be noted that, although RB-RVM does not perform at the highest level, has the advantage that needs no tuning of parameters, albeit at substantially increased

Method	MSE_{tr}	MSE_{val}	Cor. supp	Wr. supp	MIT (sec)	Outliers
RB-RVM	3.9825	3.6918	-	-	0.416	5%
RAM ($\lambda = 0.2, \mu = 22$)	2.0534	1.8592	100%	0.1 %	0.010	5%
KGARD ($\lambda = 0.15, \varepsilon = 46$)	**1.7381**	**1.5644**	100 %	0.3 %	0.009	5%
RB-RVM	4.2382	3.8977	-	-	0.419	10%
RAM ($\lambda = 0.2, \mu = 18$)	2.2281	1.9926	100%	0.9 %	0.013	10%
KGARD ($\lambda = 0.15, \varepsilon = 44$)	**1.8854**	**1.6750**	100 %	0.5 %	0.016	10%
RB-RVM	4.5749	4.2181	-	-	0.418	15%
RAM ($\lambda = 0.2, \mu = 17$)	2.5944	2.2846	100%	1.6 %	0.016	15%
KGARD ($\lambda = 0.2, \varepsilon = 42$)	**2.1968**	**1.9375**	99.9 %	0.9 %	0.024	15%
RB-RVM	5.7051	5.0540	-	-	0.418	20%
RAM ($\lambda = 0.2, \mu = 16$)	3.0593	2.6703	99.9%	2.3 %	0.020	20%
KGARD ($\lambda = 0.4, \varepsilon = 42$)	**3.0293**	**2.6113**	99.9 %	1 %	0.033	20%

Table 4: Performance evaluation for each method, for the case where the input data lies on the 2-dimensional space and the output $\underline{f} \in \mathcal{H}$ is considered as a linear combination of a few kernels. The inlier noise is considered random Gaussian with $\sigma = 3$ and for various fractions of outliers, the training and validation MSE, the percentage of correct support recovered and the mean implementation time (MIT), is listed.

computational cost. On the contrary, the pair of tuning parameters for RAM, renders the method very difficult to be fully optimized (in terms of MSE), in practise. In contrast, taking into account the physical interpretation of ϵ and λ associated with KGARD, in the noise denoising task, we have developed a method for automatic user-free choice of these variables.

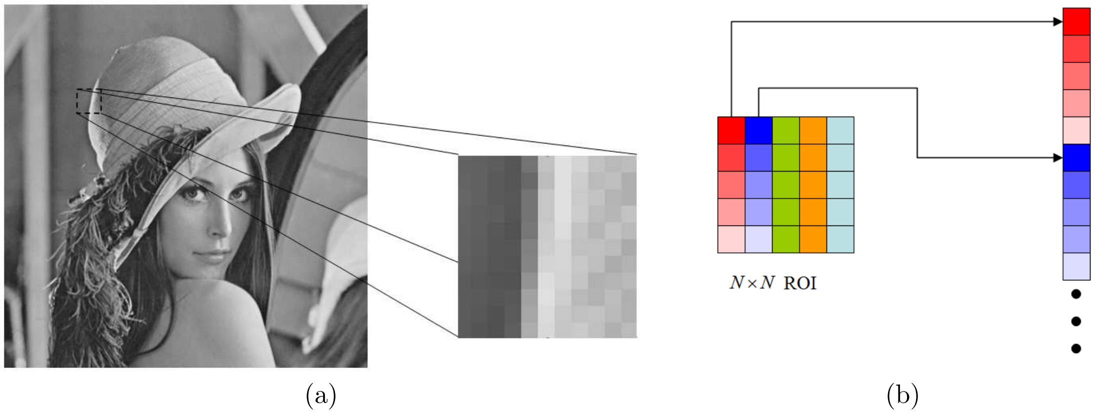
7 Application in Image Denoising

In this section, in order to test the capabilities and verify the performance of the proposed algorithmic scheme, we use the KGARD framework to address one of the most popular problems that rise in the field of image processing: the task of removing noise from a digital image. The source of noise in this case can be either errors of the imaging system itself (e.g., hardware or software errors, transmission errors, quantization errors), errors that occur due to limitations of the imaging system (e.g., small size of the sensor), or errors that are generated by the environment (e.g., low light, heat, e.t.c.). Typically, the noisy image is modeled as follows:

$$g(x, x') = \underline{g}(x, x') + v(x, x'),$$

for $x, x' \in [0, 1]$, where $\underline{g}$ is the original noise-free image and v the additive noise. Given the noisy image g, the objective of any image denoising method is to obtain an estimate of the original image $\underline{g}$. In most cases, we assume that the image noise is Gaussian additive, independent at each pixel, and independent of the signal intensity, or that it contains spikes or impulses (i.e., salt and pepper noise). However, there are cases where the noise model follows other probability density functions (e.g., the Poisson distribution or the uniform distribution). Typical methods that have been proposed to address the image denoising task include (a) the wavelet-based image denoising methods, which dominate the research in recent years [45, 46, 47], (b) methods based on Partial Differential Equations [48], (c) neighborhood filters, and (d) methods of non linear modeling using local expansion approximation techniques [49]. The majority of these methods assume a specific type of noise model. In fact, most of them require some sort of a priori knowledge of the noise distribution. In contrast to this approach, the more recently introduced denoising methods based on kernel ridge regression (KRR) make no assumptions about the underlying noise model, and thus, they can effectively treat more complex models [18].

In this section, we demonstrate how the proposed KGARD algorithmic scheme can be used to treat the image denoising problem in cases where the noise model includes impulses. To this end, we adopt a more complex additive noise model that can be decomposed into two parts: a bounded noise component and a sparse noise model that comprises impulses. We will present two different



(a) (b)

Figure 3: (a) A square $N \times N$ region of intest (ROI). (b) Rearranging the pixels of a ROI.

denoising methods to deal with this type of noise. The first one is directly based on KGARD algorithmic scheme, while the second method splits the denoising procedure into two parts: the identification and removal of the impulses is first carried out, via the KGARD and then the output is fed into a cutting edge wavelet based denoising method to remove the bounded component. In the following, we describe both methods in more detail.

7.1 Splitting the Image into ROIs

In the proposed denoising method, we adopt the well known and popular strategy of dividing the “noisy” image into smaller $N \times N$ square regions of interest (ROIs), as it is illustrated in Figure 3. Then, we rearrange the pixels so that to form a row vector. Instead of applying the denoising process to the entire image, we process each ROI individually in sequential order. This is done for two reasons: (a) Firstly, the time needed to solve the optimization tasks considered in the next sections increases polynomially with N^2 and (b) working with each ROI separately enables us to change the parameters of the model in an adaptive manner, to account for the different level of details in each ROI. The rearrangement shown in Figure 3 implies that, the pixel (i, j) (i.e., i-th row, j-th column) is placed at the n-th position of the respective vector, where $n = (i-1) \cdot N + j$.

7.2 Modeling the Image and the Noise

In kernel ridge regression denoising methods, one assumes that each ROI represents the points on the surface of a continuous function, $\underline{g}$, of two variables defined on $[0,1] \times [0,1]$. The pixel values of the noise-free and the noisy digitized ROIs are represented as $\underline{\zeta}_{ij} = \underline{g}(x_i, x'_j)$ and ζ_{ij} respectively (both taking values in the interval $[0, 255]$), where $x_i = (i-1)/(N-1)$, $x'_j = (j-1)/(N-1)$, for $i, j = 1, 2, ..., N$. Moreover, as the original image $\underline{g}$ is a relatively smooth function (with the exception close to the edges), we assume that it lies in an RKHS induced by the Gaussian kernel, i.e., $\underline{g} \in \mathcal{H}$, for some $\sigma > 0$. Specifically, in order to be consistent with the representer theorem, we will assume that $\underline{g}$ takes the form of a finite linear representation of kernel functions, i.e.,

$$\underline{g} = \sum_{i,j=1}^{N} \underline{\alpha}_{ij} \kappa(\cdot, (x_i, x'_j)). \tag{28}$$

After pixel rearrangement, equation (28) can be cast as:

$$\underline{g} = \sum_{n=1}^{N^2} \underline{\alpha}_n \kappa(\cdot, \boldsymbol{x}_n),$$

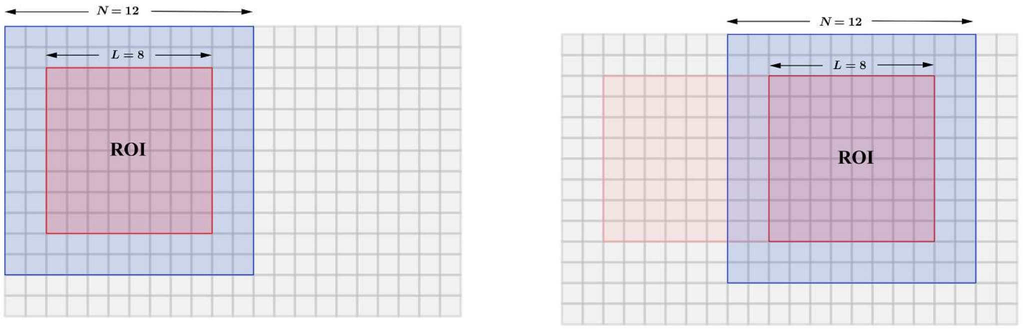


Figure 4: Two consecutive $N \times N$ ROIs. Observe that the two ROIs overlap.

where $n = (i-1) \cdot N + j$ and $\boldsymbol{x}_n = (x_i, x'_j)$. Hence, the intensity of the n-th pixel is given by

$$\underline{\zeta}_n = \underline{g}(\boldsymbol{x}_n) = \sum_{m=1}^{N^2} \underline{\alpha}_m \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m). \tag{29}$$

The model considered in this paper assumes that the intensity of the pixels of the noisy ROI can be decomposed as follows:

$$\zeta_{ij} = \underline{\zeta}_{ij} + \underline{u}_{ij} + \eta_{ij},$$

for $i, j = 1, 2, ..., N$, where η_{ij} denotes the bounded noise component and $\underline{u}_{ij}$ the possible appearance of an outlier at that pixel. In vector notation (after rearrangement), we can write

$$\boldsymbol{\zeta} = \underline{\boldsymbol{\zeta}} + \underline{\boldsymbol{u}} + \boldsymbol{\eta}, \tag{30}$$

