algorithms

MDPI

Article

Aiding Dictionary Learning Through Multi-Parametric Sparse Representation

Florin Stoican [1,*] and Paul Irofti [2]

[1] Department of Automatic Control and Computers, University Politehnica of Bucharest, 313 Spl. Independenței, 060042 Bucharest, Romania

[2] The Research Institute of the University of Bucharest (ICUB) and Department of Computer Science, University of Bucharest, Bulevardul M. Kogălniceanu 36-46, 050107 Bucharest, Romania; paul@irofti.net

* Correspondence: florin.stoican@acse.pub.ro

Received: 20 May 2019; Accepted: 25 June 2019; Published: 28 June 2019

Abstract: The ℓ_1 relaxations of the sparse and cosparse representation problems which appear in the dictionary learning procedure are usually solved repeatedly (varying only the parameter vector), thus making them well-suited to a multi-parametric interpretation. The associated constrained optimization problems differ only through an affine term from one iteration to the next (i.e., the problem's structure remains the same while only the current vector, which is to be (co)sparsely represented, changes). We exploit this fact by providing an explicit, piecewise affine with a polyhedral support, representation of the solution. Consequently, at runtime, the optimal solution (the (co)sparse representation) is obtained through a simple enumeration throughout the non-overlapping regions of the polyhedral partition and the application of an affine law. We show that, for a suitably large number of parameter instances, the explicit approach outperforms the classical implementation.

Keywords: dictionary learning; multi-parametric problem; sparse representation; cosparse representation; cross-polytopic constraint; rank-deficient quadratic cost

1. Introduction

We commonly represent m-dimensional data, or signals, through a collection of m linearly independent vectors that together form a base. For certain m-dimensional signals, or classes of signals, we have prior information or insight regarding that they are in fact perturbed versions of a much simpler, original, true signal that lies in an s-dimensional sub-space. In order to be able to retrieve the original signal, we need to have access to a proper base, containing the right s vectors that were used when the true signal was created. With such limited information, it is hard to pick such a base, especially for a large class of signals, which is why it is common to extend it with more vectors, making it a redundant base, also called a frame or, as will be used through out this paper, a dictionary. The dictionary has $n > m$ columns that are also called atoms. The idea is to have more options from which to choose s vectors.

Assume we are given a good dictionary $\boldsymbol{D} \in \mathbb{R}^{m \times n}$ that we want to use to represent a signal $\boldsymbol{y} \in \mathbb{R}^m$ with just s atoms. This leads to $\binom{s}{n}$ possible choices. As the number of atoms increases, so does the complexity of finding the best choice. Even when we decide on a set of atoms, there is still the question of their associated coefficients in the direction they are pointing towards.

The task of finding the atoms and their coefficients is called the sparse representation problem [1]. Existing algorithms pursue this task for each signal $\boldsymbol{y}$ that they receive, by solving a particular optimization problem based on $\boldsymbol{y}$ that provides the s-sparse representation $\boldsymbol{x}$.

Solving the sparse representation problem for multiple vectors $\boldsymbol{y}$ is in fact the repeated resolution of a constrained optimization problem which differs only through the m-dimensional vector $\boldsymbol{y}$ which

is to be represented. This has led us to consider a multi-parametric interpretation where the vector $\boldsymbol{y}$ becomes a parameter which governs the optimization's problem solution and its domain of validity.

While the multi-parametric approach is well-known in the optimization field [2], to the best of our knowledge, it appears not to have been employed by the dictionary learning (DL) community.

Conceptually, the idea is simple: instead of solving repeatedly very similar optimization problems (with the same structure and differing only in the current value of a parameter), we instead break the problem into two distinct stages:

(i) the offline stage: find the way in which the optimal solution depends on parameter $\boldsymbol{y}$ and store this information for later use in the online stage;
(ii) the online stage: for the current value of $\boldsymbol{y}$, retrieve and apply the a priori computed solution.

Further assume that the cost is quadratic and the constraints linear leads, in the offline stage, to a piecewise formulation with a polyhedral support [3,4]. That is, the optimal solution is an affine law in parameter $\boldsymbol{y}$ which is active on a so-called polyhedral *critical region*. Enumerating [5] all these disjoint critical regions covers completely the parameter space (or the restriction over which $\boldsymbol{y}$ is defined). Thus, at the online stage, it remains only to find the region in which the current $\boldsymbol{y}$ lies [6–8], retrieve the associated affine law and apply it. For further details with a practical bent, see the survey [9] which deals with the subtleties of multi-parametric problem in the context of *explicit MPC*.

As a first step, we rewrite the sparse representation problem in its dual form (Karash–Kuhn–Tucker) in order to enumerate sets of active constraints from which the piecewise affine law and its polyhedral support are deduced. A couple of issues differentiate this approach from other formulations encountered in the literature (e.g., [8,10] which solve a generic complementarity problem):

(i) the cost from the constrained optimization problem is rank-deficient (the dictionary is over-determined which means that the quadratic cost is defined by a semi-definite matrix);
(ii) the ℓ_1 norm used here (as a relaxation from the sparse restriction induced by the ℓ_0 norm) leads to a very-particular set of linear constraints (they define a cross-polytope, whose structure influences the optimization problem formulation [11]).

Tackling these issues leads to a particular multi-parametric formulation and represents the main contribution of this paper (beyond the usage of the multi-parametric formalism in the DL context):

(i) The first issue requires a careful decomposition of the matrices (in order to avoid degeneracy in the formulations [12]); the upshot is that the particular KKT representation appears in a simple form (which allows simple matrix decompositions).
(ii) The second requires to consider a vertex-representation of the cross-polytope (as this is a more compact representation than its equivalent half-space representation [13]). In general, the opposite is true: to a reasonable number of linear inequalities corresponds a significantly larger number of vertices. Thus, most if not all of space partitioning induced by the multi-parametric representation exploit the "half-space" description of the feasible domain [9].
(iii) We highlight a compact storage and retrieval procedure (which exploits the symmetry of the cross-polytope domain) with the potential to significantly reduce the numerical issues.

For a better understanding of the issues, we also consider some of the specific shortcomings of multi-parametric representations: the over-approximation of quadratic constraints (as appear whenever the ℓ_2 norm is used as a constraint; the enumeration problem when dealing with large-size parameters and some ideas for alleviating the issue [14].

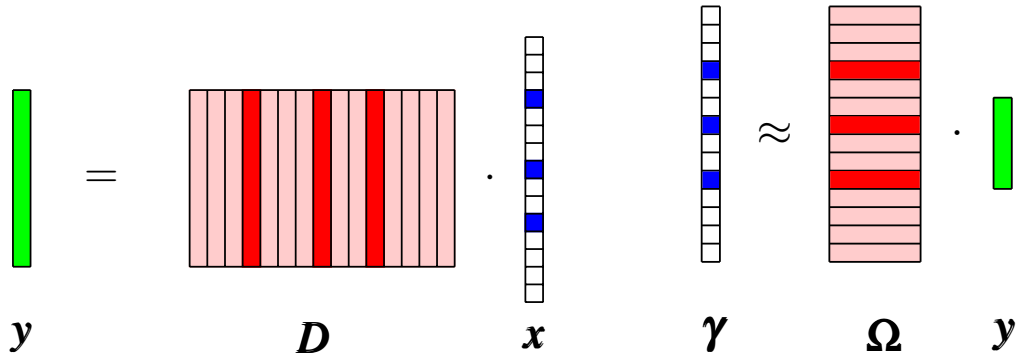
The rest of the paper is organized as follows. Section 2 presents the sparse and cosparse representations. Section 3 gives a multi-parametric solution for a generic optimization problem which is then particularized for the (co)sparse representation case in Section 4. The multi-parametric approach is tested over a standard example and comparison with the standard case is shown in Section 5. The conclusions are drawn in Section 6.

2. The Dictionary Learning Problem

Given the signal $\boldsymbol{y} \in \mathbb{R}^m$ and the fixed dictionary $\boldsymbol{D} \in \mathbb{R}^{m \times n}$, the sparse representation problem imposes an error bound ε and seeks the sparsest representation

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{x}\|_0 \quad \text{s.t.} \quad \|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}\| \leq \varepsilon, \tag{1}$$

where $\|.\|_0$ is the ℓ_0 pseudo-norm counting the non-zero entries of $\boldsymbol{x}$. The process is depicted in Figure 1a, where the full signal $\boldsymbol{y}$ uses $s = 3$ dictionary atoms for its sparse representation $\boldsymbol{x}$.



(**a**) Sparse representation. (**b**) Cosparse representation.

Figure 1. Sparse/cosparse representation of a signal ([15], Chapter 1 and 10). The used atoms are red and the nonzero coefficients are blue. The unused atoms are pink.

Due to the non-convex nature of the ℓ_0-norm, we prefer its convex ℓ_1 relaxation

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}\|^2 \quad \text{s.t.} \quad \|\boldsymbol{x}\|_1 \leq \theta, \tag{2}$$

which is known as the *lasso* problem [16]. Here, θ is a sparsity promoting bound for the 1-norm of the solution. The focus in (2) is on the approximation, such that the objective is minimized up to a point where $\boldsymbol{x}$ has too many large non-zero elements. If instead we desire the sparsest ℓ_1 solution satisfying a given approximation quality ε, the optimization problem becomes

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{x}\|_1 \quad \text{s.t.} \quad \|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}\|^2 \leq \varepsilon. \tag{3}$$

Given a set of $\boldsymbol{Y} \in \mathbb{R}^{m \times N}$ training signals, the dictionary learning (DL) problem [15] is

$$\min_{\boldsymbol{D},\boldsymbol{X}} \quad \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 \quad \text{s.t.} \quad \|\boldsymbol{x}_\ell\|_1 \leq \theta,\ \ell = 1:N, \quad \left\|\boldsymbol{d}_j\right\| = 1,\ j = 1:n, \tag{4}$$

where $\boldsymbol{D} \in \mathbb{R}^{m \times n}$ is the dictionary, whose columns are also called atoms, and $\boldsymbol{X} \in \mathbb{R}^{n \times N}$ are the resulting sparse representations. Note that (4), in contrast to (1), does not only find the sparse representation $\boldsymbol{X}$ but also learns the dictionary $\boldsymbol{D}$.

As depicted in Algorithm 1, problem (4) is usually solved by alternative optimization: fixing $\boldsymbol{D}$ we enter the sparse representation stage (step 2) where we compute the representations $\boldsymbol{X}$ by solving either of (1)–(3) for each signal $\boldsymbol{y} \in \boldsymbol{Y}$, and then we enter the dictionary update stage (step 3) where we fix $\boldsymbol{X}$ and update $\boldsymbol{D}$ by taking each atom $\boldsymbol{d} \in \boldsymbol{D}$ and refining it based on the signals that use it in their sparse representation. The process is repeated K times (step 1) until the objective in (4) converges or the error becomes satisfactory.

