The Sparse Principal Component of a Constant-rank Matrix

Megasthenis Asteris, Dimitris S. Papailiopoulos, and George N. Karystinos

Abstract

The computation of the sparse principal component of a matrix is equivalent to the identification of its submatrix with the largest maximum eigenvalue. Finding this optimal submatrix is what renders the problem $\mathcal{NP}$-hard. In this work, we prove that, if the matrix is positive semidefinite and its rank is constant, then its sparse principal component is polynomially computable. Our proof utilizes the auxiliary unit vector technique that has been recently developed to identify problems that are polynomially solvable. Moreover, we use this technique to design an algorithm which, for any sparsity value, computes the sparse principal component with complexity $\mathcal{O}\left(N^{D+1}\right)$, where N and D are the matrix size and rank, respectively. Our algorithm is fully parallelizable and memory efficient.

Index Terms

Eigenvalues and eigenfunctions, feature extraction, information processing, machine learning algorithms, principal component analysis, signal processing algorithms.

I. Introduction

Principal component analysis (PCA) is a well studied and broadly used dimensionality reduction tool. The principal components (PCs) of a set of observations on N variables capture orthogonal directions of maximum variance and offer a distance-optimal, low-dimensional representation that -for many purposes- conveys sufficient amount of information. Without additional constraints, the PCs of a data set can be computed in polynomial time in N using the eigenvalue decomposition.

A disadvantage of conventional PCA is that, in general, the extracted eigenvectors are expected to have nonzero elements in all their entries. In many applications, sparse vectors that convey information are more favorable either due to sparsity of the actual signals [1], [2] or because sparsity endows interpretability [3] when each coordinate of a PC corresponds, for example, to different words in text analysis applications or the expression of a particular

This paper was presented at the 2011 IEEE International Symposium on Information Theory (ISIT 2011), Saint Petersburg, Russia, Aug. 2011.

M. Asteris and D. S. Papailiopoulos are with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712, USA (e-mail: dimitris@utexas.edu).

G. N. Karystinos (corresponding author) is with the Department of Electronic and Computer Engineering, Technical University of Crete, Chania, 73100, Greece (e-mail: karystinos@telecom.tuc.gr).

gene in bio data sets. Thus, provided that the application requires it, some of the maximum variance of the true PCs may be traded for sparsity. Recently, there has been an increasing interest in computing sparse components of data sets with applications that range from signal processing, communication networks, and machine learning, to bioinformatics, finance, and meteorology [4]-[13].

To enforce sparsity on the exacted eigenvectors, a linearly constrained l_0-norm minimization problem is usually considered [1], [2], [14], [15]. This problem is equivalent to the sparse variance maximization, that is, the maximization of the Rayleigh quotient of a matrix under an l_0-norm constraint on the maximizing argument [4]-[6], [8], [11], [13], [16]-[23]. In both problems, due to the additional cardinality constraint that is enforced, the sparsity-aware flavor of PCA, termed sparse PCA, comes at a higher cost: sparse PCA is an $\mathcal{NP}$-hard problem [16].

To approximate sparse PCA, various methods have been introduced in the literature. A modified PCA technique based on the LASSO was introduced in [24]. In [3], a nonconvex regression-type optimization approach combined with LASSO penalty was used to approximately tackle the problem. A nonconvex technique, locally solving difference-of-convex-functions programs, was presented in [17]. Semidefinite programming (SDP) was used in [5], [22], while [18] augmented the SDP approach with an extra greedy step that offers favorable optimality guarantees under certain sufficient conditions. The authors of [4] considered greedy and branch-and-bound approaches. Generalized power method techniques using convex programs were also used to approximately solve sparse PCA [20]. A sparse-adjusted deflation procedure was introduced in [19] and in [6] optimality guarantees were shown for specific types of covariance matrices under thresholding and SDP relaxations. Iterative thresholding was also considered in [25] in conjunction with certain guarantees while a truncated power method was presented in [26].

In this present work, we prove that the sparse principal component of an $N \times N$ matrix $\mathbf{C}$ can be obtained in polynomial time under a new sufficient condition: when $\mathbf{C}$ can be written as a sum of a scaled identity matrix and a positive semidefinite update, i.e., $\mathbf{C} = \sigma \mathbf{I}_N + \mathbf{A}$, and the rank D of the update $\mathbf{A}$ is not a function of the problem size.[1] Under this condition, we show that sparse PCA is solvable with complexity $\mathcal{O}\left(N^{D+1}\right)$. Our proof utilizes the auxiliary unit vector technique that we developed in [27], [28]. This technique has been inspired by the work in [29], which reappeared in [30] and was used in [31]. It introduces an auxiliary unit vector that unlocks the constant-rank structure of a matrix (in this present work, matrix $\mathbf{A}$). The constant-rank property along with the auxiliary vector enable us to scan a constant-dimensional space and identify a polynomial number of candidate vectors (i.e., candidate solutions to the original problem). Interestingly, the optimal solution always lies among these candidates and a polynomial time search can always retrieve it. As a result, we have applied the auxiliary unit vector technique to identify the polynomial solvability of certain optimization problems and provide polynomial-time algorithms that are directly implementable, fully parallelizable, and memory efficient [32]-[34].

The rest of this paper is organized as follows. In Section II, we state the sparse PCA problem and indicate its $\mathcal{NP}$-hardness. Then, in Section III, we follow the principles of the auxiliary unit vector technique to present a proof of the polynomial solvability of the sparse PCA problem under a constant-rank condition. Moreover, we

[1]If $\sigma = 0$, then we simply have a constant-rank matrix $\mathbf{C}$.

design a novel algorithm[2] which, for any sparsity value, computes the sparse principal component with complexity $\mathcal{O}\left(N^{D+1}\right)$. Our algorithm is simply implementable, fully parallelizable, and memory efficient. Especially for the case $D = 2$, an alternative nonparallelizable version of our algorithm with complexity $\mathcal{O}\left(N^2 \log N\right)$ is presented in Section IV. A few conclusions are drawn in Section V.

II. Problem Statement

We are interested in the computation of the real, unit-norm, and at most K-sparse principal component of the $N \times N$ nonnegative definite matrix $\mathbf{C}$, i.e.,

$$\mathbf{x}_{\text{opt}} \overset{\triangle}{=} \underset{\substack{\mathbf{x}\in\mathbb{R}^N \\ \|\mathbf{x}\|=1, \|\mathbf{x}\|_0 \leq K}}{\arg\max} \left\{\mathbf{x}^T \mathbf{C} \mathbf{x}\right\}. \tag{1}$$

Interestingly, when $\mathbf{C}$ can be decomposed as a constant-rank update of the identity matrix, i.e.,

$$\mathbf{C} = \sigma \mathbf{I}_N + \mathbf{A} \tag{2}$$

where $\sigma \in \mathbb{R}$, $\mathbf{I}_N$ is the $N \times N$ identity matrix, and $\mathbf{A}$ is a nonnegative definite matrix with rank D, then the optimization (1) can always be rewritten as

$$\mathbf{x}_{\text{opt}} = \underset{\substack{\mathbf{x}\in\mathbb{R}^N \\ \|\mathbf{x}\|=1, \|\mathbf{x}\|_0 \leq K}}{\arg\max} \left\{\mathbf{x}^T \left(\sigma \mathbf{I}_N + \mathbf{A}\right) \mathbf{x}\right\} = \underset{\substack{\mathbf{x}\in\mathbb{R}^N \\ \|\mathbf{x}\|=1, \|\mathbf{x}\|_0 \leq K}}{\arg\max} \left\{\mathbf{x}^T \mathbf{A} \mathbf{x}\right\}. \tag{3}$$