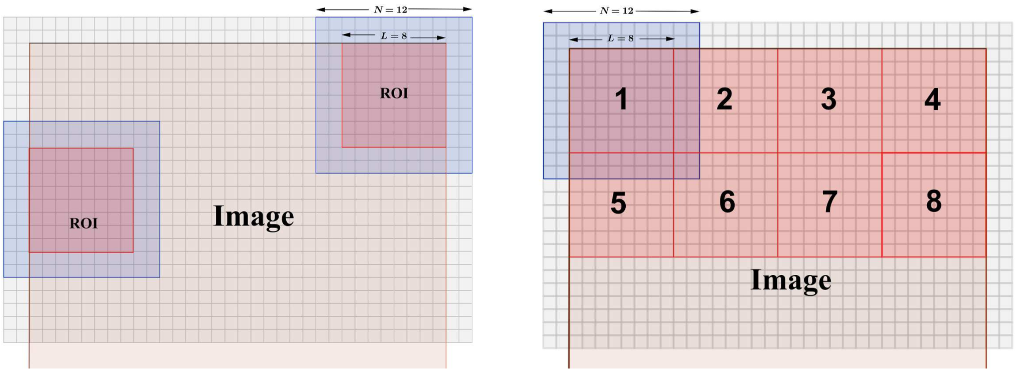
where $\boldsymbol{\zeta}, \underline{\boldsymbol{\zeta}}, \underline{\boldsymbol{u}}, \boldsymbol{\eta}, \in \mathbb{R}^{N^2}$, $\|\boldsymbol{\eta}\|_2 \leq \epsilon$ and $\underline{\boldsymbol{u}}$ is a sparse vector. Moreover, as the elements of $\underline{\boldsymbol{\zeta}}$ take the form (29), we can write $\underline{\boldsymbol{\zeta}} = \boldsymbol{K} \cdot \underline{\boldsymbol{\alpha}}$, where $\kappa_{nm} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$. In this context, we can model the denoising task as the following optimization problem:

$$\begin{array}{cc} \text{minimize}_{\boldsymbol{a}, \boldsymbol{u} \in \mathbb{R}^{N^2}, c \in \mathbb{R}} & \|\boldsymbol{u}\|_0 \\ \text{subject to} & \|\boldsymbol{\zeta} - \boldsymbol{K}\boldsymbol{a} - c\mathbf{1} - \boldsymbol{u}\|_2^2 + \lambda \|\boldsymbol{a}\|_2^2 + \lambda c^2 \leq \varepsilon, \end{array} \tag{31}$$

for some predefined $\lambda, \varepsilon > 0$. In a nutshell, problem (31) solves for the sparsest outlier's vector $\boldsymbol{u}$ and the respective $\boldsymbol{a}$ (i.e., the coefficients of the kernel expansion) that keep the error low, while at the same time preserve the smoothness of the original noise-free ROI (this is done via the regularization of the constraint's inequality). The regularization parameter λ controls the smoothness of the solution. The larger the λ is, the smoother the solution becomes, i.e., $\hat{\underline{\boldsymbol{\zeta}}} = \boldsymbol{K}\hat{\boldsymbol{\alpha}}$.

7.3 Implementation

The main mechanism of both algorithms that are presented in this section is simple. The image is divided into $N \times N$ ROIs and the KGARD algorithm is applied in each individual ROI sequentially. However, as the reconstruction accuracy of KRR methods drops near the borders of the respective domain, we have chosen to discard the values at those points. This means that although KGARD is applied to the $N \times N$ ROI, only the $L \times L$ values are used in the final reconstruction (those that are at the center of the ROI). In the sequel, we will name the $L \times L$ centered region as "reduced ROI" or rROI for short. Alternatively, one may consider that the image is actually divided into $L \times L$ non-overlapping regions (the rROIs) and these regions are extended to the size $N \times N$. This



(a) (b)

Figure 5: (a) The algorithm has reached the right end of the image, hence it moves L pixels below. (b) In this example, $L = 8$, $N = 12$. The image has been padded using 2 pixels in all dimensions. The Figure shows the 8 first rROIs.

means that the ROIs contain overlapping parts. We will also assume that the dimensions of the image are multipliers of L (if they are not, we can add dummy pixels to the end) and select N so that $N - L$ is an even number.

After the reconstruction of a specific rROI, the algorithm moves to the next one, i.e., it moves L pixels to the right (see Figure 4), or, if the algorithm has reached the right end of the image, it moves at the beginning of the line, which is placed L pixels below (see Figure 5(a)). Observe that, for this procedure to be valid, the image has to be padded by adding $(N - L)/2$ pixels along all dimensions. In this paper, we chose to pad the image by repeating border elements[8]. For example, if we select $L = 8$ and $N = 12$ to apply this procedure on an image with dimensions[9] 32×32, we will end up with a total of 16 overlapping ROIs, 4 per line (see Figure 5(b)).

Another important aspect of the denoising algorithm is the automated selection of the parameters λ and ϵ, that are involved in KGARD. This is an important feature, as these parameters largely control both the quality of the estimation and the recovery of the outliers and have to be tuned for each specific ROI. Naturally, it would have been intractable to require a user pre-defined pair of values (i.e., λ, ϵ) for each specific ROI. Hence, we devised simple methods to adjust these values in each ROI depending on its features.

7.3.1 Automatic Selection of the Regularization Parameter λ

This parameter controls the smoothing operation of the denoising process. The user enters a specific value for λ_0 to control the strength of the smoothening and then the algorithm adjusts this value at each ROI separately, so that λ is small at ROIs that contain a lot of "edges" and large at ROIs that contain smooth areas. Whether a ROI has edges or not is determined by the mean magnitude of the gradient at each pixel. The rationale is described in algorithm 5.

7.3.2 Automatic Computation of the Termination Parameter ϵ

The stopping criterion for KGARD, that has been adopted for the image denoising task, is slightly different than the one employed in Algorithm 4. In this case, instead of requiring the norm of the residual vector to drop below ϵ, i.e., $\|\boldsymbol{r}_{(k)}\|_2 \leq \epsilon$, we require the maximum absolute valued coordinate of $\boldsymbol{r}_{(k)}$ to drop below ϵ $\left(\left\|\boldsymbol{r}_{(k)}\right\|_\infty \leq \epsilon\right)$. The estimation of ϵ for each particular ROI is carried out as follows. Initially, a user defined parameter E_0 is selected. At each step, a histogram chart with elements $|r_{(k),i}|$ is generated, using $\left[\frac{N^2}{10}\right] + 1$ equally spaced bins along the x-axis,

[8] This can be done with the 'replicate' option of MatLab's function padarray.

[9] Observe that L divides 32.

Algorithm 5 Selection of the regularization parameter λ

```
1: Select a user-defined value $\lambda_0$.
2: Compute the magnitude of the gradient at each pixel.
3: Compute the mean gradient of each ROI, i.e., the mean value of the gradient's magnitude of all pixels that belong to the ROI.
4: Compute the mean value, $m$, and the standard deviation, $s$, of the aforementioned mean gradients.
5: ROIs with mean gradient larger than $m + s$ are assumed to be areas with fine details and the algorithm sets $\lambda = \lambda_0$.
6: All ROIs with mean gradient lower than $m - s/10$ are assumed to be smooth areas and the algorithm sets $\lambda = 15\lambda_0$.
7: For all other ROIs the algorithm sets $\lambda = 5\lambda_0$.
```

between the minimum and maximum values of $|r_{(k),i}|$. Let $\boldsymbol{h}$ denote the heights of the bars of the histogram and h_m be the minimum height of the histogram bars. Next, two real numbers, i.e., E_1, E_2, are defined. In particular, the number E_1 represents the left endpoint of the first occurrence of a minimum-height bar (i.e., the first bar with height equal to h_m, moving from left to right). The number E_2 represents the left endpoint of the first bar, ℓ, with height h_ℓ (moving from left to right) that satisfies both $h_\ell - h_{\ell-1} \geq 1$ and $h_{\ell-1} \leq h_m + 5$, $\ell \geq 2$. This roughly corresponds to the first increasing bar, which in parallel is next to a bar with height close to the minimum height. Figure 6 demonstrates some typical examples regarding the computation of these numbers. Both E_1 and E_2 are reasonable choices for the value of ϵ (meaning that the bars to the right of these values may be assumed to represent outliers). Finally, the algorithm determines whether the histogram can be clearly divided into two parts; the first one represents the usual errors and the other the errors due to outliers by using a simple rule: if $\frac{\sqrt{\mathrm{var}(\boldsymbol{h}_{(k)})}}{\mathrm{mean}(\boldsymbol{h}_{(k)})} > 0.9$, then we the two areas can be clearly distinguished (e.g., Figure 6(a)-(c)), otherwise it is harder to separate these areas (e.g., Figure 6(d)). Note that, we use the notation $\boldsymbol{h}_{(k)}$ to refer to the heights of the histogram bar at the k step of the algorithm. The final computation of ϵ (at step k) is carried out as follows:

$$\epsilon_{(k)} = \begin{cases} \min\{E_0, E_1, E_2\}, & \text{if } \frac{\sqrt{\mathrm{var}(\boldsymbol{h}_{(k)})}}{\mathrm{mean}(\boldsymbol{h}_{(k)})} > 0.9 \\ \min\{E_0, E_1\}, & \text{otherwise.} \end{cases} \tag{32}$$

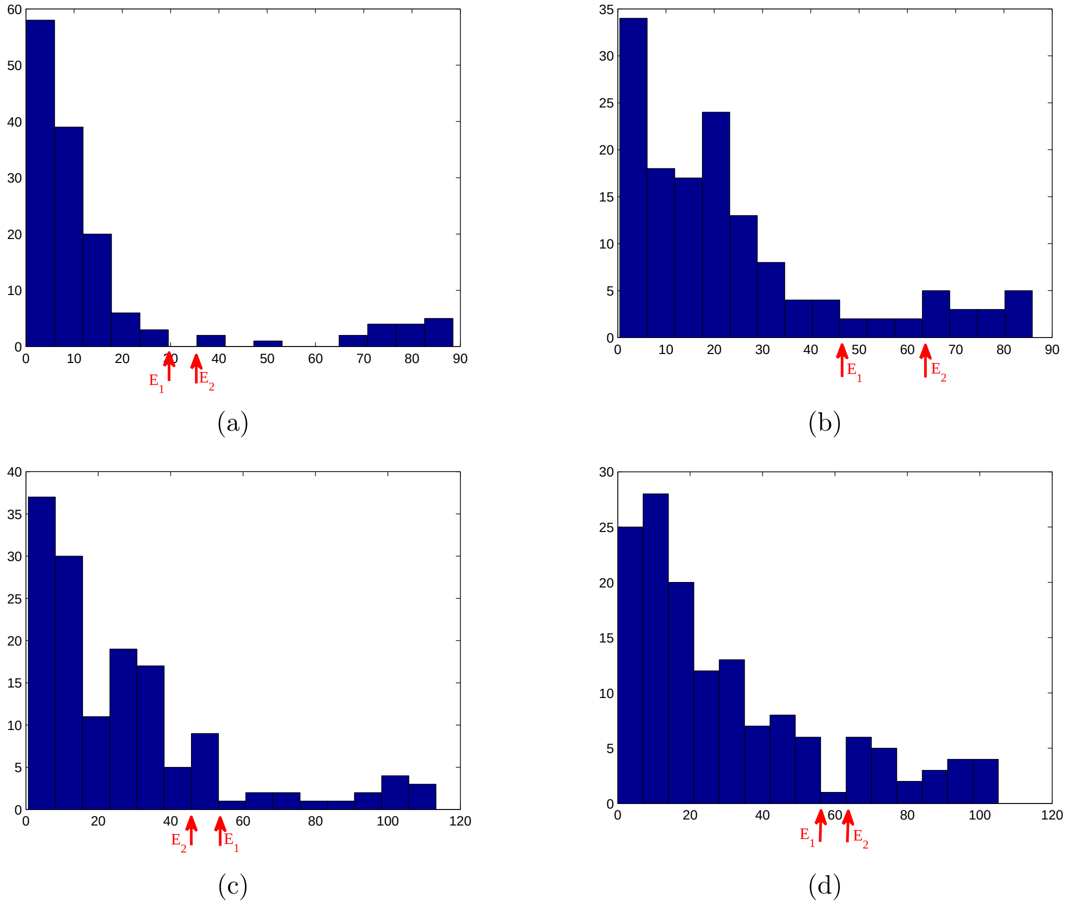
It should be noted that, the user defined parameter E_0 has little importance in the evaluation of ϵ. One may set it constantly to a value near 40 (as we did in all provided simulations). However, in cases where the image is corrupted by outliers only, a smaller value may be advisable, although it does not have a great impact on the reconstruction quality.