Algorithm 1: Alternate optimization dictionary learning

Data: signals set $\boldsymbol{Y} \in \mathbb{R}^{m \times N}$
initial dictionary $\boldsymbol{D} \in \mathbb{R}^{m \times n}$
number of iterations K
representation criteria s, ε, or θ
Result: trained dictionary $\boldsymbol{D}$

for $k = 1 : K$ **do**
  Sparse representation: keeping $\boldsymbol{D}$ fixed, obtain $\boldsymbol{x_i}$ by solving (1), (2), or (3), $i = 1 : N$
  Dictionary update: keeping the resulting representations $\boldsymbol{X}$ fixed, solve (4) to obtain dictionary $\boldsymbol{D}$;

2.1. The Cosparse Model

While the sparse representation is interested in finding the non-zero coefficients corresponding to a low-dimensional subspace where signal $\boldsymbol{x}$ lies, the cosparse representation is interested in finding its complementary subspace, the subspace orthogonal to $\boldsymbol{x}$. The sparse model can be seen as a synthesis process, focused on constructing $\boldsymbol{x}$ based on the dictionary, and the cosparse model as its analysis counter-part, where the focus is on the original signal $\boldsymbol{y}$ and its null subspace.

As depicted in Figure 1b, in the cosparse model [17], we apply the analysis dictionary $\boldsymbol{\Omega} \in \mathbb{R}^{n \times m}$ to the signal $\boldsymbol{y}$ and obtain the sparse vector

$$\gamma = \boldsymbol{\Omega} \boldsymbol{y}. \tag{5}$$

The atoms are now the rows of dictionary $\boldsymbol{\Omega}$, of which only a few are used in the multiplication with $\boldsymbol{y}$. Unlike the sparse model, here we are interested in the atoms that are not used by $\boldsymbol{y}$, the atoms that are orthogonal to it and create the zero entries in γ.

Assume that signal $\boldsymbol{y}$ is the noisy version of the original, unknown, signal $\boldsymbol{z} \in \mathbb{R}^m$ that lies in a lower-dimensional subspace

$$\boldsymbol{y} = \boldsymbol{z} + \boldsymbol{v}, \tag{6}$$

where $\boldsymbol{v}$ is white additive noise. Let $\boldsymbol{\Omega} \in \mathbb{R}^{n \times m}$ be the given cosparse dictionary. Then, recovering $\boldsymbol{z}$ means finding the low-dimensional subspace to which it belongs and implies finding the set of atoms Λ from $\boldsymbol{\Omega}$ orthogonal to $\boldsymbol{z}$. In the literature, Λ is sometimes called the cosupport of $\boldsymbol{z}$.

The ℓ_0 optimization problem becomes

$$\min_{z,\Lambda} \quad \|\boldsymbol{y} - \boldsymbol{z}\|^2 \quad \text{s.t.} \quad \boldsymbol{\Omega}_\Lambda \boldsymbol{z} = 0, \quad \text{rank}(\boldsymbol{\Omega}_\Lambda) = m - s, \tag{7}$$

where we are interested in minimizing the distance between $\boldsymbol{y}$ and $\boldsymbol{z}$ such that the cosupport defines an orthogonal $m - s$ subspace in which $\boldsymbol{z}$ lies. Thus, applying the cosparse dictionary leads to an s-sparse signal as in Figure 1b.

For our purposes, we relax the equality constraints of (7) and move to its convex form

$$\min_{z,\Lambda} \quad \|\boldsymbol{y} - \boldsymbol{z}\|^2 \quad \text{s.t.} \quad \|\boldsymbol{\Omega}_\Lambda \boldsymbol{z}\|_1 \leq \theta, \quad \text{rank}(\boldsymbol{\Omega}_\Lambda) \leq m - s, \tag{8}$$

where we induce sparsity using the ℓ_1-norm just like we did in (2). Focusing on sparsity instead of approximation, we can rewrite (8) as

$$\min_{z,\Lambda} \quad \|\boldsymbol{\Omega}_\Lambda \boldsymbol{z}\|_1 \quad \text{s.t.} \quad \|\boldsymbol{y} - \boldsymbol{z}\|^2 \leq \varepsilon, \quad \text{rank}(\boldsymbol{\Omega}_\Lambda) \leq m - s. \tag{9}$$

3. Analysis of the Multi-Parametric Formulation Induced by the DL Problem

Let us consider the constrained optimization problem

$$\min_{\boldsymbol{\xi}} \quad \|\boldsymbol{T}\boldsymbol{\xi} - \boldsymbol{y}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\Delta}\boldsymbol{\xi}\|_1 \leq \delta, \tag{10}$$

with $\boldsymbol{\xi} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{T} \in \mathbb{R}^{m \times n}$, $\boldsymbol{\Delta} \in \mathbb{R}^{p \times n}$. $\delta \in \mathbb{R}_+$ is a positive scalar.

Note that we make no assumption on the relation between space dimensions m, n, p. In fact, we will be mostly interested in the degenerate case $n > m$, which implies that mapping $\boldsymbol{T}\boldsymbol{\xi}$ is a projection from a higher dimensional space to a lower one (from $\mathbb{R}^n$ to $\mathbb{R}^m$).

The goal of (10) is to provide the constrained optimal solution $\boldsymbol{\xi}^\star$ minimizing the cost while simultaneously respecting the constraints. While multiple [2] tools and theoretical algorithms can be employed to solve (10), in here we propose to exploit two of its particularities:

(i) the ℓ_1 norm constraint characterizes a scaled and/or projected cross-polytope;
(ii) the cost is affinely parametrized after parameter $\boldsymbol{y}$.

3.1. Geometrical Interpretation of the ℓ_1 Norm

The first item stems from a well-known result from polyhedral sets' state of the art [18]: the norm inequality $\|\boldsymbol{\xi}\|_1 \leq 1$ describes the n-th order cross-polytope (the dual of a hypercube):

$$C_n = \text{conv}\{\pm \boldsymbol{e}_i\} = \{\boldsymbol{\xi} \in \mathbb{R}^n : \|\boldsymbol{\xi}\|_1 \leq 1\}, \tag{11}$$

where $\boldsymbol{e}_i$ are the column vectors that have '1' on the i-th position and '0' anywhere else. Consequently, it follows that $\|\boldsymbol{\Delta}\boldsymbol{\xi}\|_1 \leq \delta$ is equivalent with

$$\{\boldsymbol{\xi} \in \mathbb{R}^n : \|\boldsymbol{\Delta}\boldsymbol{\xi}\|_1 \leq \delta\} = \text{conv}\{\pm \boldsymbol{\Delta}_i^+\} \cdot \delta, \tag{12}$$

with $\boldsymbol{\Delta}_i^+ \in \mathbb{R}^n$, the column vectors of matrix $\boldsymbol{\Delta}^+$, the pseudo-inverse of matrix $\boldsymbol{\Delta}$. Through (11), we have that $\boldsymbol{\Delta}\boldsymbol{\xi} = \text{conv}\{\pm \boldsymbol{e}_i\}$. Taking $\boldsymbol{\Delta}^+$ which verifies $\boldsymbol{\Delta}^+\boldsymbol{\Delta} = \boldsymbol{I}$ allows for writing relation (12). Writing explicitly the convex sum from (12), we arrive at

$$\{\boldsymbol{\xi} \in \mathbb{R}^n : \|\boldsymbol{\Delta}\boldsymbol{\xi}\|_1 \leq \delta\} = \{\boldsymbol{\xi} : \boldsymbol{\xi} = \boldsymbol{F}\boldsymbol{\alpha},\ \boldsymbol{\alpha} \geq 0,\ \mathbf{1}^\top \boldsymbol{\alpha} = \delta\}. \tag{13}$$

Note that we made the notation (for compactness reasons) $\boldsymbol{F} = \begin{bmatrix} \boldsymbol{\Delta}^+ & -\boldsymbol{\Delta}^+ \end{bmatrix}$ and that $\boldsymbol{\alpha} \in \mathbb{R}^{2n}$. Any permutation of column vectors $\pm \boldsymbol{\Delta}_i^+$ in $\boldsymbol{F}$ is equally valid. We have simply chosen this form as it permits for compactly writing $\boldsymbol{F}$ with respect to $\boldsymbol{\Delta}$. With these notations, we rewrite (10) into the form:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \|\boldsymbol{T}\boldsymbol{F}\boldsymbol{\alpha} - \boldsymbol{y}\|_2^2, \\ \text{s.t.} \quad & \boldsymbol{\alpha} \geq 0,\ \mathbf{1}^\top \boldsymbol{\alpha} = \delta. \end{aligned} \tag{14}$$

Note that the optimal solution of (14) is $\boldsymbol{\alpha}^\star$ and, thus, $\boldsymbol{\xi}^\star = \boldsymbol{F}\boldsymbol{\alpha}^\star$.

Remark 1. *Any polyhedral set has a dual representation: half-space (as a finite intersection of linear inequalities) and convex (as a convex sum of extreme points and linear combination of rays—in which case it is not bounded). Most often, the half-space representation is preferred since it is more compact (there are usually many fewer constraints than extreme points). On the other hand, the cross-polytope is the go-to counter-example: there are* $2n$ *extreme points in* (11) *but to these correspond* 2^n *inequalities in the half-space representation. It is thus significantly more efficient to exploit the convex representation in this particular case.*

3.2. Karush–Kuhn–Tucker Form

The second particularity of (10) (or, equivalently, of (14)) is that the optimization depends on parameter $\boldsymbol{y}$. Inspired by the approach followed in explicit MPC constructions [9] (and which, to the best of our knowledge, has no equivalence in the dictionary learning state of the art), we propose to provide an explicit description of the optimal solution in terms of parameter $\boldsymbol{y}$.

The first step is to rewrite (14) in its dual form (i.e., the Karash–Kuhn–Tucker representation—see [2] or any standard optimization book for a more in-depth description):

$$\text{stationarity:} \quad \nabla\left(\|\boldsymbol{y}-\boldsymbol{TF\alpha}\|_2^2\right)+\nabla\left(-\boldsymbol{\alpha}\right)\cdot\boldsymbol{\lambda}+\nabla\left(\mathbf{1}^\top\boldsymbol{\alpha}\right)\cdot\mu=0, \tag{15a}$$

$$\text{primal feasibility:} \quad -\boldsymbol{\alpha}\leq 0,\ \mathbf{1}^\top\boldsymbol{\alpha}=\delta, \tag{15b}$$

$$\text{dual feasibility:} \quad \boldsymbol{\lambda}\geq 0, \tag{15c}$$

$$\text{complementarity:} \quad \boldsymbol{\lambda}\times\left(-\boldsymbol{\alpha}\right)=0. \tag{15d}$$

The ∇ operator denotes the gradient operation applied for: the cost, the inequalities and the equality appearing in the primal problem (14). $\boldsymbol{\lambda}\in\mathbb{R}^{2n}$ and $\mu\in\mathbb{R}$ are the Lagrangian multipliers (also called dual variables) associated with the inequalities, respectively the equality constraint of (14). '$\times$' denotes the complementarity condition which requires that either the i-th Lagrangian multiplier λ_i is zero or the i-th inequality is active ($-\alpha_i\leq 0\mapsto -\alpha_i=0$).