Since $\mathbf{A}$ is nonnegative and has rank D, it can be decomposed as

$$\mathbf{A} = \mathbf{V}\mathbf{V}^T, \tag{4}$$

where

$$\mathbf{V} \overset{\triangle}{=} [\mathbf{v}_1 \ \ \mathbf{v}_2 \ \ \cdots \ \ \mathbf{v}_D] \tag{5}$$

is an $N \times D$ matrix, and problem (1) can be written as

$$\mathbf{x}_{\text{opt}} = \underset{\substack{\mathbf{x}\in\mathbb{R}^N \\ \|\mathbf{x}\|=1, \|\mathbf{x}\|_0 \leq K}}{\arg\max} \left\{\mathbf{x}^T \mathbf{V}\mathbf{V}^T \mathbf{x}\right\} = \underset{\substack{\mathbf{x}\in\mathbb{R}^N \\ \|\mathbf{x}\|=1, \|\mathbf{x}\|_0 \leq K}}{\arg\max} \left\|\mathbf{V}^T \mathbf{x}\right\|. \tag{6}$$

For the optimization problem in (6), we note that

$$\max_{\substack{\mathbf{x}\in\mathbb{R}^N \\ \|\mathbf{x}\|=1, \|\mathbf{x}\|_0 \leq K}} \left\|\mathbf{V}^T \mathbf{x}\right\| = \max_{\substack{\mathcal{I}\subset[N] \\ |\mathcal{I}|=K}} \max_{\substack{\mathbf{x}\in\mathbb{R}^K \\ \|\mathbf{x}\|=1}} \left\|\mathbf{V}_{\mathcal{I},:}^T \mathbf{x}\right\| \tag{7}$$

where $[N] \overset{\triangle}{=} \{1, 2, \ldots, N\}$. In (7), set $\mathcal{I} \subset [N]$ (which we call the *support*) consists of the indices of the K potentially nonzero elements of $\mathbf{x} \in \mathbb{R}^N$. For a given support $\mathcal{I}$, the inner maximization is a K-dimensional principal-component problem, where $\mathbf{V}_{\mathcal{I},:}$ is the corresponding $K \times D$ submatrix of $\mathbf{V}$. The solution to the inner maximization is denoted by

$$\mathbf{x}(\mathcal{I}) \overset{\triangle}{=} \underset{\substack{\mathbf{x}\in\mathbb{R}^K \\ \|\mathbf{x}\|=1}}{\arg\max} \left\|\mathbf{V}_{\mathcal{I},:}^T \mathbf{x}\right\|. \tag{8}$$

[2]Early versions of our algorithm appeared in [35], [36].

and equals the principal left singular vector of $\mathbf{V}_{\mathcal{I},:}$. Then, our optimization problem in (7) becomes

$$\mathcal{I}_{\text{opt}} \overset{\triangle}{=} \arg\max_{\substack{\mathcal{I}\subset[N] \\ |\mathcal{I}|=K}} \{\sigma_{\max}(\mathbf{V}_{\mathcal{I},:})\} \tag{9}$$

where $\sigma_{\max}(\mathbf{V})$ denotes the principal singular value of matrix $\mathbf{V}$. That is, to solve our original problem in (1), according to (9), we need to find the K-row submatrix of $\mathbf{V}$ whose principal singular value is the maximum one among all submatrices. The indices that are contained in the optimal support $\mathcal{I}_{\text{opt}}$ that solves (9) correspond to the nonzero loadings of the solution $\mathbf{x}_{\text{opt}}$ to (1). Then, according to (8), the values of these nonzero loadings are directly computed by the left singular vector of $\mathbf{V}_{\mathcal{I}_{\text{opt}},:}$.

From the above discussion, it turns out that the hardness of the original problem in (1) comes from the identification of the optimal support $\mathcal{I}_{\text{opt}}$ in (9). To obtain the optimal support $\mathcal{I}_{\text{opt}}$, we could simply perform an exhaustive search among all $\binom{N}{K}$ possible supports $\mathcal{I}$ and compare them against the metric of interest in (9). However, if K is not constant but grows with N, then such an approach has complexity that is exponential in N, indicating the $\mathcal{NP}$-hardness of (1), which was shown in [16]. In this present work, we show that, if the rank D of $\mathbf{A}$ is constant, then (9) can be solved in time polynomial in N. In fact, we develop an algorithm that has complexity $\mathcal{O}\left(N^{D+1}\right)$ and returns $\mathcal{O}\left(N^{D}\right)$ candidate supports, one of which is guaranteed to be the solution to (9). Then, by an exhaustive search among only these candidate supports, we identify the optimal support in (9) and, hence, the sparse principal component of $\mathbf{A}$ and $\mathbf{C}$ with complexity polynomial in N, for any sparsity value K between 1 and N (that is, even if K grows with N).

III. Computation of the Sparse Principal Component in Time $\mathcal{O}\left(N^{D+1}\right)$

Prior to presenting the main result for the general rank-D case, in the following subsection we provide insights as to why the sparse principal component of constant-rank matrices can be solved in polynomial time by first considering the trivial case $D = 1$.

A. Rank-1: A motivating example

In this case, $\mathbf{A}$ has rank 1 and $\mathbf{V} = \mathbf{v}$. For a given support $\mathcal{I}$, we have $\mathbf{V}_{\mathcal{I},:} = \mathbf{v}_{\mathcal{I}}$. Then, our optimization problem in (7) becomes

$$\max_{\substack{\mathcal{I}\subset[N] \\ |\mathcal{I}|=K}} \max_{\substack{\mathbf{x}\in\mathbb{R}^K \\ \|\mathbf{x}\|=1}} \left|\mathbf{v}_{\mathcal{I}}^T\mathbf{x}\right| \tag{10}$$

where, for any given support $\mathcal{I}$, the corresponding vector in (8) is

$$\mathbf{x}(\mathcal{I}) = \arg\max_{\substack{\mathbf{x}\in\mathbb{R}^K \\ \|\mathbf{x}\|=1}} \left|\mathbf{v}_{\mathcal{I}}^T\mathbf{x}\right| = \frac{\mathbf{v}_{\mathcal{I}}}{\|\mathbf{v}_{\mathcal{I}}\|}. \tag{11}$$

Therefore, (7) becomes

$$\max_{\substack{\mathcal{I}\subset[N] \\ |\mathcal{I}|=K}} \left|\mathbf{v}_{\mathcal{I}}^T\frac{\mathbf{v}_{\mathcal{I}}}{\|\mathbf{v}_{\mathcal{I}}\|}\right| = \max_{\substack{\mathcal{I}\subset[N] \\ |\mathcal{I}|=K}} \|\mathbf{v}_{\mathcal{I}}\| \tag{12}$$

and the optimal support is

$$\mathcal{I}_{\text{opt}} = \underset{\substack{\mathcal{I} \subset [N] \\ |\mathcal{I}| = K}}{\arg\max} \|\mathbf{v}_{\mathcal{I}}\| = \underset{\substack{\mathcal{I} \subset [N] \\ |\mathcal{I}| = K}}{\arg\max} \sum_{n \in \mathcal{I}} v_n^2. \tag{13}$$

That is, to determine the solution to (9), we only need to compare the elements of $|\mathbf{v}|$ and select the K largest ones. Then, their indices are the elements of $\mathcal{I}_{\text{opt}}$.

The above observation, although simple, turns out to be critical for the developments that follow. Hence, to simplify the presentation, we define function *top*$_k$ which is parameterized in an integer k, has input a vector $\mathbf{u}$ of length $N \geq k$, and returns the indices of the k largest values in $|\mathbf{u}|$:

$$\text{top}_k(\mathbf{u}) \overset{\triangle}{=} \underset{\substack{\mathcal{I} \subset [N] \\ |\mathcal{I}| = k}}{\arg\max} \|\mathbf{u}_{\mathcal{I}}\|. \tag{14}$$

Function $\text{top}_k(\mathbf{u})$ operates by selecting the indices of the k largest values among $|u_1|$, $|u_2|$, $\ldots$, $|u_N|$. Its complexity is $\mathcal{O}(N)$ [37].