7.3.3 Direct KGARD Implementation

The first denoising method, which we call "Kernel GARD Denoising" (or KGARD for short), is described in Algorithm 6. The algorithm requires five user-defined parameters: (a) the regularization parameter, λ_0, (b) the Gaussian kernel width, σ, (c) the OMP termination parameter ϵ, (d) the size of the ROI, N and (e) the size of the rROIs, that are used in the reconstruction, i.e., L. However, these parameters are somehow interrelated. We will discuss these issues in the next sections.

7.3.4 KGARD Combined with BM3D (KGARD-BM3D)

The second denoising method is actually a two-step procedure, that combines the outliers detection properties of KGARD with the denoising capabilities of a standard off-the-shelf denoising method. The KGARD algorithm is applied onto the noisy image, but this time the obtained denoised image $\hat{\boldsymbol{I}}$ is discarded and only the positions and values of the reconstructed outliers are taken into



(a) (b) (c) (d)

Figure 6: Histograms of the residual vectors used in the automatic computation of ϵ.

consideration. These are subtracted from the original noisy image and a cutting edge wavelet-based method with the name BM3D is applied to the result [47]. In this setting (which is the one we propose) the KGARD is actually used to detect the outliers and remove them, while the BM3D methods takes over afterwards to clean the bounded noise. Figure 7 illustrates this procedure. This method requires the same parameters as KGARD, plus the parameter s, which is needed by the BM3D algorithm[10].

7.4 Parameter Selection

This section is devoted on providing guidelines for the selection of the user defined parameters for the proposed denoising algorithms. Typical values of N range between 8 and 16. Values of N near 8, or even lower, increase the time required to complete the denoising process with no significant improvements in most cases. However, if the image contains a lot of "fine details" this may be advisable. In these cases, smaller values for the width of the Gaussian kernel, σ, may also enhance the performance, since in this case the regression task is more robust to abrupt changes. However, we should note that σ is inversely associated with the size[11] of the ROI, N, hence if one increases N, one should decrease σ proportionally, i.e., keeping the product $N \cdot \sigma$ constant. We

[10] BM3D is built upon the assumption that the image is corrupted by Gaussian noise. Hence, the parameter s is the variance of that Gaussian noise, if this is known a-priori, or some user-defined estimate. However, it has been demonstrated that BM3D can also efficiently remove other types of noise, if s is adjusted properly [18].

[11] For example, if $N = 12$ and $\sigma = 0.3$, then the kernel width is equal to 3.6 pixels. It is straightforward to see that, if N decreases to say 8, then the kernel width that will provide a length of 3.6 pixels is $\sigma = 0.45$.

Algorithm 6 KGARD for image denoising

```
Input: the original noisy image $\boldsymbol{I}$ and the parameters $\lambda_0$, $\sigma$, $E_0$, $N$, $L$.
Output: the denoised image $\hat{\boldsymbol{I}}$ and the outliers' image $\hat{\boldsymbol{O}}$.
Build the kernel matrix $\boldsymbol{K}$.
if the dimensions of the original image are not multiplies of $L$ then
    Add initial padding
Form $\hat{\boldsymbol{I}}$ and $\hat{\boldsymbol{O}}$ so that they have the same dimensions as $\boldsymbol{I}$.
Add padding with size $N - L$ around the image.
Divide the image into $N \times N$ ROIs and compute the regularization parameters of each ROI
  according to algorithm 5.
for each ROI $\boldsymbol{R}$ do
    Rearrange the pixels of $\boldsymbol{R}$ to form the vector $\boldsymbol{\zeta}$.
    Run the modified KGARD algorithm on the set $\boldsymbol{\zeta}$ with parameter $\lambda$ and stoping criterion
  as described in section 7.3.2.
    Let $\hat{\boldsymbol{a}}$, $\hat{\boldsymbol{u}}$ be the solution according to KGARD algorithm.
    Compute the denoised vector $\hat{\boldsymbol{\zeta}} = \boldsymbol{K}\hat{\boldsymbol{a}}$.
    Rearrange the elements of $\hat{\boldsymbol{\zeta}}$ to form the denoised ROI $\hat{\boldsymbol{R}}$.
    Extract the centered $L \times L$ rROI from $\hat{\boldsymbol{R}}$.
    Use the values of the rROI to set the values of the corresponding pixels in $\hat{\boldsymbol{I}}$.
    Rearrange the elements of $\hat{\boldsymbol{u}}$ to form the outliers' ROI.
    Extract the centered $L \times L$ values of the outliers' ROI.
    Use these values to set the values of the corresponding outliers in $\hat{\boldsymbol{O}}$.
    Move to the next ROI.
Remove the initial padding on $\hat{\boldsymbol{I}}$ and $\hat{\boldsymbol{O}}$ (if needed).
```

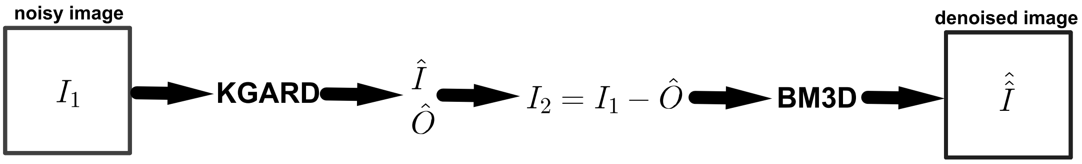


Figure 7: The KGARD-BM3D denoising method. First the KGARD is applied on the noisy image to extract the outliers and then BM3D removes the bounded noise.

have observed that the values $N = 12$ and $\sigma = 0.3$ (which result to a product equal to $N \cdot \sigma = 3.6$) are adequate to remove moderate noise from a typical image. In cases where the image has many details and edges, N and σ should be adjusted to provide a lower product (e.g., $N = 12$ and $\sigma = 0.15$, so that $N \cdot \sigma = 1.8$). For images corrupted by high noise, this product should become larger. Finally, λ controls the importance of regularization on the final result. Large values imply a strong smoothing operation, while small values (close to zero) reduce the effect of regularization leading to a better fit, however, it may lead to overfitting.

For the experiments presented in this paper, we fixed the size of the ROIs using $N = 12$ and $L = 8$. These are reasonable choices that provide fast[12] results with high reconstruction accuracy. Hence, only the values for σ and λ_0 need to be adjusted according to the density of the details in the image and the amount of noise. We have found that the values of σ that provide adequate results range between 0.1 and 0.4. Similarly, typical values of λ_0 range from 0.1 to 1. Finally, the constant E_0 was set equal to 40 for all cases.

The parameter s of the BM3D method is adjusted according to the amount of noise presented in the image. It ranges between very small values (e.g, 5), when only a small amount of bounded

[12] A typical denoising task using either KGARD or KGARD-BM3D implemented in MATLAB takes less than a minute on a moderate computer.

noise is present, to significantly larger values (e.g., 20 or 40) if the image is highly corrupted.

7.5 Experiments on Images Corrupted by Synthetic Noise

In this section, we present an extensive set of experiments on grayscale images that have been corrupted by mixed noise, which comprises a Gaussian component and a set of impulses (± 100). The intensity of the gaussian noise has been ranged between 15 dB and 25 dB and the percentage of impulses from 5% to 20%. The tests were performed on three very popular images: the *Lena*, the *boat* and the *Barbara* images, that are included in Waterloo's image repository. Each test has been performed 50 times and the respective mean PSNRs are reported. The parameters have been tuned so that to provide the best result (in terms of MSE). In Table 5, the two proposed methods are applied to the *Lena* image and they are compared with BM3D (the state of the art wavelet-based method) and an image denoising method based on (RB-RVM). For the latter, we chose a simple implementation, similar to the one we propose in our methods: the image is divided into ROIs and the RB-RVM algorithm is applied to each ROI sequentially. The parameters were selected to provide the best possible results in terms of PSNR. Tables 6 and 7 apply BM3D and KGARD-BM3D on the *boat* and *Barbara* images. The size of the ROIs has been set to $N = 12$ and $L = 8$ for the Lena and *boat* image. As the *Barbara* image has more finer details (e.g., the stripes of the pants) we have set $N = 12$ and $L = 4$ for this image. Moreover, one can observe that for this image, we have used a lower value for σ and λ as indicated in section 7.4. Figures 8, 9 and 10 show the obtained denoised images on a specific experiment (20 dB Gaussian noise and 10% outliers). The experiments show that the proposed method (KGARD-BM3D) enhances significantly the denoising capabilities of BM3D, especially for low and moderate intensities of the Gaussian noise. If the Gaussian component becomes prominent (e.g., at 15 dB) then the two methods provide similar results.

Finally, it is noted that we chose not to include RAM or any ℓ_1-based denoising method, as this would require efficient techniques to adaptively control its parameters, i.e., λ, μ at each ROI (similar to the case of KGARD), which remains an open issue. Having to play with both parameters, makes the tuning computationally demanding. This is because the number of iterations for the method to converge to a reasonable solution increases substantially, once the parameters are moved away from their optimal (in terms of MSE) values[13].

[13] If the parameters are not optimally tuned, the denoising process may take more than an hour to complete in MATLAB on a moderate computer.