Applying the gradient, the KKT form (15) becomes

$$2\left(\boldsymbol{TF}\right)^\top\cdot\left(\boldsymbol{TF\alpha}-\boldsymbol{y}\right)-\boldsymbol{\lambda}+\mathbf{1}\mu=0, \tag{16a}$$

$$\boldsymbol{\lambda}\geq 0,\ -\boldsymbol{\alpha}\leq 0,\ \mathbf{1}^\top\boldsymbol{\alpha}=\delta, \tag{16b}$$

$$\boldsymbol{\lambda}\times\left(-\boldsymbol{\alpha}\right)=0. \tag{16c}$$

Under Slater's conditions of optimality [2], the primal variable $\boldsymbol{\alpha}$ and dual variables $\boldsymbol{\lambda},\mu$ verifying (16) describe the optimum (primal and dual).

Using the well-known result that $\text{rank}(AB)\leq\min(\text{rank}(A)\,\text{rank}(B))$ and that for a matrix $A\in\mathbb{R}^{r\times q}$, $\text{rank}(A)\leq\min(r,q)$, we have that $\text{rank}(\boldsymbol{TF})\leq\min\left(\min(m,n),\min(n,2n)\right)=\min(m,n)$. Thus, the semi-definite matrix $\left(\boldsymbol{TF}\right)^\top\boldsymbol{TF}$ appearing in (16a) is rank-deficient (is a square matrix of dimension $2n$ but its rank is at most $\min(m,n)$). Hereinafter, we consider (the opposite case would unfold in the same manner and is not followed here) that $m\leq n$ and proceed accordingly (i.e., $\min(m,n)=m$).

As a next step, we consider the SVD decomposition $\boldsymbol{T\Delta}=\boldsymbol{U\Sigma W}^\top$ with $\boldsymbol{U}\in\mathbb{R}^{m\times m}$ and $\boldsymbol{V}\in\mathbb{R}^{n\times n}$ orthogonal matrices and $\boldsymbol{\Sigma}\in\mathbb{R}^{m\times n}$ the matrix containing the singular values of $\boldsymbol{T\Delta}$. This decomposition allows for writing $\boldsymbol{TF}=\boldsymbol{U\Sigma V}^\top$, where $\boldsymbol{V}^\top=\begin{bmatrix}\boldsymbol{W}^\top & -\boldsymbol{W}^\top\end{bmatrix}\in\mathbb{R}^{n\times 2n}$. For further use, we consider the decompositions $\boldsymbol{\Sigma}=\begin{bmatrix}\boldsymbol{\Sigma_1} & \mathbf{0}\end{bmatrix}$ and $\boldsymbol{V}^\top=\begin{bmatrix}V_1 & V_2\end{bmatrix}^\top$, where $\boldsymbol{\Sigma_1}\in\mathbb{R}^{m\times m}$ and $\boldsymbol{V}_1^\top\in\mathbb{R}^{m\times 2n}$, $\boldsymbol{V}_2^\top\in\mathbb{R}^{(n-m)\times 2n}$. Introducing these notations in (16) allows for rewriting (16a) as follows:

$$\boldsymbol{V}_1^\top\boldsymbol{\alpha}-\left(\boldsymbol{\Sigma}_\mathbf{1}^{-1}\right)^\top\boldsymbol{U}^\top\boldsymbol{y}-\frac{1}{2}\left(\boldsymbol{\Sigma}_\mathbf{1}^{-2}\right)^\top\boldsymbol{V}_1^\top\left(\boldsymbol{\lambda}-\mathbf{1}\mu\right)=0, \tag{17a}$$

$$\boldsymbol{V}_2^\top\left(\boldsymbol{\lambda}-\mathbf{1}\mu\right)=0. \tag{17b}$$

Remark 2. *Note that we assume that $\boldsymbol{\Sigma_1}$ is full-rank, which, coupled with its square form, implies that it is non-singular (and consequently admits an inverse). Particular cases where $\boldsymbol{\Sigma_1}$ is singular can be treated similarly, by extracting the non-singular part and resizing accordingly $\boldsymbol{V}_1^\top,\boldsymbol{V}_2^\top$.*

3.3. Multi-Parametric Interpretation

The difficulty of solving (16), even when making use of (17), lies in the complementarity condition (16c). There are multiple formulations in the literature [4,8,10], each proposing a particular 'flavor'. In what follows, we make use of one of the most common approaches: the *active set* method [6]. This approach enumerates (the selection of sets of active constraints and the stop criteria for the search of feasible combinations are subject to significant research and still an open topic [14]) combinations of active (those for which $\lambda_i > 0$) and inactive constraints (those for which $\lambda_i = 0$). Subsequently, problem (16) is solved for each of the particular combinations of active constraints.

As a first step, we form submatrices associated with the active/inactive set of constraints ($\widetilde{(\cdot)}$ for those corresponding to active constraints and $\widehat{(\cdot)}$ for those corresponding to inactive constraints (for example, $\tilde{\boldsymbol{V}}_2^\top$ is the matrix constructed from the columns of $\boldsymbol{V}_2^\top$ which correspond to active constraints—those for which $\tilde{\alpha}_i = 0$) and reformulate (16) correspondingly:

$$\begin{bmatrix} \hat{\boldsymbol{V}}_1^\top \\ \mathbf{1}^\top \end{bmatrix} \hat{\boldsymbol{\alpha}} - \begin{bmatrix} \left(\boldsymbol{\Sigma}_1^{-1}\right)^\top \boldsymbol{U}^\top \boldsymbol{y} + \frac{1}{2}\left(\boldsymbol{\Sigma}_1^{-2}\right)^\top \left(\tilde{\boldsymbol{V}}_1^\top \tilde{\boldsymbol{\lambda}} - \boldsymbol{V}_1^\top \mathbf{1}\mu\right) \\ \delta \end{bmatrix} = 0, \tag{18a}$$

$$\tilde{\boldsymbol{V}}_2^\top \tilde{\boldsymbol{\lambda}} - \boldsymbol{V}_2^\top \mathbf{1}\mu = 0, \tag{18b}$$

$$\tilde{\boldsymbol{\lambda}} > 0,\ \tilde{\boldsymbol{\alpha}} = 0, \tag{18c}$$

$$\hat{\boldsymbol{\lambda}} = 0,\ \hat{\boldsymbol{\alpha}} > 0. \tag{18d}$$

We note with $\tilde{m}, \hat{m}$ the number of active, respectively inactive, constraints (hence $\tilde{m} + \hat{m} = n$).

At this stage, most multi-parametric formulations employ various heuristics for the selection of the set of active constraints [5,8] and solve (18). We proceed similarly with the observation that the cardinality of the set of active constraints has to lead to a well-defined problem (18). In other words, we expect that $\hat{m} \leq m + 1$ (i.e., the number of rows in the matrix left-multiplying $\hat{\boldsymbol{\alpha}}$ in (18a) is at least equal with the number of columns, which means that the matrix admits a full-rank pseudo-inverse).

Having to accommodate the case $\hat{m} < m + 1$ requires a further partitioning of the matrices appearing in (18), this time in row-blocks: we add the subscript '1' for the sub-matrices formed from the first $\hat{m} - 1$ rows and subscript '2' for the sub-matrices formed from the remaining $m - \hat{m} + 1$ rows. Note that, as before, the partitioning is not unique and any permutation which does not change the number of rows is admissible. For example, $\hat{\boldsymbol{V}}_1^\top \in \mathbb{R}^{m \times \hat{m}}$ is partitioned into $\hat{\boldsymbol{V}}_1^\top = \begin{bmatrix} \hat{\boldsymbol{V}}_{11} & \hat{\boldsymbol{V}}_{12} \end{bmatrix}^\top$ where $\hat{\boldsymbol{V}}_{11}^\top \in \mathbb{R}^{(\hat{m}-1) \times \hat{m}}$ and $\hat{\boldsymbol{V}}_{12}^\top \in \mathbb{R}^{(m-\hat{m}+1) \times \hat{m}}$. With these notations, (18a)–(18b) become:

$$\begin{bmatrix} \hat{\boldsymbol{V}}_{11}^\top \\ \mathbf{1}^\top \end{bmatrix} \hat{\boldsymbol{\alpha}} - \begin{bmatrix} \left(\boldsymbol{\Sigma}_{11}^{-1}\right)^\top \boldsymbol{U}^\top \boldsymbol{y} + \frac{1}{2}\left(\boldsymbol{\Sigma}_{11}^{-2}\right)^\top \left(\tilde{\boldsymbol{V}}_1^\top \tilde{\boldsymbol{\lambda}} - \boldsymbol{V}_1^\top \mathbf{1}\mu\right) \\ \delta \end{bmatrix} = 0, \tag{19a}$$

$$\hat{\boldsymbol{V}}_{12}^\top \hat{\boldsymbol{\alpha}} - \left[\left(\boldsymbol{\Sigma}_{12}^{-1}\right)^\top \boldsymbol{U}^\top \boldsymbol{y} + \frac{1}{2}\left(\boldsymbol{\Sigma}_{12}^{-2}\right)^\top \left(\tilde{\boldsymbol{V}}_1^\top \tilde{\boldsymbol{\lambda}} - \boldsymbol{V}_1^\top \mathbf{1}\mu\right)\right] = 0, \tag{19b}$$

$$\tilde{\boldsymbol{V}}_2^\top \tilde{\boldsymbol{\lambda}} - \boldsymbol{V}_2^\top \mathbf{1}\mu = 0. \tag{19c}$$