We conclude this subsection by noting that, if $D = 1$, then the optimal support in (9) is

$$\mathcal{I}_{\text{opt}} = \text{top}_K(\mathbf{v}) \tag{15}$$

and is computed with linear complexity.

B. Rank-D: Utilizing the auxiliary unit vector technique

We consider now the case where $\mathbf{A}$ has rank $D \geq 1$ and, hence, $\mathbf{V}$ is an $N \times D$ matrix. Without loss of generality (w.l.o.g.), we assume that each row of $\mathbf{V}$ has at least one nonzero element, i.e., $\mathbf{V}_{n,1:2} \neq \mathbf{0}_{1\times 2}$. Indeed, as explained in [28], if there exists an index $n \in [N]$ such that $\mathbf{V}_{n,:} = \mathbf{0}$, then, independently of the value of the corresponding element x_n of $\mathbf{x}$, the contribution of this row to the value of $\left\|\mathbf{V}^T\mathbf{x}\right\|$ in (6) will be zero. Hence, there is no point in "spending" in x_n a weight that could be distributed to other elements of $\mathbf{x}$; we can ignore the nth row of $\mathbf{V}$, replace $\mathbf{V}$ by $\mathbf{V}_{[N]-\{n\},:}$, and, hence, reduce the problem size from N to $N-1$. In the final solution $\mathbf{x}_{\text{opt}}$, x_n will be set to zero.

In our subsequent developments, we rely on the auxiliary unit vector technique that was introduced in [27] for matrices of rank $D = 2$ and generalized in [28] for matrices of rank $D \geq 2$. This technique utilizes an auxiliary vector $\mathbf{c}$ to generate the subspace spanned by the columns of $\mathbf{V}$ and result in a rank-1 problem for each value of $\mathbf{c}$. Interestingly, for several problems, the number of different solutions that we obtain as $\mathbf{c}$ scans the unit-radius hypersphere has polynomial size. If the rank-1 problem for each value of $\mathbf{c}$ is polynomially solvable (as, for example, in the optimization problem of this work, as indicated in Subsection III-A), then the optimal solution is obtained with overall polynomial complexity. In a few words, the auxiliary unit vector technique of [27], [28] is a fully parallelizable and memory efficient technique that translates D-dimensional problems into a polynomial collection of rank-1 problems among which one results in the overall optimal solution.

For our sparse-principal-component problem, the auxiliary unit vector technique works as follows. Consider a unit vector $\mathbf{c} \in \mathbb{R}^D$ and Cauchy-Schwartz Inequality, according to which, for any $\mathbf{a} \in \mathbb{R}^D$,

$$\left|\mathbf{a}^T\mathbf{c}\right| \leq \|\mathbf{a}\| \, \|\mathbf{c}\| = \|\mathbf{a}\| \tag{16}$$

with equality if and only if $\mathbf{c}$ is colinear to $\mathbf{a}$. Then,

$$\max_{\mathbf{c}\in\mathbb{R}^D, \|\mathbf{c}\|=1} \left|\mathbf{a}^T\mathbf{c}\right| = \|\mathbf{a}\|. \tag{17}$$

Using (17), our optimization problem in (7) becomes

$$\max_{\substack{\mathcal{I}\subset[N]\\|\mathcal{I}|=K}} \max_{\substack{\mathbf{x}\in\mathbb{R}^K\\\|\mathbf{x}\|=1}} \left\|\mathbf{V}_{\mathcal{I},:}^T\mathbf{x}\right\| = \max_{\substack{\mathcal{I}\subset[N]\\|\mathcal{I}|=K}} \max_{\substack{\mathbf{x}\in\mathbb{R}^K\\\|\mathbf{x}\|=1}} \max_{\substack{\mathbf{c}\in\mathbb{R}^D\\\|\mathbf{c}\|=1}} \left|\mathbf{x}^T\mathbf{V}_{\mathcal{I},:}\mathbf{c}\right| = \max_{\substack{\mathbf{c}\in\mathbb{R}^D\\\|\mathbf{c}\|=1}} \max_{\substack{\mathcal{I}\subset[N]\\|\mathcal{I}|=K}} \max_{\substack{\mathbf{x}\in\mathbb{R}^K\\\|\mathbf{x}\|=1}} \left|\mathbf{x}^T\mathbf{u}_{\mathcal{I}}(\mathbf{c})\right| \tag{18}$$

where

$$\mathbf{u}(\mathbf{c}) \overset{\triangle}{=} \mathbf{V}\mathbf{c}. \tag{19}$$

The rightmost equality in (18) is obtained by interchanging the maximizations. This is a critical step of the auxiliary unit vector technique. It unlocks the constant-rank structure of $\mathbf{V}$ and allows us to consider a simple rank-1 problem for each value of $\mathbf{c}$. Indeed, for each $\mathbf{c} \in \mathbb{R}^D$, the inner double maximization problem

$$\max_{\substack{\mathcal{I}\subset[N]\\|\mathcal{I}|=K}} \max_{\substack{\mathbf{x}\in\mathbb{R}^K\\\|\mathbf{x}\|=1}} \left|\mathbf{u}_{\mathcal{I}}(\mathbf{c})^T\mathbf{x}\right| \tag{20}$$

is equivalent to the rank-1 optimization problem in (10) that, according to (15), results in the optimal support (for fixed $\mathbf{c}$)

$$\mathcal{I}(\mathbf{c}) \overset{\triangle}{=} \text{top}_K(\mathbf{u}(\mathbf{c})) \tag{21}$$

which is obtained with complexity $\mathcal{O}(N)$. Then, according to (18), the solution to our original problem in (9) is met by collecting all possible supports $\mathcal{I}(\mathbf{c})$ as $\mathbf{c}$ scans the unit-radius D-dimensional hypersphere. That is, $\mathcal{I}_{\text{opt}}$ in (9) belongs to

$$\mathcal{S} \overset{\triangle}{=} \bigcup_{\mathbf{c}\in\mathbb{R}^D, \|\mathbf{c}\|=1} \mathcal{I}(\mathbf{c}). \tag{22}$$

Set $\mathcal{S}$ contains candidate supports $\mathcal{I} \subset [N]$ one of which is the solution to our original optimization problem. If $\mathcal{S}$ was available, then one would have to compare the elements of $\mathcal{S}$ against the metric of interest in (9) to obtain the optimal support $\mathcal{I}_{\text{opt}}$. Therefore, the size of $\mathcal{S}$ and the complexity to build $\mathcal{S}$ determine the overall complexity to solve (9). Our major contribution in this work is that we show that the cardinality of $\mathcal{S}$ is upper bounded by

$$|\mathcal{S}| \leq 2^{D-1}\binom{D}{\lfloor\frac{D}{2}\rfloor}\binom{N}{D} = \mathcal{O}\left(N^D\right) \tag{23}$$

and develop an algorithm to build $\mathcal{S}$ with complexity $\mathcal{O}\left(N^{D+1}\right)$. After $\mathcal{S}$ is constructed, each element (support) $\mathcal{I}$ of it is mapped to the principal singular value of the $K \times D$ matrix $\mathbf{V}_{\mathcal{I},:}$ with complexity $\mathcal{O}\left(KD^2\right) = \mathcal{O}(K)$, since D is constant. Finally, all computed singular values are compared with each other to obtain the optimal support $\mathcal{I}_{\text{opt}}$ in (9). Then, the solution to our original problem in (1) is the principal left singular vector of the $K \times D$ matrix

$\mathbf{V}_{\mathcal{I}_{\text{opt}},:}$, computed with complexity $\mathcal{O}\left(KD^2\right) = \mathcal{O}\left(K\right)$. Therefore, we compute the optimal support $\mathcal{I}_{\text{opt}}$ and the sparse principal component of a rank-D matrix with complexity $\mathcal{O}\left(N^{D+1} + N^D K\right) = \mathcal{O}\left(N^{D+1}\right)$.