Method	Parameters	Gaussian Noise	Impulses (±100)	PSNR
BM3D	$s = 30$	25 dB	5%	32.2 dB
RB-RVM	$\sigma = 0.3$	25 dB	5%	31.78 dB
KGARD	$\sigma = 0.3$, $\lambda = 1$	25 dB	5%	33.91 dB
KGARD-BM3D	$\sigma = 0.3$, $\lambda = 1$, $s = 10$	25 dB	5%	**36.2 dB**
BM3D	$s = 30$	25 dB	10%	30.84 dB
RB-RVM	$\sigma = 0.3$	25 dB	10%	31.25 dB
KGARD	$\sigma = 0.3$, $\lambda = 1$, $\epsilon = 40$	25 dB	10%	33.49 dB
KGARD-BM3D	$\sigma = 0.3$, $\lambda = 1$, $s = 10$	25 dB	10%	**35.67 dB**
BM3D	$s = 45$	25 dB	20%	29.28 dB
RB-RVM	$\sigma = 0.4$	25 dB	20%	30.3 dB
KGARD	$\sigma = 0.4$, $\lambda = 1$	25 dB	20%	32.04 dB
KGARD-BM3D	$\sigma = 0.4$, $\lambda = 1$, $s = 15$	25 dB	20%	**33.69 dB**
BM3D	$s = 30$	20 dB	5%	31.83 dB
RB-RVM	$\sigma = 0.4$	20 dB	5%	29.3 dB
KGARD	$\sigma = 0.3$, $\lambda = 1$	20 dB	5%	32.35 dB
KGARD-BM3D	$\sigma = 0.3$, $\lambda = 1$, $s = 15$	20 dB	5%	**34.24 dB**
BM3D	$s = 35$	20 dB	10%	30.66 dB
RB-RVM	$\sigma = 0.4$	20 dB	10%	29.09 dB
KGARD	$\sigma = 0.3$, $\lambda = 1$	20 dB	10%	31.94 dB
KGARD-BM3D	$\sigma = 0.3$, $\lambda = 1$, $s = 15$	20 dB	10%	**33.81 dB**
BM3D	$s = 50$	20 dB	20%	29.86 dB
RB-RVM	$\sigma = 0.4$	20 dB	20%	28.29 dB
KGARD	$\sigma = 0.4$, $\lambda = 1$	20 dB	20%	30.72 dB
KGARD-BM3D	$\sigma = 0.4$, $\lambda = 1$, $s = 15$	20 dB	20%	**32.06 dB**
BM3D	$s = 35$	15 dB	5%	30.87 dB
RB-RVM	$\sigma = 0.6$	15 dB	5%	26.74 dB
KGARD	$\sigma = 0.3$, $\lambda = 1.5$	15 dB	5%	29.12 dB
KGARD-BM3D	$\sigma = 0.3$, $\lambda = 1$, $s = 25$	15 dB	5%	**31.18 dB**
BM3D	$s = 40$	15 dB	10%	29.94 dB
RB-RVM	$\sigma = 0.4$	15 dB	10%	25.85 dB
KGARD	$\sigma = 0.3$, $\lambda = 2$	15 dB	10%	28.47 dB
KGARD-BM3D	$\sigma = 0.3$, $\lambda = 1$ $s = 25$	15 dB	10%	**30.77 dB**
BM3D	$s = 40$	15 dB	20%	28.78 dB
RB-RVM	$\sigma = 0.4$	15 dB	20%	25 dB
KGARD	$\sigma = 0.4$, $\lambda = 3$	15 dB	20%	27.87 dB
KGARD-BM3D	$\sigma = 0.4$, $\lambda = 1$, $s = 35$	15 dB	20%	**29.66 dB**

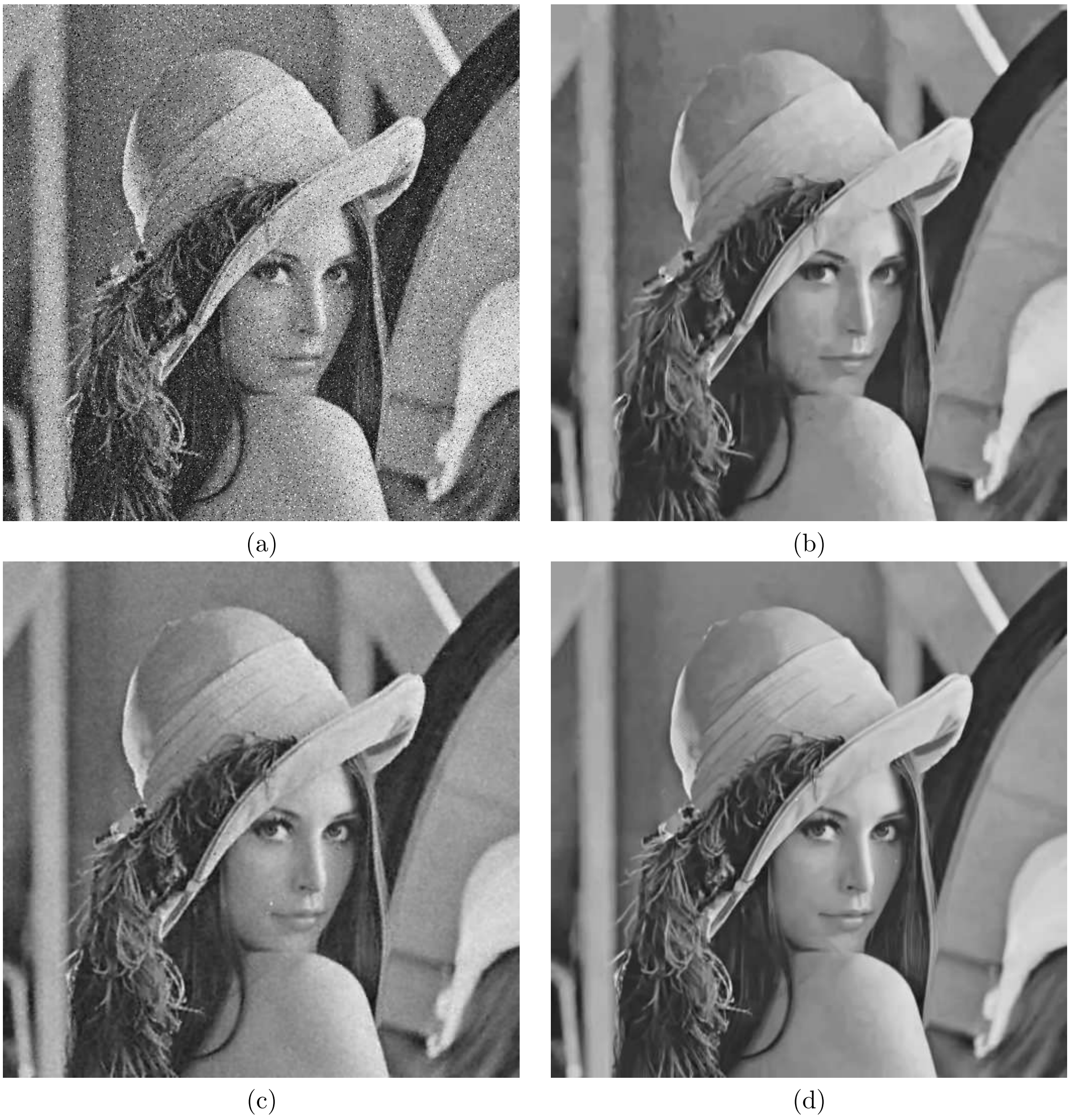
Table 5: Denoising performed on the *Lena* image corrupted by various types and intensities of noise using the proposed methods, the robust RVM approach and the state of the art wavelet method BM3D.

Method	Parameters	Gaussian Noise	Impulses (±100)	PSNR
BM3D	$s = 25$	25 dB	5%	30.57 dB
KGARD-BM3D	$\sigma = 0.3$, $\lambda = 1$, $s = 10$	25 dB	5%	**34.61 dB**
BM3D	$s = 30$	25 dB	10%	29.41 dB
KGARD-BM3D	$\sigma = 0.3$, $\lambda = 1$, $s = 10$	25 dB	10%	**33.86 dB**
BM3D	$s = 45$	25 dB	20%	27.64 dB
KGARD-BM3D	$\sigma = 0.4$, $\lambda = 1$, $s = 15$	25 dB	20%	**31.62 dB**
BM3D	$s = 30$	20 dB	5%	30.16 dB
KGARD-BM3D	$\sigma = 0.3$, $\lambda = 1$, $s = 10$	20 dB	5%	**32.19 dB**
BM3D	$s = 35$	20 dB	10%	28.97 dB
KGARD-BM3D	$\sigma = 0.3$, $\lambda = 1$, $s = 15$	20 dB	10%	**31.52 dB**
BM3D	$s = 50$	20 dB	20%	27.49 dB
KGARD-BM3D	$\sigma = 0.4$, $\lambda = 1$ $s = 15$	20 dB	20%	**29.7 dB**
BM3D	$s = 35$	15 dB	5%	**29.1 dB**
KGARD-BM3D	$\sigma = 0.3$, $\lambda = 1$, $s = 25$	15 dB	5%	28.54 dB
BM3D	$s = 40$	15 dB	10%	**28.13 dB**
KGARD-BM3D	$\sigma = 0.3$, $\lambda = 1$, $s = 25$	15 dB	10%	28.11 dB
BM3D	$s = 50$	15 dB	20%	**27.07 dB**
KGARD-BM3D	$\sigma = 0.4$, $\lambda = 1$, $s = 40$	15 dB	20%	26.99 dB

Table 6: Denoising performed on the *boat* image corrupted by various types and intensities of noise the state of the art wavelet method BM3D with and without detection of outliers.