Rearranging (19) to highlight the unknown terms ($\hat{\boldsymbol{\alpha}}$, $\tilde{\boldsymbol{\lambda}}$ and μ) leads to the following compact form:

$$\begin{bmatrix} \hat{\boldsymbol{V}}_{11}^\top \\ \mathbf{1}^\top \end{bmatrix} \hat{\boldsymbol{\alpha}} = \begin{bmatrix} \left(\boldsymbol{\Sigma}_{11}^{-1}\right)^\top \boldsymbol{U}^\top \\ 0 \end{bmatrix} \boldsymbol{y} + \begin{bmatrix} 0 \\ \delta \end{bmatrix} + \begin{bmatrix} \left(\frac{1}{2}\boldsymbol{\Sigma}_{11}^{-2}\right)^\top \\ 0 \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{V}}_1^\top & -\boldsymbol{V}_1^\top \mathbf{1} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\lambda}} \\ \mu \end{bmatrix}, \tag{20a}$$

$$\begin{bmatrix} \frac{1}{2}\left(\boldsymbol{\Sigma}_{12}^{-2}\right)^\top \tilde{\boldsymbol{V}}_1^\top & -\frac{1}{2}\left(\boldsymbol{\Sigma}_{12}^{-2}\right)^\top \boldsymbol{V}_1^\top \mathbf{1} \\ \tilde{\boldsymbol{V}}_2^\top & -\boldsymbol{V}_2^\top \mathbf{1} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\lambda}} \\ \mu \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{V}}_{12}^\top \\ 0 \end{bmatrix} \hat{\boldsymbol{\alpha}} - \begin{bmatrix} \left(\boldsymbol{\Sigma}_{12}^{-1}\right)^\top \boldsymbol{U}^\top \\ 0 \end{bmatrix} \boldsymbol{y}. \tag{20b}$$

Note that the matrices appearing to the left of the equal sign in both (20a) and (20b) are full-row rank: the first is in $\mathbb{R}^{\hat{m}\times\hat{m}}$ and the second is in $\mathbb{R}^{(n-\hat{m}+1)\times(2n-\hat{m}+1)}$. Thus, we can proceed as follows:

(i) we obtain $\begin{bmatrix}\tilde{\boldsymbol{\lambda}}^\top & \mu\end{bmatrix}^\top$ from (20b) as a function of $\hat{\boldsymbol{\alpha}}$ and $\boldsymbol{y}$;
(ii) we introduce it in (20a) and obtain $\hat{\boldsymbol{\alpha}}$ as an affine form of $\boldsymbol{y}$;
(iii) we go back in (20b), replace the now-known $\hat{\boldsymbol{\alpha}}$ and obtain $\begin{bmatrix}\tilde{\boldsymbol{\lambda}}^\top & \mu\end{bmatrix}^\top$ as an affine form of $\boldsymbol{y}$.

For ease of notation, we assume that the steps mentioned above lead to

$$\hat{\boldsymbol{\alpha}}(\boldsymbol{y}) = \hat{\boldsymbol{M}}_\alpha \boldsymbol{y} + \hat{\boldsymbol{N}}_\alpha, \quad \tilde{\boldsymbol{\lambda}}(\boldsymbol{y}) = \tilde{\boldsymbol{M}}_\lambda \boldsymbol{y} + \tilde{\boldsymbol{N}}_\lambda, \quad \mu(\boldsymbol{y}) = \boldsymbol{M}_\mu \boldsymbol{y} + \boldsymbol{N}_\mu. \tag{21}$$

Combining with the fact that $\tilde{\boldsymbol{\alpha}} = \boldsymbol{0}, \hat{\boldsymbol{\lambda}} = \boldsymbol{0}$, as required in (18c)–(18d), allows for writing (by interlacing rows of zeroes in matrices $\hat{\boldsymbol{M}}_\alpha$, $\hat{\boldsymbol{N}}_\alpha$, $\tilde{\boldsymbol{M}}_\lambda$, $\tilde{\boldsymbol{N}}_\lambda$) the explicit solutions for $\boldsymbol{\alpha}, \boldsymbol{\lambda}, \mu$:

$$\boldsymbol{\alpha}(\boldsymbol{y}) = \boldsymbol{M}_\alpha \boldsymbol{y} + \boldsymbol{N}_\alpha, \quad \boldsymbol{\lambda}(\boldsymbol{y}) = \boldsymbol{M}_\lambda \boldsymbol{y} + \boldsymbol{N}_\lambda, \quad \mu(\boldsymbol{y}) = \boldsymbol{M}_\mu \boldsymbol{y} + \boldsymbol{N}_\mu. \tag{22}$$

Recalling that $\tilde{\boldsymbol{\lambda}} > 0$ and $\hat{\boldsymbol{\alpha}} > 0$, as required in (18c)–(18d), and using the notation from (21) gives the feasible domain over which the affine laws (21) describe the optimal solution:

$$\mathcal{R} = \{\boldsymbol{y} : \hat{\boldsymbol{M}}_\alpha \boldsymbol{y} + \hat{\boldsymbol{N}}_\alpha > 0, \tilde{\boldsymbol{M}}_\lambda \boldsymbol{y} + \tilde{\boldsymbol{N}}_\lambda > 0\} = \left\{\boldsymbol{y} : \begin{bmatrix}\hat{\boldsymbol{M}}_\alpha \\ \tilde{\boldsymbol{M}}_\lambda\end{bmatrix} \boldsymbol{y} + \begin{bmatrix}\hat{\boldsymbol{N}}_\alpha \\ \tilde{\boldsymbol{N}}_\lambda\end{bmatrix} > 0\right\}. \tag{23}$$

To recapitulate: starting from an initially chosen collection of active constraints, we have arrived at explicit formulations, parametrized affinely in $\boldsymbol{y}$, for the primal solution $\boldsymbol{\alpha}$, the dual solutions $\boldsymbol{\lambda}, \mu$ as in (22) and the region (23) for which these solutions are optimal. Note that (23) may turn to be infeasible (the empty set). In this case, it simply means that the associated collection of active constraints is not a feasible one and it should be discarded from the enumeration.

Enumerating all feasible combinations of active constraints will [9], thus, partition the parameter domain (the space $\mathbb{R}^m$ or some restriction of it in which $\boldsymbol{y}$ lies) into a non-overlapping collection of regions (23), which is the support of a piecewise law affine in $\boldsymbol{y}$, given by (22):

$$\boldsymbol{\alpha}(\boldsymbol{y}) = \boldsymbol{M}_\alpha^i \boldsymbol{y} + \boldsymbol{N}_\alpha^i, \quad \forall \boldsymbol{y} \in \mathcal{R}^i = \{\boldsymbol{y} : \hat{\boldsymbol{M}}_\alpha^i \boldsymbol{y} + \hat{\boldsymbol{N}}_\alpha^i > 0, \tilde{\boldsymbol{M}}_\lambda^i \boldsymbol{y} + \tilde{\boldsymbol{N}}_\lambda^i > 0\}. \tag{24}$$

Index i appearing in (24) augments the shorthand 'hat' and 'tilde' notation used earlier for compactness reasons (that is, each time we had, e.g., $\tilde{\boldsymbol{V}}_1$ we should had have written $\tilde{\boldsymbol{V}}_1^i$ to highlight that we were considering the i-th combination of active constraints).

Remark 3. *With (24) computed offline, at runtime, we only need to identify the index i for which $\boldsymbol{y} \in \mathcal{R}^i$ holds and apply the corresponding law $\boldsymbol{\alpha}(\boldsymbol{y})$. For large dimensions, this may create storing and retrieval problems (as the number of regions $\mathcal{R}^i$ increases exponentially). Various solutions, with limited success, are proposed in the literature [4,7,14].*

Remark 4. *Due to the continuity of the piecewise law $\boldsymbol{\alpha}(y)$ and the fact that partition $\cup_i \mathcal{R}^i$ covers the entire domain, we do not require the analysis of cases where more than $m+1$ constraints are inactive [12].*

3.4. Illustrative Example

For illustration purposes, we depict a proof-of-concept example. We take $\boldsymbol{T} = \begin{bmatrix}1 & 0 & \sqrt{2} \\ 0 & 1 & \sqrt{2}\end{bmatrix}$, $\boldsymbol{\Delta} = \boldsymbol{I}_3$, $\delta = 1$, thus the the space dimensions are $m = 2$, $n = 3$, $p = 3$. Having taken $(\Delta = \boldsymbol{I}_3, \delta = 1)$ means that $\|\boldsymbol{\Delta} x\|_1 \leq \delta$ describes the three-dimensional, cross-polytope $C_3 = \text{conv}\,(\pm\boldsymbol{e}_1, \pm\boldsymbol{e}_2, \pm\boldsymbol{e}_3)$, as shown (semi-transparent red object) in Figure 2.

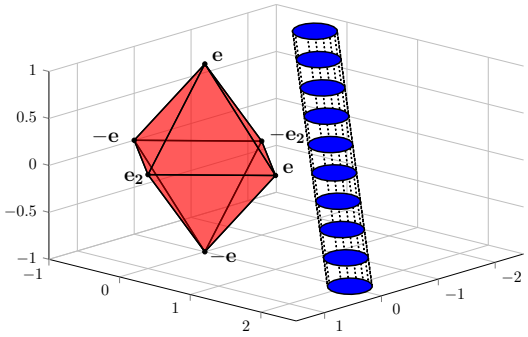
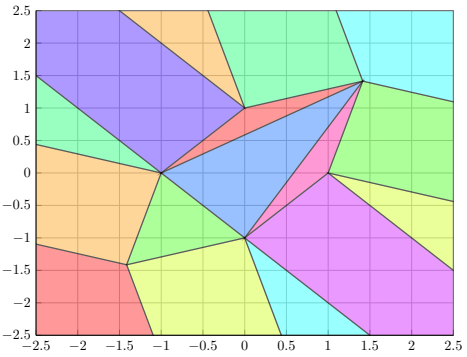


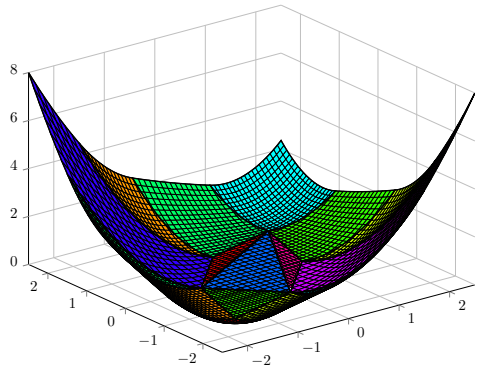
Figure 2. Feasible domain and degenerate cost surface.

The geometrical interpretation of solving (10) is that the unconstrained optimum (the one for which $\boldsymbol{T}\boldsymbol{\xi} - \boldsymbol{y} = 0$) is projected on the feasible domain, thus leading to the constrained optimum (i.e., the solution $\boldsymbol{\xi}^\star$). This is done by inflating the sub-level cost sets (those for which the cost remains constant on their boundary) until there is an intersection with the feasible domain. Since, in the particular case of (10), the quadratic cost depends on a semi-definite matrix ($\boldsymbol{T}^\top \boldsymbol{T}$ is not full-rank); this means that the optimum cost lies in the null subspace of matrix $\boldsymbol{T}^\top T$ and the sub-level sets correspond to a degenerate ellipsoid (the blue-and-dotted cylinder shown in Figure 2). Note that, in (14) and its subsequent variations, we made a change of variable that increases the search space from $\mathbb{R}^3$ to $\mathbb{R}^6$, which is, of course, not representable in 3D.