A constructive proof is presented in detail in the next three subsections. To give some insight of the proof, we begin with the simple case $D = 2$. Then, we generalize our proof for the case of any arbitrary D.

C. Rank-2: A simple instance of our proof

If $D = 2$, then $\mathbf{V}$ has size $N \times 2$ and the auxiliary vector $\mathbf{c}$ is a length-2, unit vector that, as in [27], can be parameterized in an auxiliary angle $\phi \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right]$, that is,

$$\mathbf{c}(\phi) \triangleq \begin{bmatrix} \sin\phi \\ \cos\phi \end{bmatrix}, \quad \phi \in \Phi \triangleq \left(-\frac{\pi}{2}, \frac{\pi}{2}\right]. \tag{24}$$

Hence, $\mathbf{c}(\phi)$ lies on the unit-radius semicircle.[3] Then, the candidate set in (22) is re-expressed as

$$\mathcal{S} = \bigcup_{\phi \in \Phi} \mathcal{I}(\phi) \tag{25}$$

where, according to (21),

$$\mathcal{I}(\phi) \triangleq \text{top}_K(\mathbf{u}(\phi)) \tag{26}$$

and, according to (19),

$$\mathbf{u}(\phi) \triangleq \mathbf{V}\mathbf{c}(\phi)\,. \tag{27}$$

That is, for any given $\phi \in \Phi$, the corresponding support $\mathcal{I}(\phi)$ is obtained with complexity $\mathcal{O}(N)$ by selecting the indices of the K absolutely largest elements of $\mathbf{u}(\phi)$.

However, why should ϕ simplify the computation of a solution? The intuition behind the auxiliary unit vector technique is that every element of $\mathbf{u}(\phi)$ is actually a continuous function of ϕ, i.e., a curve (or 1-manifold) in ϕ, since

$$\mathbf{u}(\phi) = \mathbf{V}\mathbf{c}(\phi) = \begin{bmatrix} V_{1,1}\sin\phi + V_{1,2}\cos\phi \\ V_{2,1}\sin\phi + V_{2,2}\cos\phi \\ \vdots \\ V_{N,1}\sin\phi + V_{N,2}\cos\phi \end{bmatrix}. \tag{28}$$

Hence, the support that corresponds to the K absolutely largest elements of $\mathbf{u}(\phi)$ at a given point ϕ is a function of ϕ. Due to the continuity of the curves and the discrete nature of the support, we expect that the support $\mathcal{I}(\phi)$ will retain the same elements in an area around ϕ. Therefore, we expect the formation of intervals on Φ, within which the indices of the K absolutely largest elements of $\mathbf{u}(\phi)$ remain unaltered. A support $\mathcal{I}$ might change only if the sorting of the amplitudes of two elements in $\mathbf{u}(\phi)$, say $|u_i(\phi)|$ and $|u_j(\phi)|$, changes. This occurs at points ϕ where $|u_i(\phi)| = |u_j(\phi)|$, that is, points where two curves intersect. Finding all these *intersection points*

[3]We ignore the other semicircle because any pair of angles ϕ_1 and ϕ_2 with difference π results in opposite vectors $\mathbf{c}(\phi_1) = -\mathbf{c}(\phi_2)$ which, however, are equivalent with respect to the optimization metric in (18) and produce the same support $\mathcal{I}\left(\mathbf{c}(\phi_1)\right) = \mathcal{I}\left(\mathbf{c}(\phi_2)\right)$ in (21).

is sufficient to determine intervals and construct all possible candidate supports $\mathcal{I}$. Among all candidate supports, lies the support that corresponds to the optimal K-sparse principal component. Exhaustively checking the supports $\mathcal{I}$ of all intervals suffices to retrieve the optimal one. Surprisingly, the number of these intervals is exactly equal to number of possible intersections among the amplitudes of $|\mathbf{u}(\phi)|$, which is exactly equal to $2\binom{N}{2} = \mathcal{O}\left(N^2\right)$, counting all possible combinations of element pairs.

Before we proceed, in Fig. 1, we illustrate the interval partitioning of Φ, where we set $N = 4$ and $K = 2$ and plot the 4 curves that originate from the 4 rows of $|\mathbf{u}(\phi)| = |\mathbf{V}\mathbf{c}(\phi)|$. Intervals are formed, within which the sorting of the curves does not change. The borders of the intervals are denoted by vertical dashed lines at points of curve intersections. Our approach creates 12 intervals which exceeds the total number of possible supports, however this is not true for greater values of N. Moreover, we use R_i to denote *regions* in Φ, within which, although the sorting changes, the set of K-largest curves does not change. These regions is an interesting feature that might further decrease the number of intervals we need to check. We exploit this interesting feature in the serial implementation of our algorithm in Section IV.

D. Rank-2: Algorithmic developments and complexity

Our goal is the construction of all possible candidate K-sparse vectors, determined by the support $\mathcal{I}$ of each interval on Φ. This is a two-step process. First, we identify interval borders and, then, we determine the supports associated with these intervals.

Algorithmic Steps: We first determine all possible intersections of curve pairs in $|\mathbf{u}(\phi)|$. Any pair $\{i, j\}$ of distinct elements in $|\mathbf{u}(\phi)|$ is associated with two intersections: $u_i(\phi) = u_j(\phi)$ and $u_i(\phi) = -u_j(\phi)$. Solving these two equations with respect to ϕ determines a possible point where a new support of indices of the K-largest values of $|\mathbf{u}(\phi)|$ might occur.

Observe that, at an intersection point, the intersecting curves are absolutely equal, i.e., $|u_i(\phi)| = |u_j(\phi)|$. Exactly at the point of intersection, all but 2 (the ith and jth) coordinates of a candidate K-sparse vector can be determined by solving a rank-1 instance of the problem. However, we are left with ambiguity with respect to 2 coordinates i and j that needs to be resolved. To resolve this ambiguity, we can visit the “rightmost” point of Φ, that is, $\frac{\pi}{2}$. There, due to the continuity of $u_i(\phi)$ and $\pm u_j(\phi)$, the sortings within the two cells defined by the two intersections, will be the identical, or opposite sortings of $u_i\left(\frac{\pi}{2}\right)$ and $\pm u_j\left(\frac{\pi}{2}\right)$, depending on whether $u_i(\phi)$ and $\pm u_j(\phi)$ are both positive, or negative at the intersection point, respectively.

Having described how to resolve ambiguities, we have fully described a way to calculate the support $\mathcal{I}$ at any intersection point. Apparently, all intersection points, that is, all $\binom{N}{2}$ pairwise combinations of elements in $|\mathbf{u}(\phi)|$, have to be examined to yield a corresponding support $\mathcal{I}$.

Computational Complexity: A single intersection point can be computed in time $\mathcal{O}(1)$. At an intersection point, we have to determine the Kth-order element of $|\mathbf{u}(\phi)|$ and the $K-1$ elements larger than that, which can be done in time $\mathcal{O}(N)$. Resolving an ambiguity costs $\mathcal{O}(1)$. Therefore, in total, finding a single support $\mathcal{I}$ costs $\mathcal{O}(N)$.