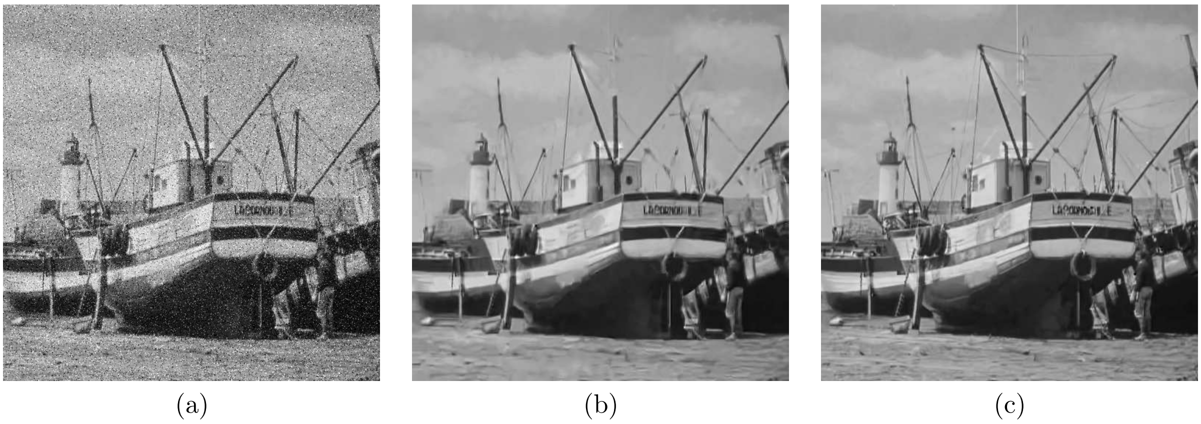
Method	Parameters	Gaussian Noise	Impulses (±100)	PSNR
BM3D	$s = 25$	25 dB	5%	31.06 dB
KGARD-BM3D	$\sigma = 0.15$, $\lambda = 0.1$, $s = 15$	25 dB	5%	**33.45 dB**
BM3D	$s = 30$	25 dB	10%	29.4 dB
KGARD-BM3D	$\sigma = 0.15$, $\lambda = 0.1$, $s = 20$	25 dB	10%	**31.25 dB**
BM3D	$s = 45$	25 dB	20%	27.78 dB
KGARD-BM3D	$\sigma = 0.15$, $\lambda = 0.2$, $s = 30$	25 dB	20%	**28.03 dB**
BM3D	$s = 25$	20 dB	5%	30.69 dB
KGARD-BM3D	$\sigma = 0.15$, $\lambda = 0.1$, $s = 15$	20 dB	5%	**32.24 dB**
BM3D	$s = 35$	20 dB	10%	29.2 dB
KGARD-BM3D	$\sigma = 0.15$, $\lambda = 0.1$, $s = 20$	20 dB	10%	**30.43 dB**
BM3D	$s = 50$	20 dB	20%	**27.68 dB**
KGARD-BM3D	$\sigma = 0.15$, $\lambda = 0.15$, $s = 30$	20 dB	20%	27.48 dB
BM3D	$s = 30$	15 dB	5%	29.71 dB
KGARD-BM3D	$\sigma = 0.15$, $\lambda = 0.1$, $s = 25$	15 dB	5%	**29.97 dB**
BM3D	$s = 40$	15 dB	10%	28.41 dB
KGARD-BM3D	$\sigma = 0.15$, $\lambda = 0.1$, $s = 30$	15 dB	10%	**28.73 dB**
BM3D	$s = 50$	15 dB	20%	**27.27 dB**
KGARD-BM3D	$\sigma = 0.15$, $\lambda = 0.1$, $s = 45$	15 dB	20%	26.39 dB

Table 7: Denoising performed on the *Barbara* image corrupted by various types and intensities of noise using the state of the art wavelet method BM3D with and without detection of outliers.



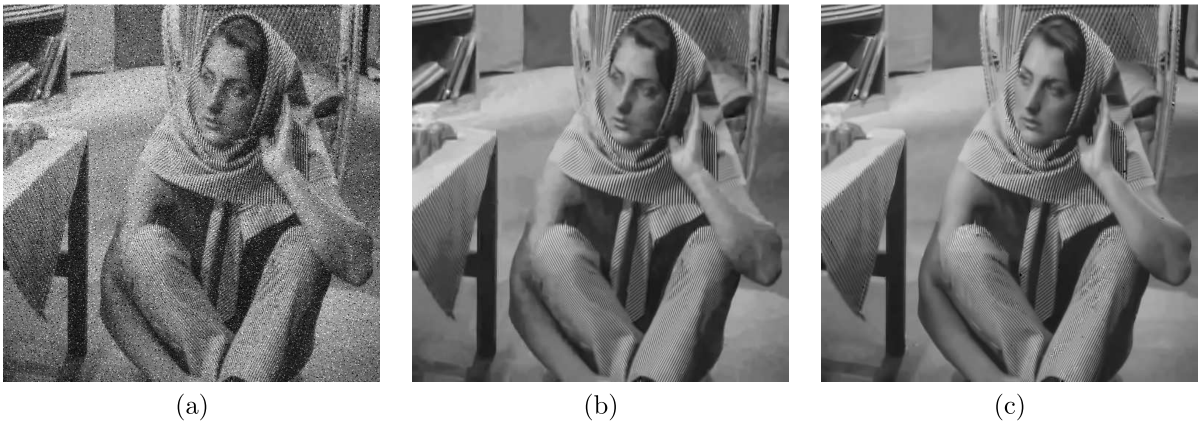
(a) (b) (c) (d)

Figure 8: (a) The *Lena* image corrupted by 20 dB of Gaussian noise and 10% outliers. (b) Denoising with BM3d (30.66 dB). (c) Denoising with KGARD (31.94 dB). (d) Denoising with joint KGARD-BM3D (33.81 dB).



(a) (b) (c)

Figure 9: (a) The *boat* image corrupted by 20 dB of Gaussian noise and 10% outliers. (b) Denoising with BM3d (28.97 dB). (c) Denoising with joint KGARD-BM3D (31.52 dB).



(a) (b) (c)

Figure 10: (a) The *Barbara* image corrupted by 20 dB of Gaussian noise and 10% outliers. (b) Denoising with BM3d (29.2 dB). (c) Denoising with joint KGARD-BM3D (30.43 dB).

Appendix A. Proof of Theorem 3

Proof. Our analysis is based on the *singular value decomposition* (SVD) for matrix $\boldsymbol{X}_{(0)} = [\boldsymbol{K}\ \boldsymbol{1}]$. Since matrix $\boldsymbol{X}_{(0)}^T\boldsymbol{X}_{(0)}$ is positive semi-definite, all of its eigenvalues are non-negative. Let $\boldsymbol{X}_{(0)} = \boldsymbol{QSV}^T$, where $\boldsymbol{Q}, \boldsymbol{V}$ are orthogonal, i.e., $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{QQ}^T = \boldsymbol{I}_N$ and $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{VV}^T = \boldsymbol{I}_{N+1}$ and $\boldsymbol{S}$ is the matrix of dimension $N \times (N+1)$ of the form $\boldsymbol{S} = \begin{bmatrix}\boldsymbol{\Sigma} & \boldsymbol{0}\end{bmatrix}$, where $\boldsymbol{\Sigma}$ is a diagonal matrix, with diagonal entries $\sigma_i \geq 0,\ i = 1, ..., N$. For simplification, the notation σ_M will be used to denote the maximum singular value of matrix $\boldsymbol{X}_{(0)}$.

The proposed method, attempts to solve at each step, the regularized Least Squares (LS) task (19) for the selection of matrix $\boldsymbol{B}$. The latter task is equivalent to a LS problem in the augmented space[14] at each k-step, i.e., (23), where $\boldsymbol{D}_{(k)} = \begin{bmatrix}\boldsymbol{X}_{(k)} \\ \sqrt{\lambda}\boldsymbol{B}_{(k)}\end{bmatrix}$, $\boldsymbol{\theta} = \begin{pmatrix}\boldsymbol{\alpha} \\ c\end{pmatrix}$, $\boldsymbol{X}_{(k)} = \begin{bmatrix}\boldsymbol{X}_{(k-1)} & \boldsymbol{e}_{j_k}\end{bmatrix}$ and $\boldsymbol{B}_{(k)} = \begin{bmatrix}\boldsymbol{B}_{(k-1)} & \boldsymbol{0} \\ \boldsymbol{0}^T & 0\end{bmatrix}$. Thus, the LS solution at each k-step could be expressed as:

$$\hat{\boldsymbol{z}}_{(k)} = (\boldsymbol{D}_{(k)}^T\boldsymbol{D}_{(k)})^{-1}\boldsymbol{D}_{(k)}^T\begin{pmatrix}\boldsymbol{y} \\ \boldsymbol{0}\end{pmatrix} = (\boldsymbol{X}_{(k)}^T\boldsymbol{X}_{(k)} + \lambda\boldsymbol{B}_{(k)})^{-1}\boldsymbol{X}_{(k)}^T\boldsymbol{y} \tag{33}$$

and the respective residual of the lower dimensional space is

$$\boldsymbol{r}_{(k)} = \boldsymbol{y} - \boldsymbol{X}_{(k)}\hat{\boldsymbol{z}}_{(k)} = \boldsymbol{y} - \boldsymbol{X}_{(k)}(\boldsymbol{X}_{(k)}^T\boldsymbol{X}_{(k)} + \lambda\boldsymbol{B}_{(k)})^{-1}\boldsymbol{X}_{(k)}^T\boldsymbol{y}. \tag{34}$$

Step $k = 0$:
Initially, $\boldsymbol{B}_{(0)} = \boldsymbol{I}_{N+1}$ and $\mathcal{S}_0 = \{1, \ldots, N+1\}$ (no index has been selected for the outlier estimate), thus $\boldsymbol{X}_{(0)} = [\boldsymbol{K}\ \boldsymbol{1}]$. Hence, the expression for the initial LS solution $\hat{\boldsymbol{z}}_{(0)}$, is obtained from equation (34) for $k = 0$. Employing the SVD decomposition for matrix $\boldsymbol{X}_{(0)}$, we have

$$\boldsymbol{X}_{(0)}^T\boldsymbol{X}_{(0)} + \lambda\boldsymbol{I}_{N+1} = \boldsymbol{V}\underbrace{\begin{bmatrix}\boldsymbol{\Sigma}^2 + \lambda\boldsymbol{I}_N & \boldsymbol{0} \\ \boldsymbol{0}^T & \lambda\end{bmatrix}}_{\boldsymbol{\Lambda}}\boldsymbol{V}^T = \boldsymbol{V\Lambda V}^T. \tag{35}$$

Combining (34) for $k = 0$ with (35), leads to

$$\boldsymbol{r}_{(0)} = \boldsymbol{y} - \boldsymbol{QGQ}^T\boldsymbol{y}, \tag{36}$$

where $\boldsymbol{G} = \boldsymbol{\Sigma}(\boldsymbol{\Sigma}^2 + \lambda\boldsymbol{I}_N)^{-1}\boldsymbol{\Sigma}$ is a diagonal matrix with entries

$$g_{ii} = \frac{\sigma_i^2}{\sigma_i^2 + \lambda},\ i = 1, 2, ..., n.$$

Furthermore, since $\boldsymbol{y} = \boldsymbol{X}_{(0)}\underline{\boldsymbol{\theta}} + \underline{\boldsymbol{u}}$, substituting in (36) leads to

$$\boldsymbol{r}_{(0)} = \underline{\boldsymbol{u}} + \boldsymbol{QFV}^T\underline{\boldsymbol{\theta}} - \boldsymbol{QGQ}^T\underline{\boldsymbol{u}}, \tag{37}$$

where $\boldsymbol{F} = \boldsymbol{S} - \boldsymbol{GS} = [\underbrace{\boldsymbol{\Sigma} - \boldsymbol{G\Sigma}}_{\boldsymbol{\Phi}}\ \boldsymbol{0}]$. Matrix $\boldsymbol{\Phi}$ is also diagonal, with values

$$\phi_{ii} = \frac{\lambda\sigma_i}{\sigma_i^2 + \lambda},\ i = 1, 2, ..., N.$$

At this point it is required to explore some of the unique properties of matrices $\boldsymbol{G}$ and $\boldsymbol{F}$. Recall that the (matrix) 2-norm of a diagonal matrix is equal to the maximum absolute value of the diagonal entries. Hence, it is clear that

$$||\boldsymbol{G}||_2 = \sigma_M^2/(\sigma_M^2 + \lambda) \text{ and } ||\boldsymbol{F}||_2 = ||\boldsymbol{\Phi}||_2 \leq \sqrt{\lambda}/2, \tag{38}$$

[14] This is due to the fact that $\boldsymbol{B}$ is a projection matrix (based on the ℓ_2 regularization model).

since $g(\sigma) = \frac{\sigma^2}{\sigma^2+\lambda}$ is a strictly increasing function of $\sigma \geq 0$ and $\phi(\sigma) = \frac{\lambda\sigma}{\sigma^2+\lambda}$ receives a unique maximum, which determines the upper bound for the matrix 2-norm.