Continuing with the geometric interpretation: the sub-level surface is inflated and touches the feasible domain at some point on its boundary. As long as this point is described by the same combination of active constraints (this means in Figure 2 that it is described by the same combination of vertices), we have that the solution $\alpha(\boldsymbol{y})$ is given as in (24) for a fixed index i. Moreover, we identify the region $\mathcal{R}^i$ in which $\boldsymbol{y}$ lies. For this numerical example, the 16 regions $\mathcal{R}^i$ obtained through the active set enumeration procedure are shown in Figure 3a and the associated cost value in Figure 3b.



(**a**) parameter domain partition



(**b**) piecewise cost value

Figure 3. Representation of the multi-parametric interpretation.

We have not illustrated the actual solution $\alpha^\star(\boldsymbol{y})$ (due to its dimension). We can however note that, as expected, no more than $m = 2$ vertices (corresponding to an edge in Figure 2) are active for any of the critical regions.

4. Integration of the Multi-Parametric Formulation in the Representation Problem

The sparse representation stage of (4) is a repeated computation of (2), parametrized after signals $\boldsymbol{y} \in \boldsymbol{Y}$. Thus, we may re-interpret (2) as a multi-parametric problem. This means that an explicit representation of the optimal solution can be obtained as a piecewise function, linear in parameter $\boldsymbol{y}$: $\boldsymbol{x}^\star(\boldsymbol{y})$. Thus, the offline part consists in computing the critical regions and their associated laws,

whereas the online part is simply a lookup procedure which identifies and subsequently applies the law associated with the current value of parameter $\boldsymbol{y}$.

In what follows, we proceed to show in Section 4.1 how Section 3 applies to the formulations of Section 2 and to highlight some of the remaining difficulties in Sections 4.2 and 4.3.

4.1. Multi-Parametric Formulations for the Sparse and Cosparse Representations

In what follows, we particularize the results of Section 3 for the sparse and cosparse dictionary representations from Section 2.

For compactness, in Section 3, we have dealt with a somewhat generic constrained optimization problem (particularized in the fact that we start with cross-polytopic constraints and with a rank-deficient quadratic cost). This is justified by Table 1, which shows that both the sparse (2) and the cosparse representation (8) are particular instances of (14). A few details are in order.

Table 1. Equivalences of the sparse and cosparse representations w.r.t. form (14).

Sparse Representation (2)	**Form** (14)	**Cosparse Representation** (8)
$\boldsymbol{x} \in \mathbb{R}^n$	$\boldsymbol{\xi}$	$\boldsymbol{z} \in \mathbb{R}^m$
$\boldsymbol{y} \in \mathbb{R}^m$	$\boldsymbol{y}$	$\boldsymbol{y} \in \mathbb{R}^m$
$\boldsymbol{D} \in \mathbb{R}^{m \times n}$	$\boldsymbol{T}$	$\boldsymbol{I} \in \mathbb{R}^m$
$\boldsymbol{I} \in \mathbb{R}^{n \times n}, \mathbf{1} \in \mathbb{R}^n$	$\boldsymbol{\Delta}, \boldsymbol{\delta}$	$\boldsymbol{\Omega} \in \mathbb{R}^{n \times m}, \mathbf{1} \in \mathbb{R}^n$
$\begin{bmatrix} \boldsymbol{D} & -\boldsymbol{D} \end{bmatrix} \in \mathbb{R}^{m \times 2n}$	$\boldsymbol{TF}$	$\begin{bmatrix} \boldsymbol{\Omega} & -\boldsymbol{\Omega} \end{bmatrix} \in \mathbb{R}^{n \times 2m}$

The sparse representation (2) meshes easily onto (14) with the observation that the constraints describe a canonical cross-polytope (not scaled or projected in a lower-space). Thus, the application of the multi-parametric proceeds directly and requires no further clarification.

The cosparse representation can as well be written in form (14) with some caveats. Foremost, since it is natural to take $m \ll n$, we may consequently assume that $m \leq 2n$. In this case, $\boldsymbol{TF}$ is full column-rank and thus $(\boldsymbol{TF})^\top (\boldsymbol{TF}) \in \mathbb{R}^{2m \times 2m}$ is full-rank, which simplifies the reasoning around (19)–(20). On the other hand, the constraint representation is more challenging. In (8), there appears $\boldsymbol{\Omega}_\Lambda$, which is the restriction of $\boldsymbol{\Omega}$ by the elimination of $m - s$ columns. This is done in (8) through the addition of a rank condition. Within the framework of Section 3, we can instead restrict the enumeration to those sets of active constraints which verify the rank condition.

4.2. The Enumeration Roadblock

The "dirty" secret in multi-parametric programming is that the number of critical regions explodes exponentially with the parameter space size and with the complexity of the feasible domain [9]. Moreover, the number of feasible (non-empty) regions is not usually known a priori [6].

In this particular case, the cross-polytope's specific structure means that we can bound a priori the number of critical regions and enumerate them constructively. To do this, we recall [11] that, for each pair of non-opposite vertices, there is an edge joining them. More generally, each set of $k+1$ orthogonal vertices corresponds to a distinct k-dimensional face that contains them (particular, more familiar instances of the 'k-dimensional face' are the $0, 1, n-1$-dimensional faces—the vertices, edges and, respectively, facets of a polyhedron). The number of k-dimensional faces in an n-dimensional cross-polytope is therefore

$$2^{k+1} \binom{n}{k+1}. \tag{25}$$

As highlighted in Section 3, the constrained optimum $\boldsymbol{\xi}_i^\star = \boldsymbol{F}\boldsymbol{\alpha}_i^\star$ is found by fixing $\hat{m}$ of the cross-polytope's vertices (i.e., those for which $\hat{\boldsymbol{\alpha}} > 0$). Due to the cross-polytope's properties, we know that these vertices describe an $\hat{m} - 1$-dimensional face. With the full-rank condition employed in (18),

$\hat{m} \leq m+1$, and applying the bound from (25), we have that the total number of faces with dimension m or lower is

$$\sum_{\hat{m}=1}^{m+1} 2^{\hat{m}} \binom{n}{\hat{m}}. \tag{26}$$

Each $\hat{m}$-dimensional face corresponds to (a possibly empty) region (23). Thus, the total number of feasible combinations of active constraints is (26).

Unfortunately, depending on the values of m, n, the number of regions may prove to be prohibitively large (both from a computational and from a memory footprint/retrieval viewpoint).

While there are multiple approaches that attempt to alleviate the issue [14], there is no panacea. In what follows, we consider a simple approach: regions $\mathcal{R}^i$ are added to the partition only on an "as-needed" basis. More precisely, $\boldsymbol{y}$ is searched in the list of already-computed regions; if found, the affine law (24) is applied; if not, (20) is solved for the current parameter $\boldsymbol{y}$ in order to identify the optimal solution and the critical region over which it is valid. Once found, these are stored for further use.

A sketch of the approach is shown in Algorithm 2 (for simplicity's sake, only for the sparse representation case is the cosparse case similar and can be deduced easily).

Algorithm 2: Multi-parametric implementation of the sparse problem.

Data: signals set $\boldsymbol{Y} \in \mathbb{R}^{m \times N}$
collection of critical regions $\mathcal{P} = \{\emptyset\} \subset \mathbb{R}^m$
piecewise affine law $(\mathcal{M} = \{\emptyset\}, \setminus = \{\emptyset\}) \subset \mathbb{R}^{n \times m} \times \mathbb{R}^n$

1 **for** $k = 1 : N$ **do**
2 Find index of the region containing the current value of parameter $\boldsymbol{y}$:
3 $$i = \arg \min_{i, \mathcal{R}^i \in \mathcal{P}} \boldsymbol{y} \in \mathcal{R}^i;$$
4 **if** $i = \{\emptyset\}$ **then**
5 solve (20) for the current value of $\boldsymbol{y}$ obtaining the affine law $(\boldsymbol{M}_\alpha, \boldsymbol{N}_\alpha)$ as in (22) and the critical region $\mathcal{R}$, as in (23);
6 update the partition and the affine law collections: $\mathcal{P} = \mathcal{P} \cup \{\mathcal{R}\}$, $\mathcal{M} = \mathcal{M} \cup \{\boldsymbol{M}_\alpha\}$ and $\setminus = \mathcal{N} \cup \{\boldsymbol{N}_\alpha\}$;
7 change the index to point to the last collection element: $i = \#\mathcal{P}$;
8 apply the selected law to the current parameter: $\boldsymbol{\alpha} = \boldsymbol{M}^i_\alpha \boldsymbol{y} + \boldsymbol{N}^i_\alpha$, $\boldsymbol{x} = \boldsymbol{F}\boldsymbol{\alpha}$

Steps 3 and 6 of the algorithm deserve some additional thought. Recall that $\boldsymbol{V}^\top = \begin{bmatrix} \boldsymbol{W}^\top & -\boldsymbol{W}^\top \end{bmatrix}$ has a symmetrical structure. Coupling this with the observations around (25)–(26) shows that, for each combination of $\hat{m}$ active constraints (i.e., those for which $\hat{\boldsymbol{\alpha}} > 0$), there are $2^{\hat{m}}$ permutations in the column selection from $\boldsymbol{V}^\top$ such that the resulted sub-matrices differ only through the signs of their columns. That is, there exist suitably chosen diagonal matrices $\hat{\boldsymbol{P}}, \tilde{\boldsymbol{P}}$ with ± 1 entries such that, e.g., $\tilde{\boldsymbol{V}}_1^\top \mapsto \tilde{\boldsymbol{V}}_1^\top \tilde{\boldsymbol{P}}$, $\hat{\boldsymbol{V}}_{12}^\top \mapsto \hat{\boldsymbol{V}}_1^\top \hat{\boldsymbol{P}}$. Recalling that, for an active set selection $\hat{(\cdot)}, \tilde{(\cdot)}$ there exist solutions $\boldsymbol{\alpha}(\boldsymbol{y}), \boldsymbol{\lambda}(\boldsymbol{y})$, introducing the sign permutations characterized by $\hat{\boldsymbol{P}}, \tilde{\boldsymbol{P}}$ into (20) leads to a similar sign permutation of the solution:

$$\begin{cases} \hat{\boldsymbol{\alpha}}(\boldsymbol{y}) &= \hat{\boldsymbol{M}}_\alpha \boldsymbol{y} + \hat{\boldsymbol{N}}_\alpha, \\ \tilde{\boldsymbol{\lambda}}(\boldsymbol{y}) &= \tilde{\boldsymbol{M}}_\lambda \boldsymbol{y} + \tilde{\boldsymbol{N}}_\lambda, \end{cases} \xrightarrow{\hat{\boldsymbol{P}}, \tilde{\boldsymbol{P}}} \begin{cases} \hat{\boldsymbol{P}}\hat{\boldsymbol{\alpha}}(\boldsymbol{y}) &= \hat{\boldsymbol{M}}_\alpha \boldsymbol{y} + \hat{\boldsymbol{N}}_\alpha, \\ \tilde{\boldsymbol{P}}\tilde{\boldsymbol{\lambda}}(\boldsymbol{y}) &= \tilde{\boldsymbol{M}}_\lambda \boldsymbol{y} + \tilde{\boldsymbol{N}}_\lambda, \end{cases} \xrightarrow{\hat{\boldsymbol{P}}^2 = \boldsymbol{I}, \tilde{\boldsymbol{P}}^2 = \boldsymbol{I}} \begin{cases} \hat{\boldsymbol{\alpha}}(\boldsymbol{y}) &= \hat{\boldsymbol{P}}\hat{\boldsymbol{M}}_\alpha \boldsymbol{y} + \hat{\boldsymbol{P}}\hat{\boldsymbol{N}}_\alpha, \\ \tilde{\boldsymbol{\lambda}}(\boldsymbol{y}) &= \tilde{\boldsymbol{P}}\tilde{\boldsymbol{M}}_\lambda \boldsymbol{y} + \tilde{\boldsymbol{P}}\tilde{\boldsymbol{N}}_\lambda. \end{cases} \tag{27}$$