Constructing all candidate supports $\mathcal{I}$ requires examining all $2\binom{N}{2}$ points, implying a total construction cost of $2\binom{N}{2} \times \mathcal{O}(N) = \mathcal{O}\left(N^3\right)$.

E. Rank-D: A generalized proof

In the general case, $\mathbf{V}$ is a $N \times D$ matrix. In this subsection, we present our main result where we prove that the problem of identifying the K-sparse principal component of a rank-D matrix is solvable with complexity $\mathcal{O}\left(N^{D+1}\right)$. The statement is true for any value of K (that is, even if K is a function of N). The rest of this subsection contains a constructive proof of this statement.

Since $\mathbf{V}$ has size $N \times D$, the auxiliary vector $\mathbf{c}$ is a length-D, unit vector. We begin our constructive proof by introducing the auxiliary-angle vector $\boldsymbol{\phi} \in \Phi^{D-1}$ and parameterizing $\mathbf{c}$, as in [28], according to

$$\mathbf{c}(\boldsymbol{\phi}) \stackrel{\triangle}{=} \begin{bmatrix} \sin\phi_1 \\ \cos\phi_1 \sin\phi_2 \\ \cos\phi_1 \cos\phi_2 \sin\phi_3 \\ \vdots \\ \cos\phi_1 \cos\phi_2 \ldots \sin\phi_{D-1} \\ \cos\phi_1 \cos\phi_2 \ldots \cos\phi_{D-1} \end{bmatrix}. \tag{29}$$

Hence, $\mathbf{c}(\boldsymbol{\phi})$ lies on the unit-radius semihypersphere.[4] Then, the candidate set in (22) is re-expressed as

$$\mathcal{S} = \bigcup_{\boldsymbol{\phi} \in \Phi^{D-1}} \mathcal{I}(\boldsymbol{\phi}) \tag{30}$$

where, according to (21),

$$\mathcal{I}(\boldsymbol{\phi}) \stackrel{\triangle}{=} \operatorname{top}_K(\mathbf{u}(\boldsymbol{\phi})) \tag{31}$$

and, according to (19),

$$\mathbf{u}(\boldsymbol{\phi}) \stackrel{\triangle}{=} \mathbf{V}\mathbf{c}(\boldsymbol{\phi}). \tag{32}$$

That is, for any given $\boldsymbol{\phi} \in \Phi^{D-1}$, the corresponding support $\mathcal{I}(\boldsymbol{\phi})$ is obtained with complexity $\mathcal{O}(N)$ by selecting the indices of the K absolutely largest elements of $\mathbf{u}(\boldsymbol{\phi})$.

To gain some intuition into the purpose of inserting the auxiliary-angle vector $\boldsymbol{\phi}$, notice that every element of

$$\mathbf{u}(\boldsymbol{\phi}) = \begin{bmatrix} \mathbf{V}_{1,:}\mathbf{c}(\boldsymbol{\phi}) \\ \mathbf{V}_{2,:}\mathbf{c}(\boldsymbol{\phi}) \\ \vdots \\ \mathbf{V}_{N,:}\mathbf{c}(\boldsymbol{\phi}) \end{bmatrix} = \begin{bmatrix} V_{1,1}\sin\phi_1 + \sum_{d=2}^{D-1} V_{1,d} \prod_{i=1}^{d-1} \cos\phi_i \sin\phi_d + V_{1,D} \prod_{i=1}^{D-1} \cos\phi_i \\ V_{2,1}\sin\phi_1 + \sum_{d=2}^{D-1} V_{2,d} \prod_{i=1}^{d-1} \cos\phi_i \sin\phi_d + V_{2,D} \prod_{i=1}^{D-1} \cos\phi_i \\ \vdots \\ V_{N,1}\sin\phi_1 + \sum_{d=2}^{D-1} V_{N,d} \prod_{i=1}^{d-1} \cos\phi_i \sin\phi_d + V_{N,D} \prod_{i=1}^{D-1} \cos\phi_i \end{bmatrix} \tag{33}$$

[4]As in the rank-2 case, we ignore the other semihypersphere because any pair of vectors $\boldsymbol{\phi}$ and $\tilde{\boldsymbol{\phi}}$ whose first elements ϕ_1 and $\tilde{\phi}_1$, respectively, have difference π results in opposite vectors $\mathbf{c}(\boldsymbol{\phi}) = -\mathbf{c}(\tilde{\boldsymbol{\phi}})$ which, however, are equivalent with respect to the optimization metric in (18) and produce the same support $\mathcal{I}\left(\mathbf{c}(\boldsymbol{\phi})\right) = \mathcal{I}(\mathbf{c}(\tilde{\boldsymbol{\phi}}))$ in (21).

is actually a continuous function of $\boldsymbol{\phi}$ and so are the elements of $|\mathbf{u}(\boldsymbol{\phi})|$. That is, each element of $|\mathbf{u}(\boldsymbol{\phi})|$ is a hypersurface (or $(D-1)$-manifold) in the D-dimensional space $\Phi^{D-1} \times [0, \infty)$. When we sort the N elements of $|\mathbf{u}(\boldsymbol{\phi})|$ at a given point $\boldsymbol{\phi}$, we actually sort the N hypersurfaces at point $\boldsymbol{\phi}$. The key observation in our algorithm is that, due to their continuity, the hypersurfaces will retain their sorting in an area "around" $\boldsymbol{\phi}$. This implies the partition of $\Phi^{D-1} \times [0, \infty)$ into cells $\mathcal{C}_1, \mathcal{C}_2, \ldots$, each of which (say, cell $\mathcal{C}$) is associated with a single set $\mathcal{I}^+(\mathcal{C}) \subset [N]$ of indices of hypersurfaces that lie above $\mathcal{C}$ and a single set $\mathcal{I}^-(\mathcal{C}) = [N] - \mathcal{I}^+(\mathcal{C})$ of indices of hypersurfaces that lie below it. Moreover, each cell $\mathcal{C}$ contains at least one vertex (that is, intersection of D hypersurfaces). Finally, for any $\boldsymbol{\phi} \in \Phi$, there is a unique cell $\mathcal{C} \subset \Phi^{D-1} \times [0, \infty)$, called "normal," which contains uncountably many points in $\{\boldsymbol{\phi}\} \times [0, \infty)$ and is associated with a single index-set $\mathcal{I}^+(\mathcal{C})$ of cardinality K (that is, exactly K hypersurfaces lie above $\mathcal{C}$). In fact, the indices of these K hypersurfaces (that is, the elements of $\mathcal{I}^+(\mathcal{C})$) are the elements of support $\mathcal{I}(\boldsymbol{\phi})$. Although our discussion refers to the general-D case, for illustrative purposes we consider again the case $D = 2$ and, in Fig. 2, we revisit the example that we presented in Subsection III-C. The normal cells that are created by the $N = 4$ curves are the shaded ones. These cells carry the property that lie below exactly $K = 2$ curves. We observe that there is a one-to-one correspondence between normal cells and regions R_i.

According to (30) and the above observations, we need to determine the index-set $\mathcal{I}^+(\mathcal{C})$ of every normal cell $\mathcal{C}$ in the partition. If we collect all such index-sets, then we have constructed $\mathcal{S}$ in (30). This will be achieved if, instead, we identify all cells in $\Phi^{D-1} \times [0, \infty)$ and, for each cell, determine the K largest hypersurfaces that lie above an arbitrary point of it. The latter will return the desired index-set $\mathcal{I}^+(\mathcal{C})$ if the cell is normal. In Fig. 2, we observe that, for each normal cell, the indices of the $K = 2$ largest curves that lie above it can be computed at the leftmost vertex of it (we can ignore the leftmost normal cell because it produces the same indices with the rightmost one). In the following, we identify all cells in the partition and compute a size-K support $\mathcal{I}$ for each such cell. This way, we obtain the index-set of any normal cell, among which one is the optimal support $\mathcal{I}$ in (9).