Finally, it should be noted that if no outliers exist in the noise,the algorithm terminates due to the fact that the norm of the initial residual is less than (or equal to) ϵ. However, this scenario is rather insignificant since no robust modeling is required. Thus, if our goal is for the method to be able to handle various types of noise that includes outliers (e.g. Gaussian noise plus impulses), we assume that $\|\boldsymbol{r}_{(0)}\|_2 > \epsilon$. In such a case, KGARD identifies an outlier selecting an index from the set $\tilde{\mathcal{S}}_0^c = \{1, 2, ..., N\}$.

At the first selection step, as well as at every next step, we should impose a condition, so that the method identifies and selects an index that belongs to the support of the sparse outlier vector. To this end, let $\mathcal{T}$ denote the support of the sparse outlier vector $\underline{\boldsymbol{u}}$. In order for KGARD to select a column $\boldsymbol{e}_i$ from matrix $\boldsymbol{I}_N$ that belongs to $\mathcal{T}$, we should impose

$$|r_{(0),i}| > |r_{(0),j}|, \text{ for all } i \in \mathcal{T} \text{ and } j \in \mathcal{T}^c. \tag{39}$$

The key is to establish appropriate bounds, which guarantee the selection of a correct index that belongs to $\mathcal{T}$. Therefore, we first need to develop bounds on the following inner products. Using (38), the Cauchy-Schwarz inequality and the fact that $\boldsymbol{Q}, \boldsymbol{V}$ are orthonormal, it is easy to verify that

$$\begin{aligned} |\langle \boldsymbol{e}_l, \boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}} \rangle| &= \left|(\boldsymbol{Q}^T\boldsymbol{e}_l)^T\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}}\right| \leq \left\|\boldsymbol{Q}^T\boldsymbol{e}_l\right\|_2 \left\|\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}}\right\|_2 \\ &\leq \|\boldsymbol{F}\|_2 \left\|\boldsymbol{V}^T\underline{\boldsymbol{\theta}}\right\|_2 \leq \frac{\sqrt{\lambda}}{2}\|\underline{\boldsymbol{\theta}}\|_2 \end{aligned} \tag{40}$$

as well as

$$\begin{aligned} |\langle \boldsymbol{e}_l, \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}} \rangle| &= \left|(\boldsymbol{Q}^T\boldsymbol{e}_l)^T\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}}\right| \leq \left\|\boldsymbol{Q}^T\boldsymbol{e}_l\right\|_2 \left\|\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}}\right\|_2 \\ &\leq \|\boldsymbol{G}\|_2 \left\|\boldsymbol{Q}^T\underline{\boldsymbol{u}}\right\|_2 = \frac{\sigma_M^2}{\sigma_M^2+\lambda}\|\underline{\boldsymbol{u}}\|_2, \end{aligned} \tag{41}$$

for all $l = 1, 2, ..., N$. Thus, we have that

$$\begin{aligned} |r_{(0),i}| = |\langle \boldsymbol{r}_{(0)}, \boldsymbol{e}_i \rangle| &= |\langle \underline{\boldsymbol{u}} + \boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}} - \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}}, \boldsymbol{e}_i \rangle| \geq \\ &\geq |\underline{u}_i| - |\langle \boldsymbol{e}_i, \boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}} \rangle| - |\langle \boldsymbol{e}_i, \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}} \rangle| \geq \\ &\geq \min|\underline{u}| - \frac{\sqrt{\lambda}}{2}\|\underline{\boldsymbol{\theta}}\|_2 - \frac{\sigma_M^2}{\sigma_M^2+\lambda}\|\underline{\boldsymbol{u}}\|_2 > \\ &> \min|\underline{u}| - \frac{\sqrt{2\lambda}}{2}\|\underline{\boldsymbol{\theta}}\|_2 - \frac{\sigma_M^2}{\sigma_M^2+\lambda}\|\underline{\boldsymbol{u}}\|_2, \end{aligned} \tag{42}$$

for any $i \in \mathcal{T}$ and

$$\begin{aligned} |r_{(0),j}| = |\langle \boldsymbol{r}_{(0)}, \boldsymbol{e}_j \rangle| &= |\langle \boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}} - \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}}, \boldsymbol{e}_j \rangle| \leq \\ &\leq |\langle \boldsymbol{e}_j, \boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}} \rangle| + |\langle \boldsymbol{e}_j, \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}} \rangle| \leq \\ &\leq \frac{\sqrt{\lambda}}{2}\|\underline{\boldsymbol{\theta}}\|_2 + \frac{\sigma_M^2}{\sigma_M^2+\lambda}\|\underline{\boldsymbol{u}}\|_2 < \\ &< \frac{\sqrt{2\lambda}}{2}\|\underline{\boldsymbol{\theta}}\|_2 + \frac{\sigma_M^2}{\sigma_M^2+\lambda}\|\underline{\boldsymbol{u}}\|_2, \end{aligned} \tag{43}$$

for all $j \in \mathcal{T}^c$, where equation (37) and inequalities (40) and (41), have also been used. Hence, imposing (39), leads to (26). It should be noted, that a reason that could lead to the violation of (26), is for the term $\min|\underline{u}| - \sqrt{2\lambda}\|\underline{\boldsymbol{\theta}}\|_2$ to be non-positive. Thus, since the regularization parameter is fine tuned by the user, we should select $\lambda < \left(\min|\underline{u}| / \|\underline{\boldsymbol{\theta}}\|_2\right)^2/2$. If the condition is guaranteed, then at the first selection step, a column indexed $j_1 \in \mathcal{T}$ is selected. The set of active columns that participates in the LS solution of the current step is then $\mathcal{S}_1 = \{j_1\} \subseteq \mathcal{T}$ and thus $\boldsymbol{X}_{(1)} = \begin{bmatrix}\boldsymbol{X}_0 & \boldsymbol{e}_{j_1}\end{bmatrix}$ and $\boldsymbol{B}_{(1)} = \begin{bmatrix}\boldsymbol{I}_{N+1} & \boldsymbol{0} \\ \boldsymbol{0}^T & 0\end{bmatrix}$.

Step $k = 1$:
After the selection of the first column, follows the inversion of matrix

$$\boldsymbol{D}_{(1)}^T \boldsymbol{D}_{(1)} = \boldsymbol{X}_{(1)}^T \boldsymbol{X}_{(1)} + \lambda \boldsymbol{B}_{(1)} = \begin{bmatrix}\boldsymbol{X}_{(0)}^T \boldsymbol{X}_{(0)} + \lambda \boldsymbol{I}_{N+1} & \boldsymbol{X}_{(0)}^T \boldsymbol{e}_{j_1} \\ \boldsymbol{e}_{j_1}^T \boldsymbol{X}_{(0)} & 1\end{bmatrix}.$$

Applying the *Matrix inversion Lemma*, combined with (35), leads to

$$(\boldsymbol{D}_{(1)}^T \boldsymbol{D}_{(1)})^{-1} = \begin{bmatrix}\boldsymbol{V}\boldsymbol{\Lambda}^{-1}\boldsymbol{V}^T + \frac{1}{\beta}\boldsymbol{V}\boldsymbol{\Gamma}^T\boldsymbol{Q}^T\boldsymbol{e}_{j_1}\boldsymbol{e}_{j_1}^T\boldsymbol{Q}\boldsymbol{\Gamma}\boldsymbol{V}^T & -\frac{1}{\beta}\boldsymbol{V}\boldsymbol{\Gamma}^T\boldsymbol{Q}^T\boldsymbol{e}_{j_1} \\ -\frac{1}{\beta}\boldsymbol{e}_{j_1}^T\boldsymbol{Q}\boldsymbol{\Gamma}\boldsymbol{V}^T & \frac{1}{\beta}\end{bmatrix},$$

where $\boldsymbol{\Gamma} = [\boldsymbol{\Sigma}(\boldsymbol{\Sigma}^2 + \lambda \boldsymbol{I}_N)^{-1} \ \boldsymbol{0}]$ and $\beta = 1 - \boldsymbol{e}_{j_1}^T \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T \boldsymbol{e}_{j_1} = 1 - \left\|\boldsymbol{G}^{1/2}\boldsymbol{Q}^T\boldsymbol{e}_{j_1}\right\|_2^2$. The regularized least squares solution is obtained from (33), for $k = 1$, and after substitution into (34), leads to the new residual vector:

$$\boldsymbol{r}_{(1)} = \boldsymbol{y} - \boldsymbol{X}_{(1)}\hat{\boldsymbol{z}}_{(1)} = \boldsymbol{P}_{(1)}\underline{\boldsymbol{u}} + \boldsymbol{P}_{(1)}\boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}} - \boldsymbol{P}_{(1)}\boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}}, \tag{44}$$

where $\boldsymbol{P}_{(1)} = \boldsymbol{I}_N + \frac{1}{\beta}\boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\boldsymbol{e}_{j_1}\boldsymbol{e}_{j_1}^T - \frac{1}{\beta}\boldsymbol{e}_{j_1}\boldsymbol{e}_{j_1}^T$.