The last implication exploits the structure of $\hat{\boldsymbol{P}}, \tilde{\boldsymbol{P}}$ by noting that their squared forms are the identity matrices. The meaning of (27) is twofold:

(i) only a single representative of the same family of active constraints has to be stored (Step 6 of Algorithm 2); the remaining $2^{\hat{m}} - 1$ variations may be deduced by multiplying with suitable $\hat{\boldsymbol{P}}, \tilde{\boldsymbol{P}}$;
(ii) the inclusion test (Step 3 of Algorithm 2) has to account for the sign permutations; this can be done, e.g., by checking whether $\left[\hat{\boldsymbol{M}}_{\alpha}\boldsymbol{y} + \hat{\boldsymbol{N}}_{\alpha}\right] \odot \left[\tilde{\boldsymbol{M}}_{\lambda}\boldsymbol{y} + \tilde{\boldsymbol{N}}_{\lambda}\right]_j \geq \boldsymbol{0}$ holds. We denoted with '$\odot$' the elementwise product and with $[\cdot]_j$ the selection of $\hat{m}$ (out of $\tilde{m}$) rows which correspond to the ones appearing in $\hat{\boldsymbol{\alpha}}$. Recall that weights $\boldsymbol{\alpha}$ come in pairs (since they are attached to the unit vectors $\pm\boldsymbol{e}_i$). This means that in a pair of indices $(i, i+n), \forall i \in \{1 \dots n\}$ there can be at most an active constraint (either the i - th or the $i+n$ - th). Thus, whenever we permute the signs we, in fact, switch these indices between the active and inactive sets.

Some remarks are in order.

Remark 5. *Algorithm 2 only provides an alternative for step 2 of Algorithm 1. Still, the same reasoning can be extended to the dictionary learning as it is, a relatively similar optimization problem.*

Remark 6. *Completing online the partition of critical regions (as is done in step 6 of Algorithm 1) may still lead to an exorbitant number of regions (as it may happen that solving for each $\boldsymbol{y}$ requires describing a significant part of the entire partitioning). We expect this to happen when the set of signals $\boldsymbol{Y}$ is chosen randomly. Fortunately, as stated earlier, $\boldsymbol{Y}$ is a collection of corrupted signals coming from a smaller sub-space. We expect thus that the number of regions actually described is much less than the total number of feasible ones.*

Remark 7. *Another approach, not described here, is to merge the explicit and implicit approaches [14]. Instead of enumerating all sets of active constraints for a relatively large parameter dimension, we may choose to consider a reduced number of constraints. Thus, in the offline stage, we compute and store a reasonable number of regions and in the online stage we solve a reduced-size optimization problem (based on information retrieved from the stored lookup tables).*

Remark 8. *While here, we store and search sequentially through the collection of regions (23), and we note that more efficient storing options are possible. A generic solution is to use the Least Recently Used (LRU) algorithm to keep a fixed number of regions while minimizing the chance of a miss during search [19]. Of possible use is the adjacency graph structure in which the cross-polytope's faces are (since the critical regions (23) are in a one-to-one relation with the cross-polytopes's faces).*

Remark 9. *The computational costs of Algorithm 2 can be divided between t_r (retrieval time when searching through the stored critical regions), t_s (time spent solving an online optimization procedure) and t_k (time spent constructing a critical region and its associated affine law). Among these, t_k is negligible as it requires only matrix operations. As long as, on average, enough signals $\mathbf{y}$ are found in the stored regions, the procedure is useful: it suffices that a signal $\mathbf{y}$ is found more often than $\left\lfloor \frac{t_s}{t_r} \right\rfloor$.*

4.3. On the Multi-Parametric Interpretation of the 'Given Approximation Quality' Case

For simplicity, we discuss only the sparse representation case shown in (7). The cosparse case (9) can be treated similarly and is not detailed here.

Recall that, as stated in Section 2, the "true" DL formulations are those shown in (1) and (7) where the ℓ_0 norm appears (either explicitly in the first case or implicitly in the second case). Thus, all of the other formulations appearing in Section 2 are relaxations (exploiting the sparsity induced by the ℓ_1 norm, appearing either in the cost or as a constraint) which aim to simplify the computational effort.

Until here, we have concentrated on these forms where the ℓ_1 norm appears in the constraint part of the optimization problem. Now, we mention some of the drawbacks and possible solutions for the other class of relaxations (those for which the ℓ_1 norm appears in the cost). First, we consider an over-approximation of the quadratic constraint. Any ellipsoid (and therefore the ball $\|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}\|_2^2 \leq \epsilon$)

is a convex set, which by definition means that it can be approximated arbitrarily well by a polyhedral set [18], say, a construction like

$$\{\boldsymbol{x}:\ \boldsymbol{F}(\boldsymbol{D}\boldsymbol{x}-\boldsymbol{y})\leq\theta\}\subset\{\boldsymbol{x}:\ \|\boldsymbol{y}-\boldsymbol{D}\boldsymbol{x}\|_2^2\leq\epsilon\}\subset\{\boldsymbol{x}:\ \boldsymbol{F}(\boldsymbol{D}\boldsymbol{x}-\boldsymbol{y})\leq(1+\epsilon)\theta\}, \tag{28}$$

with F,θ appropriately chosen.

Proceeding as in Section 3, we arrive at the dual (KKT) form

$$\begin{aligned}
\text{stationarity:}\quad & \mathbf{0}\in\partial\left(||\boldsymbol{x}||_1\right)+\nabla(\boldsymbol{F}\boldsymbol{D}\boldsymbol{x}-\boldsymbol{F}\boldsymbol{y}-\theta)\cdot\lambda, && \text{(29a)}\\
\text{primal feasibility:}\quad & \boldsymbol{F}\boldsymbol{D}\boldsymbol{x}-\boldsymbol{F}\boldsymbol{y}-\theta\leq 0, && \text{(29b)}\\
\text{dual feasibility:}\quad & \boldsymbol{\lambda}\geq 0, && \text{(29c)}\\
\text{complementarity:}\quad & \boldsymbol{\lambda}\times(\boldsymbol{F}\boldsymbol{D}\boldsymbol{x}-\boldsymbol{F}\boldsymbol{y}-\theta)=0. && \text{(29d)}
\end{aligned}$$

There are several issues that make (29) challenging to solve as in Section 3:

(i) the gradient is not well defined in inflexion points (where at least one of the vector's components is zero); this requires the use of the 'sub-gradient' notion which is set-valued (29a) becomes '$\mathbf{0}\in\text{sign}(\boldsymbol{x})+(\boldsymbol{F}\boldsymbol{D})^\top\cdot\boldsymbol{\lambda}$', where

$$\text{sign}(x_i)=\begin{cases}1, & x_i>0,\\ -1, & x_i<0,\\ [-1,1], & x_i=0.\end{cases}$$

(ii) the problem may be relatively large (depending on the number of inequalities used to over-approximate the initial quadratic constraint).

Remark 10. *An alternative to item 4.3 is to consider a zonotopic representation of the quadratic constraints. The ellipsoid they describe is in fact a zonoid (a set which can be approximated arbitrarily well by a zonotope [20]). The advantage of this method is that, in generator representation, a zonotope has a much more compact form (which directly leads to a reduced KKT form).*

5. Results

In this section, we present a few numerical simulations on different types of data. We start with synthetic data generated from a random Gaussian distribution such that the signals evenly populate the space in which they lie. Then, we investigate graph signals produced by networks of water distribution, thus studying the impact on data with embedded structure. Finally, we present simulations on images where data items represent small pixel patches. The scripts have been written in Matlab 2018b (MathWorks, Natick, MA, USA), using the MPT3 toolbox [21] and Yalmip [22]. See Supplementary Materials for access to the source code implementation.

5.1. Synthetic Data

In our first experiment, we use synthetic signals. We generate a random dictionary of $m\times n$ elements drawn from a zero mean and unit variance Gaussian distribution. After normalizing the dictionary, we randomly choose s atoms to generate r sparse signals whose coefficients are also picked at random. Our goal is to generate N total signals that are perturbed, dense versions of r clean, sparse signals. We call the r signals seeds and we proceed to perturb each seed N/r times with different additive white Gaussian noise:

$$\boldsymbol{y}_i=\boldsymbol{D}_0\boldsymbol{x}+\boldsymbol{v}_i,\quad i=1:N/r, \tag{30}$$

where $\boldsymbol{D}_0$ is the random dictionary, $\boldsymbol{x}$ the sparse seed signal and $\boldsymbol{v}_i$ the additive noise. Signal $\boldsymbol{y}_i$ is thus a dense signal; all its elements are non-zero. We repeat (30) r times to generate N synthetic signals that we store in $\boldsymbol{Y}$, which is thus known to have a sparse structure.

In the following we generate $n = 1500$ signals, of $m = 20$ elements each, from $r = \{15, 150, 1500\}$ seeds built as the linear combination of $s = \{3, 4, 5\}$ vectors from a dictionary with $n = 50$ atoms.

We proceed with the construction shown in Algorithm 2 where the critical regions and associated optimal laws are computed as-needed. As stated earlier, this is justified by the excessive number of potentially feasible sets of active constraints (26).

Note that we use the real dictionary $\boldsymbol{D}_0$. For a full analysis (not the object of this paper), we may repeat the procedure for all the intermediate dictionaries obtained along the dictionary learning procedure (step 4 in Algorithm 1).

In Figures 4 and 5, we depict the plots in red, blue, green to denote sparsity (3, 4 and 5 respectively) and add diamond, triangle and square markers to denote the seed size (15, 150 and 150, respectively). Thus, a blue with diamond-shaped marker denotes the case $s = 4, r = 15$.