Since each cell contains at least one vertex, we only need to find all vertices in the partition and determine $\mathcal{I}^+$ for all neighboring cells. Recall that a vertex is an intersection of D hypersurfaces. Consider D arbitrary hypersurfaces; say, for example, $|u_1(\boldsymbol{\phi})|$, $|u_2(\boldsymbol{\phi})|$, …, $|u_D(\boldsymbol{\phi})|$. Their intersection satisfies $|u_1(\boldsymbol{\phi})| = |u_2(\boldsymbol{\phi})| = \ldots = |u_D(\boldsymbol{\phi})|$ and is computed by solving the system of equations

$$\left\{ \begin{array}{c} u_1(\boldsymbol{\phi}) \pm u_2(\boldsymbol{\phi}) = 0 \\ u_1(\boldsymbol{\phi}) \pm u_3(\boldsymbol{\phi}) = 0 \\ \vdots \\ u_1(\boldsymbol{\phi}) \pm u_D(\boldsymbol{\phi}) = 0 \end{array} \right\} \tag{34}$$

or, equivalently,

$$\begin{bmatrix} \mathbf{V}_{1,:} \pm \mathbf{V}_{2,:} \\ \mathbf{V}_{1,:} \pm \mathbf{V}_{3,:} \\ \vdots \\ \mathbf{V}_{1,:} \pm \mathbf{V}_{D,:} \end{bmatrix} \mathbf{c}(\boldsymbol{\phi}) = \mathbf{0}. \tag{35}$$

For any sign combination, the solution to the latter consists of the spherical coordinates of the unit vector in the null space of the $(D-1) \times D$ leftmost matrix.[5] Then, the index-set $\mathcal{I}^+$ that corresponds to a neighboring cell is computed by

$$\text{top}_K(\mathbf{V}\mathbf{c}(\boldsymbol{\phi})). \tag{36}$$

Note that the D intersecting hypersurfaces have the same value at $\boldsymbol{\phi}$. Hence, (36) returns ambiguity regarding the sorting of these particular D hypersurfaces. If $d < D$ hypersurfaces of these belong to the K largest ones, then, due to this ambiguity, we have to consider all $\binom{D}{d}$ combinations of d hypersurfaces among the D intersecting ones, where $\binom{D}{d} < \binom{D}{\lfloor \frac{D}{2} \rfloor}$. Finally, we have to repeat the above procedure for all 2^{D-1} sign combinations in (35) and any combination of D intersecting hypersurfaces among the N ones. The total number of combinations is $\binom{N}{D}$, hence the cardinality of $\mathcal{S}$ is upper bounded by $2^{D-1}\binom{D}{\lfloor \frac{D}{2} \rfloor}\binom{N}{D} = \mathcal{O}\left(N^D\right)$.

A pseudocode that includes all the above steps of our algorithm is presented in Fig. 3. Our algorithm's complexity to build $\mathcal{S}$ is determined by the complexity to build each element of it (i.e., each index-set $\mathcal{I}^+$) for each examined intersection through (36). Note that *top*$_k$ has complexity $\mathcal{O}(N)$ and the cardinality of $\mathcal{S}$ is $\mathcal{O}\left(N^D\right)$. Hence, the overall complexity to build $\mathcal{S}$ is $\mathcal{O}\left(N^{D+1}\right)$. Finally, we mention that the computation of each element of $\mathcal{S}$ is performed independently of each other. Therefore, the proposed algorithm to build $\mathcal{S}$ and solve (1) or, equivalently, (9) with complexity $\mathcal{O}\left(N^{D+1}\right)$ is fully parallelizable and memory efficient.

IV. An Algorithm of Complexity $\mathcal{O}\left(N^2 \log N\right)$ for Rank-2 Matrices

In the special case of a rank-2 matrix $\mathbf{A}$ (i.e., $D=2$), the algorithm that we presented in Section III computes the sparse principal component of $\mathbf{A}$ with complexity $\mathcal{O}\left(N^3\right)$ for any sparsity value. In this present section, we develop an algorithm that computes the sparse principal component of $\mathbf{A}$ with complexity $\mathcal{O}\left(N^2 \log N\right)$.

A. *A serial algorithm for rank-2 matrices*

For the rank-2 case, the algorithm of Subsection III-D relied on identifying a polynomial number of non-overlapping intervals on Φ, each associated with a candidate support set $\mathcal{I}$. These intervals are induced by the pairwise intersections of curves in $|\mathbf{u}(\phi)|$. The candidate support set associated with an interval is determined at the leftmost point of the interval. The two-step algorithm first computes all $2\binom{N}{2}$ pairwise intersection points. In

[5]If the $(D-1) \times D$ matrix is full-rank, then its null space has rank 1 and $\mathbf{c}(\boldsymbol{\phi})$ is uniquely determined (within a sign ambiguity which, however, does not affect the final decision on the index-set). If, instead, the $(D-1) \times D$ matrix is rank-deficient, then the intersection of the D hypersurfaces (i.e., the solution of (35)) is a p-manifold (with $p \geq 1$) on the D-dimensional space and does not generate a new cell. Hence, combinations of D rows of $\mathbf{V}$ that result in linearly dependent rows of the $(D-1) \times D$ matrix in (35) can be simply ignored.

the second step, it determines $\mathcal{I}(\hat{\phi})$, the set of indices of the K largest elements of $|\mathbf{u}(\hat{\phi})|$, at each intersection point $\hat{\phi}$ individually, in linear time. In this Section, we present an algorithmic enhancement that reduces the overall computational complexity, exploiting the correlation between candidate support sets of neighboring intervals.

The relative order of $|u_i(\phi)|$ and $|u_i(\phi)|$ changes only at the points where the ith and jth curves intersect. Conversely, if those two are the only curves intersecting at a point $\hat{\phi}$, then the ordering of the remaining elements of $|\mathbf{u}(\phi)|$ in the intervals immediately preceding and succeeding $\hat{\phi}$ is identical. The limited change on ordering of the curves in $|\mathbf{u}(\phi)|$ across an intersection point $\hat{\phi}$ implies that the differences between the candidate support sets of the adjacent intervals cannot be arbitrary. The support sets associated with two neighboring intervals differ in at most one element. More formally, let $\hat{\phi}$ be the intersection point of the ith and jth curves of $|\mathbf{u}(\phi)|$. There exists an $\epsilon > 0$ such that $\hat{\phi}$ is the only intersection point lying in $[\hat{\phi}-\epsilon, \hat{\phi}+\epsilon]$. If $|u_i(\phi)| < |u_j(\phi)|$ in $[\hat{\phi}-\epsilon, \hat{\phi})$, then $|u_i(\phi)| > |u_j(\phi)|$ in $(\hat{\phi}, \hat{\phi}+\epsilon]$, and vice versa. The ordering of all other curves remains unaltered over $\hat{\phi}$. If the ith and jth curves are both members of the candidate support in the interval preceding $\hat{\phi}$, then $\mathcal{I}(\hat{\phi}+\epsilon) = \mathcal{I}(\hat{\phi}-\epsilon)$. The same holds if neither is included in $\mathcal{I}(\hat{\phi}-\epsilon)$. On the contrary, if exactly one of i and j belongs to $\mathcal{I}(\hat{\phi}-\epsilon)$, then the candidate sets associated with the two neighboring intervals differ in exactly one element. If, w.l.o.g., the ith curve is the one belonging to $\mathcal{I}(\hat{\phi}-\epsilon)$, then $\mathcal{I}(\hat{\phi}+\epsilon) = \Big(\mathcal{I}(\hat{\phi}-\epsilon)\backslash\{i\}\Big) \cup \{j\}$.