The process of the augmentation of the active set, by the selection of an atom/column, continues, until the norm/length of the residual vector drops below the user-defined threshold. Thus, in order for KGARD to select an index from the set $\mathcal{T}$, we should impose

$$|r_{(1),i}| > |r_{(1),j}|, \text{ for all } i \in \mathcal{T}/\mathcal{S}_1 \text{ and } j \in \mathcal{T}^c.$$

In order to simplify (44), we need to decompose the sparse vector $\underline{\boldsymbol{u}}$ into two parts, i.e., $\underline{\boldsymbol{u}} = \underline{\boldsymbol{u}}_{\mathcal{T}/\mathcal{S}_1} + \underline{\boldsymbol{u}}_{\mathcal{S}_1}$ and with the use of simple linear algebra we have that $\boldsymbol{P}_{(1)}(\underline{\boldsymbol{u}} - \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}}) = \boldsymbol{u}_{(1)} - \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\boldsymbol{u}_{(1)}$, where $\boldsymbol{u}_{(1)} = \underline{\boldsymbol{u}}_{\mathcal{T}/\mathcal{S}_1} + \frac{1}{\beta}\left(\boldsymbol{e}_{j_1}^T\boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}}_{\mathcal{T}/\mathcal{S}_1}\right)\cdot\boldsymbol{e}_{j_1}$. Notice here, that $\mathrm{supp}(\underline{\boldsymbol{u}}) = \mathrm{supp}(\boldsymbol{u}_{(1)}) = \mathcal{T}$ and that the first and third terms of the residual are independent of matrix $\boldsymbol{P}_{(1)}$. Hence, the final form of the residual at step $k = 1$, is:

$$\boldsymbol{r}_{(1)} = \boldsymbol{u}_{(1)} + \boldsymbol{P}_{(1)}\boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}} - \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\boldsymbol{u}_{(1)}. \tag{45}$$

Since matrix $\boldsymbol{P}_{(1)}$ could not be excluded from the second term of the residual, it is required to track down at properties. Here, it should be noted, that we are interested in the norm of the vector $\boldsymbol{P}_{(1)}^T\boldsymbol{e}_l$ for every $l \neq j_1$, instead of the 2-norm of the matrix. Therefore, the l-th row of matrix $\boldsymbol{P}_{(1)}$, i.e.,

$$\boldsymbol{P}_{(1)}^T\boldsymbol{e}_l = \boldsymbol{e}_l + \omega \cdot \boldsymbol{e}_{j_1},$$

is a 2-sparse vector, with $\omega = \frac{1}{\beta}\left(\boldsymbol{e}_{j_1}^T\boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\boldsymbol{e}_l\right)$. Moreover, it is readily seen that,

$$|\omega| \leq \frac{1}{\beta}\left\|\boldsymbol{G}\right\|_2 \leq \frac{\sigma_M^2}{\lambda} < 1, \tag{46}$$

since $1/\beta \leq (\sigma_M^2 + \lambda)/\lambda$ and $\sigma_M < \sqrt{\lambda}$ as observed from (26) (also notice that $\gamma < 1$). Thus, we have that

$$\left\|\boldsymbol{P}_{(1)}^T\boldsymbol{e}_l\right\|_2 = \sqrt{1 + |\omega|^2} < \sqrt{2}. \tag{47}$$

Exploiting the latter bound, we have that

$$\begin{aligned}|\langle\boldsymbol{e}_l, \boldsymbol{P}_{(1)}\boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}}\rangle| &= \left|(\boldsymbol{Q}^T\boldsymbol{P}_{(1)}^T\boldsymbol{e}_l)^T\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}}\right| \leq \left\|\boldsymbol{P}_{(1)}^T\boldsymbol{e}_l\right\|_2 \left\|\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}}\right\|_2 < \\ &< \sqrt{2}\left\|\boldsymbol{F}\right\|_2\left\|\underline{\boldsymbol{\theta}}\right\|_2 \leq \frac{\sqrt{2\lambda}}{2}\left\|\underline{\boldsymbol{\theta}}\right\|_2.\end{aligned} \tag{48}$$

Moreover,

$$\left|\frac{1}{\beta}\boldsymbol{e}_{j_1}^T\boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}}_{\mathcal{T}/\mathcal{S}_1}\right| \leq \frac{\sigma_M^2}{\lambda}\|\underline{\boldsymbol{u}}_{\mathcal{T}/\mathcal{S}_1}\|_2 < \min|\underline{u}| \leq \left|\underline{u}_{j_1}\right|,$$

which leads to

$$\left\|\boldsymbol{u}_{(1)}\right\|_2 < \|\underline{\boldsymbol{u}}\|_2\,. \tag{49}$$

Similarly, we have that

$$\begin{aligned}|r_{(1),i}| = |\langle\boldsymbol{r}_{(1)},\boldsymbol{e}_i\rangle| &= |\langle\boldsymbol{u}_{(1)} + \boldsymbol{P}_{(1)}\boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}} - \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\boldsymbol{u}_{(1)},\boldsymbol{e}_i\rangle| \geq \\ &\geq |\underline{u}_i| - |\langle\boldsymbol{e}_i,\boldsymbol{P}_{(1)}\boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}}\rangle| - |\langle\boldsymbol{e}_i,\boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\boldsymbol{u}_{(1)}\rangle| > \\ &> \min|\underline{u}| - \frac{\sqrt{2\lambda}}{2}\|\underline{\boldsymbol{\theta}}\|_2 - \frac{\sigma_M^2}{\sigma_M^2+\lambda}\left\|\boldsymbol{u}_{(1)}\right\|_2 > \\ &> \min|\underline{u}| - \frac{\sqrt{2\lambda}}{2}\|\underline{\boldsymbol{\theta}}\|_2 - \frac{\sigma_M^2}{\sigma_M^2+\lambda}\|\underline{\boldsymbol{u}}\|_2\,,\end{aligned} \tag{50}$$

for any $i \in \mathcal{T}/\mathcal{S}_1$ and

$$\begin{aligned}|r_{(1),j}| = |\langle\boldsymbol{r}_{(1)},\boldsymbol{e}_j\rangle| &= |\langle\boldsymbol{P}_{(1)}\boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}} - \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\boldsymbol{u}_{(1)},\boldsymbol{e}_j\rangle| \leq \\ &\leq |\langle\boldsymbol{e}_j,\boldsymbol{P}_{(1)}\boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}}\rangle| + |\langle\boldsymbol{e}_j,\boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\boldsymbol{u}_{(1)}\rangle| < \\ &< \frac{\sqrt{2\lambda}}{2}\|\underline{\boldsymbol{\theta}}\|_2 + \frac{\sigma_M^2}{\sigma_M^2+\lambda}\left\|\boldsymbol{u}_{(1)}\right\|_2 < \\ &< \frac{\sqrt{2\lambda}}{2}\|\underline{\boldsymbol{\theta}}\|_2 + \frac{\sigma_M^2}{\sigma_M^2+\lambda}\|\underline{\boldsymbol{u}}\|_2\,,\end{aligned} \tag{51}$$

for all $j \in \mathcal{T}^c$, where (41), (45), (48) and (49) were used. Thus, once again, by imposing that the lower bound in (50) has to be greater than the upper bound in (51), we are led to (26). Hence, it is guaranteed, that at the current step, the column indexed $j_2 \in \mathcal{T}$ is selected and thus $\mathcal{S}_2 = \{j_1, j_2\} \subseteq \mathcal{T}$. Now that we have demonstrated how the method works for the first simple step, we present the general selection step of KGARD.

General k step:
At the k step, $\mathcal{S}_k = \{j_1, j_2, ..., j_k\} \subset \mathcal{T}$ and thus $\boldsymbol{X}_{(k)} = \begin{bmatrix}\boldsymbol{X}_{(0)} & \boldsymbol{I}_{\mathcal{S}_k}\end{bmatrix}$ and $\boldsymbol{B}_{(k)} = \begin{bmatrix}\boldsymbol{I}_{N+1} & \boldsymbol{O}_{(N+1)\times k}\\ \boldsymbol{O}_{(N+1)\times k}^T & \boldsymbol{O}_k\end{bmatrix}$. The least squares step, after the selection of the first column, requires the inversion of the matrix

$$\boldsymbol{D}_{(k)}^T\boldsymbol{D}_{(k)} = \boldsymbol{X}_{(k)}^T\boldsymbol{X}_{(k)} + \lambda\boldsymbol{B}_{(k)} = \begin{bmatrix}\boldsymbol{X}_{(0)}^T\boldsymbol{X}_{(0)} + \lambda\boldsymbol{I}_{N+1} & \boldsymbol{X}_{(0)}^T\boldsymbol{I}_{\mathcal{S}_k}\\ \boldsymbol{I}_{\mathcal{S}_k}^T\boldsymbol{X}_{(0)} & \boldsymbol{I}_k\end{bmatrix}.$$

Applying the *Matrix inversion Lemma*, combined with (35), leads to

$$(\boldsymbol{D}_{(k)}^T\boldsymbol{D}_{(k)})^{-1} = \begin{bmatrix}\boldsymbol{V}\boldsymbol{\Lambda}^{-1}\boldsymbol{V}^T + \boldsymbol{V}\boldsymbol{\Gamma}^T\boldsymbol{Q}^T\boldsymbol{I}_{\mathcal{S}_k}\boldsymbol{W}_{(k)}^{-1}\boldsymbol{I}_{\mathcal{S}_k}^T\boldsymbol{Q}\boldsymbol{\Gamma}\boldsymbol{V}^T & -\boldsymbol{V}\boldsymbol{\Gamma}^T\boldsymbol{Q}^T\boldsymbol{I}_{\mathcal{S}_k}\boldsymbol{W}_{(k)}^{-1}\\ -\boldsymbol{W}_{(k)}^{-1}\boldsymbol{I}_{\mathcal{S}_k}^T\boldsymbol{Q}\boldsymbol{\Gamma}\boldsymbol{V}^T & \boldsymbol{W}_{(k)}^{-1}\end{bmatrix},$$

where $\boldsymbol{W}_{(k)} = \boldsymbol{I}_k - \boldsymbol{I}_{\mathcal{S}_k}^T\boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\boldsymbol{I}_{\mathcal{S}_k}$. Thus, substitution into (34) leads to:

$$\boldsymbol{r}_{(k)} = \boldsymbol{P}_{(k)}\underline{\boldsymbol{u}} + \boldsymbol{P}_{(k)}\boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}} - \boldsymbol{P}_{(k)}\boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}}, \tag{52}$$

where $\boldsymbol{P}_{(k)} = \boldsymbol{I}_N + \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\boldsymbol{I}_{\mathcal{S}_k}\boldsymbol{W}_{(k)}^{-1}\boldsymbol{I}_{\mathcal{S}_k}^T - \boldsymbol{I}_{\mathcal{S}_k}\boldsymbol{W}_{(k)}^{-1}\boldsymbol{I}_{\mathcal{S}_k}^T$. To select an index from the set $\mathcal{T}$, we should impose

$$|r_{(k),i}| > |r_{(k),j}|, \text{ for all } i \in \mathcal{T}/\mathcal{S}_k \text{ and } j \in \mathcal{T}^c.$$