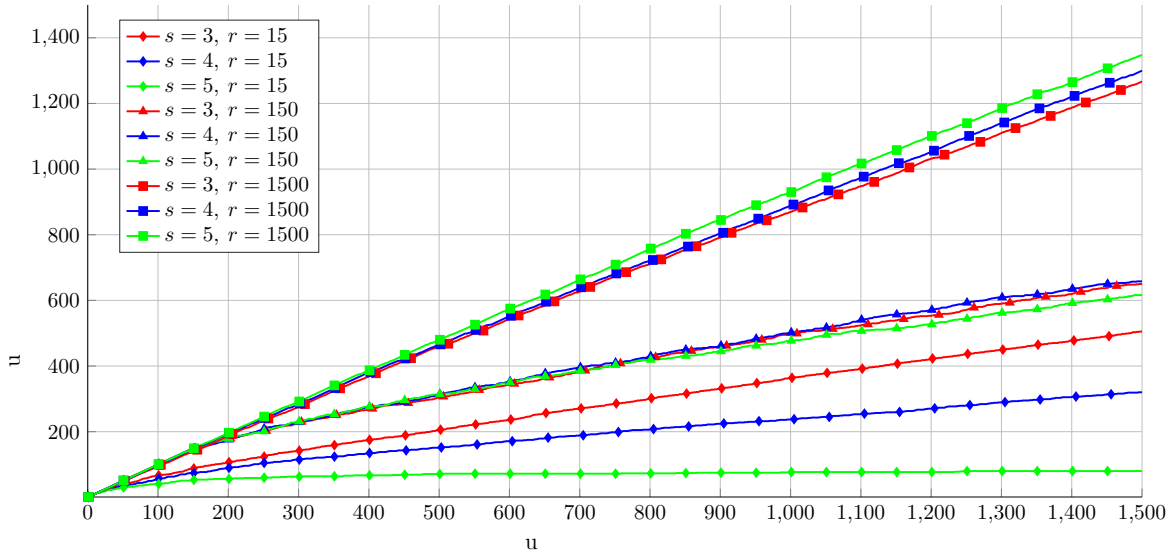


Figure 4. Number of regions versus number of iterations.

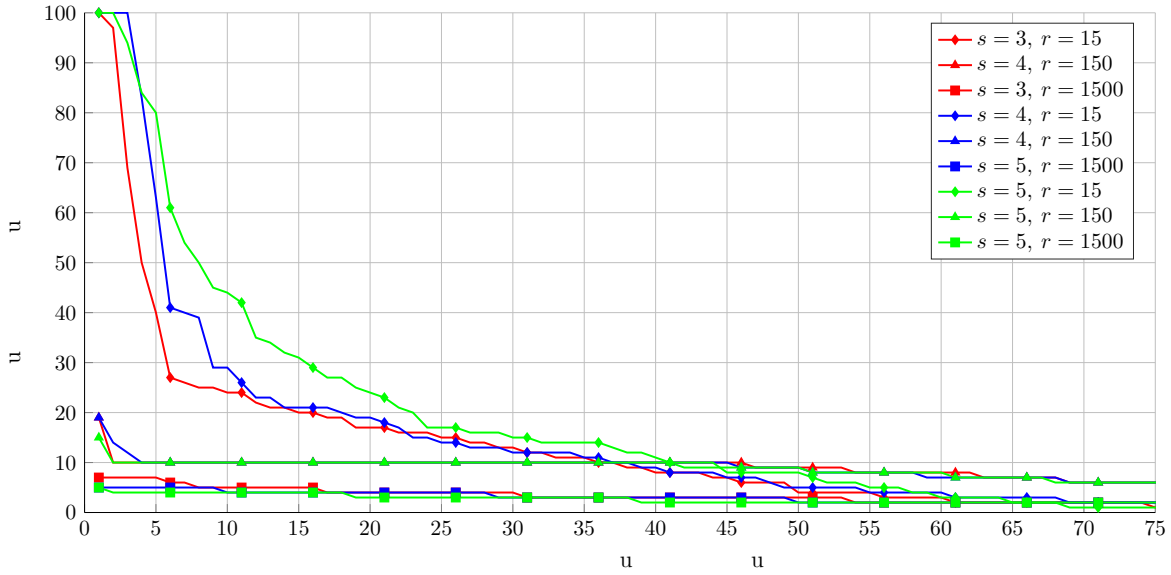


Figure 5. Number of occurrences for a region (sorted in descending order, clipped at the 75th region).

Figure 4 shows that the number of regions (23) is significantly lower than the number of iterations of the parameter vector $\boldsymbol{y}$ for all combinations of parameters s (sparsity) and r (seed).

As expected, the most relevant factor in the synthetic examples is the "randomness" of the data: the best cases are those for which $n = 1500$ vectors $\boldsymbol{y}$ are the result of only $r = 15$ seeds; still, even in the worst case, where each $\boldsymbol{y}$ is associated with a single seed ($n = r = 1500$), we have a significant difference between the number of regions and of iterations.

Figure 5 details the distribution of the inclusion occurrences (i.e., how many vector instances $\boldsymbol{y}$ stay in a certain critical region). Beyond the expected fact that to the cases with lower randomness correspond more inclusions, we also note that, in general, the plots have a steep descent. Table 2 shows the number of regions as a percentage of the total number of signals.

Table 2. Number of regions as a percentage of the total number of signals ([%]).

Sparsity/Seed Size	$r = 15$	$r = 150$	$r = 1500$
$s = 3$	33.67	43.33	84.40
$s = 4$	21.33	43.87	86.67
$s = 5$	5.33	41.13	89.87

Out of the many stored regions (23), only relatively few contain multiple vectors $\boldsymbol{y}$. This is of interest in cases of limited storage (see Remark 8).

The computation times show empirically the effect of having a large number of stored regions. While the signal shown in Figure 6 is noisy, a clear ascending trend is visible (at each iteration step, there are more regions to be searched—step 3 of Algorithm 2).

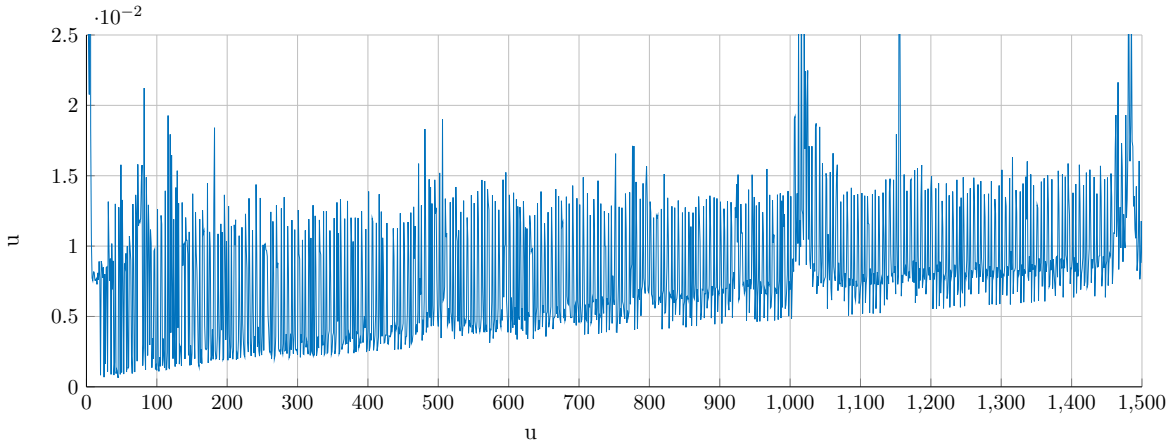


Figure 6. Evolution of the computation time versus the number of iterations (case $s = 3$, $r = 15$).

Disregarding memory considerations, the continued storage of critical regions makes sense until the point where searching through the existing regions takes longer than solving the online optimization problem.

5.2. Water Networks

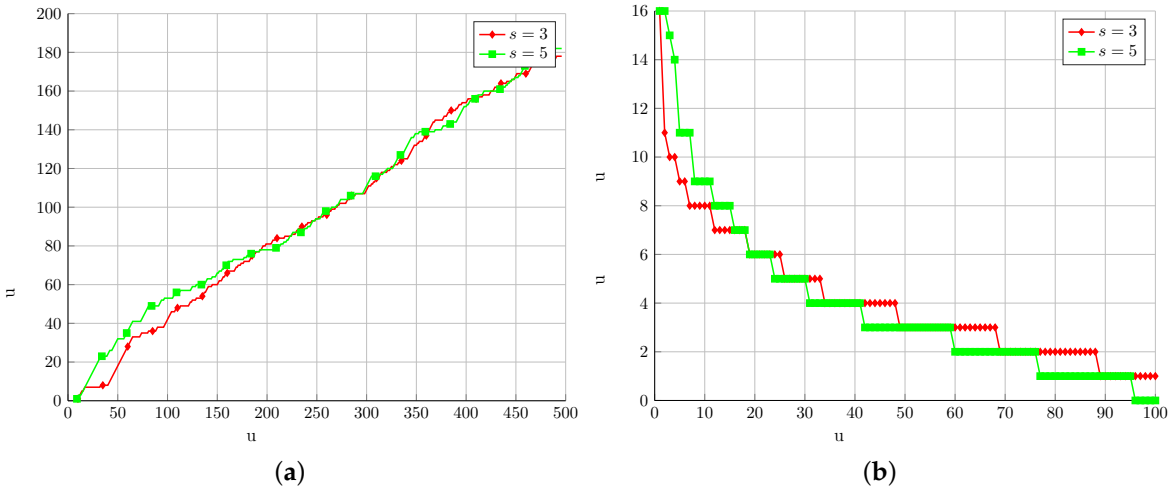
Dictionary learning has only recently been applied for modelling water networks, showing promising results at fault detection and isolation (FDI) tasks [23]. Represented as a graph, where the edges are pipes and the nodes' distribution junctions, the water network main characteristic for FDI is the recorded pressure at each of its nodes. Given a known nominal pressure vector $\boldsymbol{p}_0$, whose elements represent the nominal pressure in each junction, we compare it to the current measurement $\boldsymbol{p}$ leading to the pressure residue

$$\boldsymbol{r} = \boldsymbol{p} - \boldsymbol{p}_0. \tag{31}$$

Thus, faults are detected by the network nodes by investigating the residue vectors over time.

In our experiments, we choose the benchmark Hanoi water network that consists of 31 junctions. We iterate through each of the nodes and produce faults of various magnitudes. For each scenario, we record the pressure residue (31) and store all of them as columns in the matrix $\boldsymbol{R}$. A single fault is expected to have impact on the measured pressured of its node and its first and second level neighbours. See [23] for a thorough description of the water network and the data generation process.