The key observation is that, if the candidate support set associated with a particular interval is known, then the set associated with a neighboring interval can be determined with a constant complexity. The above observations readily suggest the following procedure for determining the candidate support sets:

1) Compute all $2\binom{N}{2}$ pairwise intersection points of the curves in $|\mathbf{u}(\phi)|$.
2) Sort the intersection points in increasing order. Let $\phi_1, \phi_2, \dots, \phi_{2\binom{N}{2}}$ be the sorted sequence.
3) Determine $\mathcal{I}(\phi_1)$, the set of indices of the K largest curves in $|\mathbf{u}(\phi)|$ at the first intersection point ϕ_1.
4) Successively determine $\mathcal{I}(\phi_t)$ at consecutive intersection points. At ϕ_t, the candidate support set $\mathcal{I}(\phi_t)$ is determined by appropriately updating $\mathcal{I}(\phi_{t-1})$, the set associated with the previous interval.

Once all candidate support sets have been collected, the optimal solution is determined as in the algorithm of Section III. An illustrative figure that explains the above steps is presented in Fig. 4, using the same 4×2 matrix $\mathbf{V}$ as the one examined in Section III.

A pseudocode that implements the above procedure (which we name the *serial* algorithm) is presented in Fig. 5. The serial algorithm improves upon the computational complexity of its parallel counterpart by circumventing the $O(N)$ construction of the candidate set at each individual intersection point. The computation and sorting of all intersection points (steps 1 and 2) is performed in $O(N^2 \log N)$ operations. The construction of $\mathcal{I}(\phi_1)$ is performed in linear time. Finally, each successive update in the last step requires $O(1)$ operations. Therefore, the overall complexity of the serial algorithm is $O\left(N^2 \log N\right)$ versus the complexity $O\left(N^3\right)$ of the algorithm that we presented in Section III for $D = 2$. Its disadvantage is that the serial algorithm is not parallelizable.

*B. A modified serial algorithm for rank-*2 *matrices*

The parallel (Section III) and serial (Subsection IV-A) algorithms described so far collect the candidate support sets by examining all pairwise intersection points. The number of distinct candidate support sets, however, can be significantly less: multiple intervals on Φ may be associated with the same support set. In the following, we describe a simple modification of the serial algorithm that aims to reduce the total number of intersection points computed and examined. Although the worst case complexity remains the same, the modified serial algorithm may significantly speed up execution in certain cases.

Consider three consecutive intersection points ϕ_1, ϕ_2, and ϕ_3, such that the candidate support sets on the two sides of ϕ_2 are different (see, for example, Fig. 6). Let $\mathcal{I}(\phi_1)$ denote the set of indices of the K largest curves in $|\mathbf{u}(\phi)|$ in the interval $[\phi_1, \phi_2)$. Similarly, $\mathcal{I}(\phi_2)$ is the set associated with $[\phi_2, \phi_3)$. The two sets differ by one element: if $\mathcal{C} = \mathcal{I}(\phi_1) \cap \mathcal{I}(\phi_2)$ is the set of the $K-1$ common elements, then $\mathcal{I}(\phi_1) = \mathcal{C} \cup \{i\}$ and $\mathcal{I}(\phi_2) = \mathcal{C} \cup \{j\}$, for some curves i and j.

At ϕ_2, over which the candidate support set changes, the two curves intersect. The jth curve joins the set of K largest curves of $|\mathbf{u}(\phi)|$, displacing the ith curve which was a member of $\mathcal{I}(\phi_1)$. In particular, the ith curve must be the smallest element of $|\mathbf{u}(\phi)|$ in $[\phi_1, \phi_2)$. To see that, assume for the sake of contradiction that among the curves in $\mathcal{I}(\phi_1)$, the lth curve was the smallest one in $[\phi_1, \phi_2)$, where $l \neq i$. Then, $u_i(\phi) > u_l(\phi) > u_j(\phi)$ in $[\phi_1, \phi_2)$. By assumption, $u_i(\phi) = u_j(\phi)$ at ϕ_2. Due to the continuity of the curves, there must exist a point in $[\phi_1, \phi_2)$ where either $u_i(\phi) = u_l(\phi)$ or $u_l(\phi) = u_j(\phi)$. However, no intersection point lies in $[\phi_1, \phi_2)$. Following a similar argument, the jth curve must be the largest curve in $[\phi_1, \phi_2)$ among those not included in $\mathcal{I}(\phi_1)$. Moreover, the jth curve becomes the Kth-order element of $|\mathbf{u}(\phi)|$ in $[\phi_2, \phi_3)$, i.e., it is the smallest curve in $\mathcal{I}(\phi_2)$.

The key observation is that, along Φ, the candidate support set $\mathcal{I}(\phi)$ changes only at the intersection points of the Kth-order curve in $|\mathbf{u}(\phi)|$. Assume that the candidate support set $\mathcal{I}(\phi_0)$ is known at $\phi_0 = -\frac{\pi}{2}$ and the ith curve is the Kth-order curve in $|\mathbf{u}(\phi)|$, i.e., the smallest curve in $\mathcal{I}(\phi_0)$. Moving along Φ, the first point where the candidate support set can potentially change is the closest intersection point of the ith curve. Let $\phi_1 \in [\phi_0, \frac{\pi}{2})$ be that point and j be the intersecting curve. If $j \notin \mathcal{I}(\phi_0)$, then the jth curve joins the set $\mathcal{I}(\phi_1)$ at ϕ_1, displacing the ith curve. If $j \in \mathcal{I}(\phi_0)$, then ϕ_1 is only a point of internal reordering for $\mathcal{I}(\phi_0)$, hence $\mathcal{I}(\phi_1) = \mathcal{I}(\phi_0)$. In either case, however, the jth curve becomes the Kth-order curve immediately after ϕ_1. Proceeding in a similar fashion, the next point where the candidate support set can potentially change is a point $\phi_2 \in [\phi_1, \frac{\pi}{2})$ closest to ϕ_1, where the jth curve intersects one of the other N curves.

A pseudocode that implements the modified serial algorithm, as described above, is presented in Fig. 7. In summary, instead of computing all pairwise intersections at the first step, the modified serial algorithm postpones the computation of the intersection points of the ith curve until the latter becomes the Kth-order curve in $|\mathbf{u}(\phi)|$. The motivation behind this enhancement lies on the fact that multiple curves might never become Kth in order. To justify this, in Fig. 8, we plot the total number of intersection points computed by the modified serial rank-2 algorithm as a function of N, for sparsity $K = N/2$. We compare with the total number of intersection points,

$2\binom{N}{2}$ and the total number of distinct candidate support sets. We observe that the total number of distinct candidate support sets (i.e., the cardinality of $\mathcal{S}$) is much smaller than the total number of intersection points between curves. The latter indicates that there is room for further reduction of the complexity of the algorithms that we developed in this present work.

V. Conclusions

We proved that the sparse principal component of an $N \times N$ matrix is computable with complexity $\mathcal{O}\left(N^{D+1}\right)$ if the matrix is positive semidefinite and its rank D is constant. This holds true for any sparsity value K (that is, even if K grows with N). Our constructive proof was accompanied by a fully parallelizable and memory efficient algorithm which computes the sparse principal component with complexity $\mathcal{O}\left(N^{D+1}\right)$. For the special case of rank-2 matrices, we developed alternative serial algorithms of complexity $\mathcal{O}\left(N^2 \log N\right)$. The construction steps and properties of the algorithms that we presented in this work indicate that implementations of even lower complexity may be possible.