Now $\boldsymbol{P}_{(k)}(\underline{\boldsymbol{u}} - \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}}) = \boldsymbol{u}_{(k)} - \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\boldsymbol{u}_{(k)}$, where $\boldsymbol{u}_{(k)} = \underline{\boldsymbol{u}}_{\mathcal{T}/\mathcal{S}_k} + \boldsymbol{I}_{\mathcal{S}_k}\boldsymbol{W}_{(k)}^{-1}\boldsymbol{I}_{\mathcal{S}_k}^T\boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\underline{\boldsymbol{u}}_{\mathcal{T}/\mathcal{S}_k}$. Hence, the final form of the residual is:

$$\boldsymbol{r}_{(k)} = \boldsymbol{u}_{(k)} + \boldsymbol{P}_{(k)}\boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T\underline{\boldsymbol{\theta}} - \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T\boldsymbol{u}_{(k)}. \tag{53}$$

Following a similar path, for $l \notin \mathcal{S}_k$, we conclude that

$$\boldsymbol{P}_{(k)}^T \boldsymbol{e}_l = \boldsymbol{e}_l + \boldsymbol{I}_{\mathcal{S}_k} \boldsymbol{W}_{(k)}^{-1} \boldsymbol{I}_{\mathcal{S}_k}^T \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T \boldsymbol{e}_l,$$

is a $(k+1)$-sparse vector. Furthermore, it is readily seen that,

$$\left\| \boldsymbol{W}_{(k)}^{-1} \boldsymbol{I}_{\mathcal{S}_k}^T \boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^T \boldsymbol{e}_l \right\|_2 \leq \frac{\sigma_M^2}{\lambda} < 1, \tag{54}$$

which leads to $\left\| \boldsymbol{P}_{(k)}^T \boldsymbol{e}_l \right\|_2 < \sqrt{2}$. Moreover,

$$|\langle \boldsymbol{e}_l, \boldsymbol{P}_{(k)} \boldsymbol{Q}\boldsymbol{F}\boldsymbol{V}^T \underline{\boldsymbol{\theta}} \rangle| < \frac{\sqrt{2\lambda}}{2} \|\underline{\boldsymbol{\theta}}\|_2 \text{ and } \|\boldsymbol{u}_{(k)}\|_2 < \|\underline{\boldsymbol{u}}\|_2. \tag{55}$$

Accordingly, the bounds for the residual are now expressed as

$$|r_{(k),i}| = |\langle \boldsymbol{r}_{(k)}, \boldsymbol{e}_i \rangle| > \min |\underline{u}| - \frac{\sqrt{2\lambda}}{2} \|\underline{\boldsymbol{\theta}}\|_2 - \frac{\sigma_M^2}{\sigma_M^2 + \lambda} \|\underline{\boldsymbol{u}}\|_2 , \tag{56}$$

for any $i \in \mathcal{T}/\mathcal{S}_k$, and

$$|r_{(k),j}| = |\langle \boldsymbol{r}_{(k)}, \boldsymbol{e}_j \rangle| < \frac{\sqrt{2\lambda}}{2} \|\underline{\boldsymbol{\theta}}\|_2 + \frac{\sigma_M^2}{\sigma_M^2 + \lambda} \|\underline{\boldsymbol{u}}\|_2 , \tag{57}$$

for all $j \in \mathcal{T}^c$, where (53) and (55) are used. Finally, imposing the lower bound of (56) to be greater than the upper bound of (57), leads to the condition (26). At the k step, it has been proved that unless the residual length is below the predefined threshold the algorithm will select another correct atom from the identity matrix and the procedure will repeat until $\mathcal{S}_k = \mathcal{T}$. At this point, KGARD has correctly identified all possible outliers and it is up to the tuning of the ϵ parameter whether the procedure terminates (and thus no extra indices are classified as outliers) or it continues and models other extra samples as outliers. □

References

[1] S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective.* Academic Press, 2015.

[2] Å. Björck, *Numerical methods for least squares problems.* Society for Industrial and Applied Mathematics, 1996, no. 51.

[3] G. Papageorgiou, P. Bouboulis, and S. Theodoridis, "Robust linear regression analysis-a greedy approach," *IEEE Transactions on Signal Processing*, vol. 63, no. 15, pp. 3872–3887, 2014.

[4] P. J. Huber, *Wiley Series in Probability and Mathematics Statistics.* Wiley Online Library, 1981.

[5] R. A. Maronna, R. D. Martin, and V. J. Yohai, *Robust statistics.* J. Wiley, 2006.

[6] P. J. Rousseeuw and A. M. Leroy, *Robust regression and outlier detection.* John Wiley & Sons, 2005, vol. 589.

[7] P. J. Huber, "The 1972 wald lecture robust statistics: A review," *The Annals of Mathematical Statistics*, pp. 1041–1067, 1972.

[8] P. J. Rousseeuw and B. C. Van Zomeren, "Unmasking multivariate outliers and leverage points," *Journal of the American Statistical Association*, vol. 85, no. 411, pp. 633–639, 1990.

[9] A. M. Leroy and P. J. Rousseeuw, "Robust regression and outlier detection," *J. Wiley&Sons, New York*, 1987.

[10] S. A. Razavi, E. Ollila, and V. Koivunen, "Robust greedy algorithms for compressed sensing," in *Signal Processing Conference (EUSIPCO), Proceedings of the 20th European.* IEEE, 2012, pp. 969–973.

[11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[12] S. Boyd, "Alternating direction method of multipliers," in *Talk at NIPS Workshop on Optimization and Machine Learning*, 2011.

[13] D. P. Wipf and B. D. Rao, "Sparse bayesian learning for basis selection," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2153–2164, 2004.

[14] Y. Jin and B. D. Rao, "Algorithms for robust linear regression by exploiting the connection to sparse signal recovery," in *International Conference on Acoustics Speech and Signal Processing (ICASSP).* IEEE, 2010, pp. 3830–3833.

[15] G. Mateos and G. B. Giannakis, "Robust nonparametric regression via sparsity control with application to load curve data cleansing," *IEEE Transactions on Signal Processing*, vol. 60, no. 4, pp. 1571–1584, 2012.

[16] K. Mitra, A. Veeraraghavan, and R. Chellappa, "Robust RVM regression using sparse outlier model," in *Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 2010, pp. 1887–1894.

[17] P. Bouboulis and S. Theodoridis, "Kernel methods for image denoising," in *Regularization, optimization, kernels, and support vector machines*, J. Suykens, M. Signoretto, and A. Argyriou, Eds., 2015.

[18] P. Bouboulis, K. Slavakis, and S. Theodoridis, "Adaptive kernel-based image denoising employing semi-parametric regularization," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1465–1479, 2010.

[19] Y. C. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Conference on Signals, Systems and Computers, Conference Record of The Twenty-Seventh Asilomar.* IEEE, 1993, pp. 40–44.

[20] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[21] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

[22] A. J. Smola and B. Schölkopf, *Learning with Kernels.* The MIT Press, 2002.

[23] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[24] K. Slavakis, P. Bouboulis, and S. Theodoridis, *Signal Processing Theory and Machine Learning: Online Learning in Reproducing Kernel Hilbert Spaces*, ser. Academic Press Library in Signal Processing, 2014, ch. 17.

[25] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, 4th Edition.* Academic press, 2008.

[26] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American Mathematical Society*, vol. 68, no. 3, pp. 337–404, May 1950.

[27] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM review*, vol. 51, no. 1, pp. 34–81, 2009.

[28] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[29] M. Vetterli and T. Kalker, "Matching pursuit for compression and application to motion compensated video coding," in *IEEE International Conference on Image Processing, 1994 (ICIP-94.)*, vol. 1. IEEE, 1994, pp. 725–729.

[30] S. Mallat, *A wavelet tour of signal processing: the sparse way.* Academic press, 2008.

[31] L. Rebollo-Neira and D. Lowe, "Optimized orthogonal matching pursuit approach," *Signal Processing Letters, IEEE*, vol. 9, no. 4, pp. 137–140, 2002.

[32] D. Needell and R. Vershynin, "Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 310–316, 2010.

[33] D. Needell and J. A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 301–321, 2009.

[34] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, "Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 58, no. 2, pp. 1094–1121, 2012.

[35] J. Wang, S. Kwon, and B. Shim, "Generalized orthogonal matching pursuit," *IEEE Transactions on Signal Processing*, vol. 60, no. 12, pp. 6202–6216, 2012.

[36] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[37] R. J. Tibshirani *et al.*, "The lasso problem and uniqueness," *Electronic Journal of Statistics*, vol. 7, pp. 1456–1490, 2013.

[38] E. J. Candes, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted 1 minimization," *Journal of Fourier analysis and applications*, vol. 14, no. 5-6, pp. 877–905, 2008.

[39] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *The journal of machine learning research*, vol. 1, pp. 211–244, 2001.

[40] G. Papageorgiou, P. Bouboulis, and S. Theodoridis, "Robust kernel-based regression using orthogonal matching pursuit," in *Machine Learning for Signal Processing (MLSP), 2013 IEEE International Workshop on.* IEEE, 2013, pp. 1–6.

[41] B. L. Sturm and M. G. Christensen, "Comparison of orthogonal matching pursuit implementations," in *Signal Processing Conference (EUSIPCO), Proceedings of the 20th European.* IEEE, 2012, pp. 220–224.

[42] W. Gander, *On the linear least squares problem with a quadratic constraint.* Computer Science Department, Stanford University, 1978.

[43] ——, "Least squares with a quadratic constraint," *Numerische Mathematik*, vol. 36, no. 3, pp. 291–307, 1980.

[44] M. Rojas and D. C. Sorensen, “A trust-region approach to the regularization of large-scale discrete forms of ill-posed problems,” *SIAM Journal on Scientific Computing*, vol. 23, no. 6, pp. 1842–1860, 2002.

[45] J. Portilla, V. Strela, M. Wainwright, and E. P. Simoncelli, “Image denoising using scale mixtures of gaussians in the wavelet domain,” *IEEE Transactions on Image Processing*, vol. 12, no. 11, pp. 1338–1351, 2003.

[46] L. Sendur and I. Selesnick, “Bivariate shrinkage functions for wavelet-based denoising exploiting interscale dependency,” *IEEE Transactions on Signal Processing*, vol. 50, no. 11, pp. 2744–2756, 2002.

[47] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, “Image denoising by sparse 3d transform-domain collaborative filtering,” *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[48] K. Seongjai, “PDE-based image restoration : A hybrid model and color image denoising,” *IEEE Transactions on Image Processing*, vol. 15, no. 5, pp. 1163–1170, 2006.

[49] H. Takeda, S. Farsiu, and P. Milanfar, “Kernel regression for image processing and reconstruction,” *IEEE Tranactions on Image Processing*, vol. 16, no. 2, pp. 349–366, 2007.