We generate 16 faults for each node leading to $n = 496$ training signals of $m = 31$ entries each. We perform DL on a dictionary of $n = 124$ atoms with sparsity $s = \{3, 5\}$ for 50 iterations and depict in Figure 7 the last sparse representation stage during training. In Figure 7a, we can see that the number of regions is around 36% of the total number of signals, which is confirmed by the number of region re-utilization depicted in Figure 7b.



(**a**) (**b**)

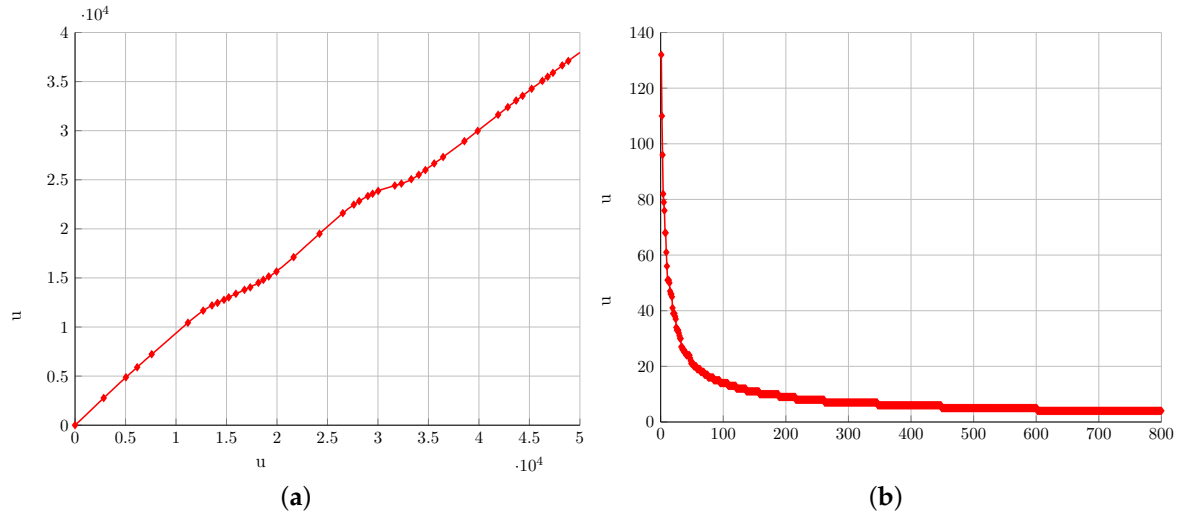
Figure 7. Illustration for the Hanoi water network. (**a**) Number of regions versus number of iterations; (**b**) Number of occurrences for a region (sorted in descending order, clipped at the 100-th region).

5.3. Images

For the last experiment, we train our dictionary on a small set of patches extracted from a given image. We use square patches of 3×3 pixels that we vectorize leading to signals $\boldsymbol{y}$ of $m = 9$ elements. It is good practice to use overlapping patches for both training and representation with the learned dictionary afterwards. For example, using overlapping patches when performing image denoising provides better results and eliminates the blocking effect.

In our simulations, we use overlapping patches with a sliding step of one pixel that we extract from the image of Lena [24]. Using $n = 3000$ patches, we train a dictionary of $n = 36$ atoms with sparsity $s = 3$ for 50 iterations. With the resulting dictionary, we proceed to represent $N_t = 50,000$ test patches from Lena that were not used in the DL process.

The results are depicted in Figure 8. We again observe that there are critical regions re-used (the number of regions is approximately 74% of the total number of signals) but we also note that the problem's size leads to relatively large storage requirements. The large problem size (at least when compared with the previous examples) hinders the critical region generation (due badly conditioned matrices) and highlights that a large number of regions need to be stored until a significant number of inclusions occur.



(**a**) (**b**)

Figure 8. Illustration for the Lena benchmark. (**a**) Number of regions versus number of iterations; (**b**) Number of occurrences for a region (sorted in descending order, clipped at the 800th region).

6. Conclusions and Future Directions

In this article, we proposed a novel sparse representation method for both the spare and cosparse models by looking at the problem as a multi-parametric formulation and solving it through the Karush–Kuhn–Tucker conditions. Thus, we solve a large optimization problem once in the beginning, and then, for each signal, we simply enumerate through a set of possible regions where the solution might lie. The advantage of this approach is the significant reduction in sparse representation execution times: if the trained dictionary is distributed together with the associated collection of regions, representation reduces to performing a few matrix-vector computations.

Our model is a good fit for online representation scenarios which is why we plan on investigating in the future its adaptation to online dictionary learning [25], where the dictionary is also updated with each new signal that is sparsely represented. Existing methods have shown that the dictionary update problem can be reduced to a simple rank-1 update $\boldsymbol{D} \leftarrow \boldsymbol{D} + \boldsymbol{\Gamma}$ ([15], Chapter 5) and we are currently working on changing $\boldsymbol{\Gamma}$ in order to exploit the existing multi-parametric model of the existing dictionary.

Further work will consider efficient storage and retrieval of the active critical region, e.g., by exploiting the symmetry inherent in the cross-polytope face lattice. This will alleviate issues pertaining to memory limitations and search through the list of stored regions (as shown in (27), a single representative suffices in characterizing an entire family of critical regions and affine laws).

Supplementary Materials: Supplementary Materials are available at https://github.com/pirofti/mp-dl.

Author Contributions: Conceptualization, F.S. and P.I.; methodology, F.S.; software, P.I.; validation, F.S.; formal analysis, F.S. and P.I; investigation, F.S. and P.I.; resources, P.I.; data curation, P.I.; writing—original draft preparation, P.I.; writing—review and editing, F.S. and P.I.

Funding: Florin Stoican was supported by the Politehnica University of Bucharest, through the internal research grant GNaC 2918 ARUT, contract no. 4/15.10.2018. Paul Irofti was supported by a grant of the Romanian Ministry of Research and Innovation, CCCDI-UEFISCDI, project number 17PCCDI/2018 within PNCDI III.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Elad, M. *Sparse and Redundant Representations: From Theory To Applications in Signal and Image Processing*; Springer Science & Business Media: Berlin, Germany, 2010.
2. Fletcher, R. *Practical Methods of Optimization*; John Wiley & Sons: Hoboken, NJ, USA, 2013.
3. Li, Z.; Ierapetritou, M.G. A method for solving the general parametric linear complementarity problem. *Ann. Oper. Res.* **2010**, *181*, 485–501. [CrossRef]

4. Mönnigmann, M.; Jost, M. Vertex based calculation of explicit MPC laws. In Proceedings of the 2012 American Control Conference (ACC), Montreal, QC, Canada, 27–29 June 2012; pp. 423–428.
5. Herceg, M.; Jones, C.N.; Kvasnica, M.; Morari, M. Enumeration-based approach to solving parametric linear complementarity problems. *Automatica* **2015**, *62*, 243–248. [CrossRef]
6. TøNdel, P.; Johansen, T.A.; Bemporad, A. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica* **2003**, *39*, 489–497. [CrossRef]
7. Tøndel, P.; Johansen, T.A.; Bemporad, A. Evaluation of piecewise affine control via binary search tree. *Automatica* **2003**, *39*, 945–950. [CrossRef]
8. Bemporad, A. A multiparametric quadratic programming algorithm with polyhedral computations based on nonnegative least squares. *IEEE Trans. Autom. Control* **2015**, *60*, 2892–2903. [CrossRef]
9. Alessio, A.; Bemporad, A. A survey on explicit model predictive control. In *Nonlinear Model Predictive Control*; Springer: Berlin, Germany, 2009; pp. 345–369.
10. Jones, C.N.; Morrari, M. Multiparametric linear complementarity problems. In Proceedings of the 45th IEEE Conference on Decision and Control, San Diego, CA, USA, 3–15 December 2006; pp. 5687–5692.
11. Coxeter, H.S.M. *Regular Polytopes*; Courier Corporation: Chelmsford, MA, USA, 1973.
12. Ahmadi-Moshkenani, P.; Johansen, T.A.; Olaru, S. On degeneracy in exploration of combinatorial tree in multi-parametric quadratic programming. In Proceedings of the 2016 IEEE 55th Conference on Decision and Control (CDC), Las Vegas, NA, USA, 12–14 December 2016; pp. 2320–2326.
13. Henk, M.; Richter-Gebert, J.; Ziegler, G.M. 16 basic properties of convex polytopes. In *Handbook of Discrete and Computational Geometry*; CRC Press: Boca Raton, FL, USA, 2004; pp. 255–382.
14. Jost, M. Accelerating the Calculation of Model Predictive Control Laws for Constrained Linear Systems. 2016. Available online: https://hss-opus.ub.ruhr-uni-bochum.de/opus4/frontdoor/index/index/docId/4568 (accessed on 27 June 2019).
15. Dumitrescu, B.; Irofti, P. *Dictionary Learning Algorithms and Applications*; Springer: Berlin, Germany, 2018; pp. XIV, 284.
16. Tibshirani, R. Regression Shrinkage and Selection via the Lasso. *J. R. Statist. Soc. B* **1996**, *58*, 267–288. [CrossRef]
17. Nam, S.; Davies, M.; Elad, M.; Gribonval, R. The cosparse analysis model and algorithms. *Appl. Comput. Harmon. Anal.* **2013**, *34*, 30–56. [CrossRef]
18. Ziegler, G.M. *Lectures on Polytopes*; Springer Science & Business Media: Berlin, Germany, 2012; Volume 152.
19. O'neil, E.J.; O'neil, P.E.; Weikum, G. The LRU-K page replacement algorithm for database disk buffering. *ACM Sigmod Rec.* **1993**, *22*, 297–306. [CrossRef]
20. Bourgain, J.; Lindenstrauss, J.; Milman, V. Approximation of zonoids by zonotopes. *Acta Math.* **1989**, *162*, 73–141. [CrossRef]
21. Herceg, M.; Kvasnica, M.; Jones, C.; Morari, M. Multi-Parametric Toolbox 3.0. In Proceedings of the European Control Conference, Zurich, Switzerland, 17–19 July 2013; pp. 502–510.
22. Löfberg, J. YALMIP: A Toolbox for Modeling and Optimization in MATLAB. In Proceedings of the CACSD Conference, Taipei, Taiwan, 2–4 September 2004.
23. Irofti, P.; Stoican, F. Dictionary Learning Strategies for Sensor Placement and Leakage Isolation in Water Networks. In Proceedings of the 20th World Congress of the International Federation of Automatic Control, Toulouse, France, 9–14 July 2017; pp. 1589–1594.
24. Weber, A. The USC-SIPI Image Database. 1997. Available online: http://sipi.usc.edu/database/ (accessed on 27 June 2019).
25. Skretting, K.; Engan, K. Recursive least squares dictionary learning. *IEEE Trans. Signal Proc.* **2010**, *58*, 2121–2130. [CrossRef]