References

[1] D. L. Donoho, M. Elad, and V. N. Temlyakov, “Stable recovery of sparse overcomplete representations in the presence of noise,” *IEEE Trans. Inf. Theory*, vol. 52, pp. 6-18, Jan. 2006.

[2] J. A. Tropp, “Just relax: Convex programming methods for identifying sparse signals in noise,” *IEEE Trans. Inf. Theory*, vol. 52, pp. 1030-1051, Mar. 2006.

[3] H. Zou, T. Hastie, and R. Tibshirani, “Sparse principal component analysis,” *J. Computational and Graphical Statistics*, vol. 15, pp. 265-286, 2006.

[4] B. Moghaddam, Y. Weiss, and S. Avidan, “Spectral bounds for sparse PCA: Exact and greedy algorithms,” *Advances in Neural Information Processing Systems*, vol. 18, pp. 915-922, 2006.

[5] A. d’ Aspremont, L. El Ghaoui, M. I. Jordan, and G. R. G. Lanckriet, “A direct formulation for sparse PCA using semidefinite programming,” *SIAM Review*, vol. 49, pp. 434-448, 2007.

[6] A. A. Amini and M. J. Wainwright, “High-dimensional analysis of semidefinite relaxations for sparse principal components,” in *Proc. IEEE ISIT 2008*, Toronto, ON, July 2008, pp. 2454-2458.

[7] E. Diederichs, A. Juditsky, V. Spokoiny, and C. Schütte, “Sparse non-Gaussian component analysis,” *IEEE Trans. Inf. Theory*, vol. 56, pp. 3033-3047, June 2010.

[8] C. Shen, S. Paisitkriangkrai, and J. Zhang, “Efficiently learning a detection cascade with sparse eigenvectors,” *IEEE Trans. Image Process.*, vol. 20, pp. 22-35, Jan. 2011.

[9] M. O. Ulfarsson and V. Solo, “Vector l_0 sparse variable PCA,” *IEEE Trans. Signal. Process.*, vol. 59, pp. 1949-1958, May 2011.

[10] C. Boutsidis, P. Drineas, and M. Magdon-Ismail, “Sparse features for PCA-like linear regression,” *Advances in Neural Information Processing Systems*, vol. 24, pp. 2285-2293, 2011.

[11] N. Singh, B. A. Miller, N. T. Bliss, and P. J. Wolfe, “Anomalous subgraph detection via sparse principal component analysis,” in *Proc. IEEE SSP 2011*, Nice, France, June 2011, pp. 485-488.

[12] I. D. Schizas and G. B. Giannakis, “Covariance eigenvector sparsity for compression and denoising,” *IEEE Trans. Signal. Process.*, vol. 60, pp. 2408-2421, May 2012.

[13] D. Wei, C. K. Sestok, and A. V. Oppenheim, “Sparse filter design under a quadratic constraint: Low-complexity algorithms,” *IEEE Trans. Signal Process.*, vol. 61, pp. 857-870, Feb. 2013.

[14] M. Elad and I. Yavneh, “A plurality of sparse representations is better than the sparsest one alone,” *IEEE Trans. Inf. Theory*, vol. 55, pp. 4701-4714, Oct. 2009.

[15] C. G. Tsinos, A. S. Lalos, and K. Berberidis, “Sparse subspace tracking techniques for adaptive blind channel identification in OFDM systems,” in *Proc. IEEE ICASSP 2012*, Kyoto, Japan, Mar. 2012, pp. 3185-3188.

[16] B. Moghaddam, Y. Weiss, and S. Avidan, “Generalized spectral bounds for sparse LDA,” in *Proc. 23rd Intern. Conf. Machine Learning*, Pittsburgh, PA, June 2006, pp. 641-648.

[17] B. K. Sriperumbudur, D. A. Torres, and G. R. G. Lanckriet, “Sparse eigen methods by DC programming,” in *Proc. 24th Intern. Conf. Machine Learning*, Corvallis, OR, June 2007, pp. 831-838.

[18] A. d’ Aspremont, F. Bach, and L. El Ghaoui, “Optimal solutions for sparse principal component analysis,” *J. Machine Learning Research*, vol. 9, pp. 1269-1294, 2008.

[19] L. Mackey, “Deflation methods for sparse PCA,” *Advances in Neural Information Processing Systems*, vol. 21, pp. 1017-1024, 2009.

[20] M. Journée, Y. Nesterov, P. Richtárik, and R. Sepulchre, “Generalized power method for sparse principal component analysis,” *J. Machine Learning Research*, vol. 11, pp. 517-553, Feb. 2010.

[21] Y. Zhang and L. El Ghaoui, “Large-scale sparse principal component analysis with application to text data,” *Advances in Neural Information Processing Systems*, vol. 24, pp. 532-539, 2011.

[22] Y. Zhang, A. d’ Aspremont, and L. El Ghaoui, “Sparse PCA: Convex relaxations, algorithms and applications,” in *Handbook on Semidefinite, Cone and Polynomial Optimization*, vol. 166, pp. 915-940, 2012.

[23] M. Grbovic, C. R. Dance, and S. Vucetic, “Sparse principal component analysis with constraints,” in *Proc. 26th AAAI Conf. Artificial Intelligence*, Toronto, ON, July 2012, pp. 935-941.

[24] T. Jolliffe, N. T. Trendafilov, and M. Uddin, “A modified principal component technique based on the LASSO,” *J. Computational and Graphical Statistics*, vol. 12, pp. 531-547, 2003.

[25] Z. Ma, "Sparse principal component analysis and iterative thresholding," arXiv preprint arXiv:1112.2432, 2011.

[26] X. T. Yuan and T. Zhang, "Truncated power method for sparse eigenvalue problems," arXiv preprint arXiv:1112.2679, 2011.

[27] G. N. Karystinos and D. A. Pados, "Rank-2-optimal adaptive design of binary spreading codes," *IEEE Trans. Inf. Theory*, vol. 53, pp. 3075-3080, Sept. 2007.

[28] G. N. Karystinos and A. P. Liavas, "Efficient computation of the binary vector that maximizes a rank-deficient quadratic form," *IEEE Trans. Inf. Theory*, vol. 56, pp. 3581-3593, July 2010.

[29] K. M. Mackenthun, Jr., "A fast algorithm for multiple-symbol differential detection of MPSK," *IEEE Trans. Commun.*, vol. 42, pp. 1471-1474, Feb./Mar./Apr. 1994.

[30] W. Sweldens, "Fast block noncoherent decoding," *IEEE Commun. Lett.*, vol. 5, pp. 132-134, Apr. 2001.

[31] I. Motedayen, A. Krishnamoorthy, and A. Anastasopoulos, "Optimal joint detection/estimation in fading channels with polynomial complexity," *IEEE Trans. Inf. Theory*, vol. 53, pp. 209-223, Jan. 2007.

[32] D. S. Papailiopoulos and G. N. Karystinos, "Maximum-likelihood noncoherent OSTBC detection with polynomial complexity," *IEEE Trans. Wireless Commun.*, vol. 9, pp. 1935-1945, June 2010.

[33] D. S. Papailiopoulos, G. Abou Elkheir, and G. N. Karystinos, "Maximum-likelihood noncoherent PAM detection," *IEEE Trans. Commun.*, vol. 61, pp. 1152-1159, Mar. 2013.

[34] A. Kyrillidis and G. N. Karystinos, "Fixed-rank Rayleigh quotient maximization by an M-phase vector," submitted to *IEEE Trans. Inf. Theory*, May 2013.

[35] M. Asteris, "Sparse rank-deficient variance maximization," Diploma Thesis, Department of ECE, Technical Univ. Crete, July 2010.

[36] M. Asteris, D. S. Papailiopoulos, and G. N. Karystinos, "Sparse principal component of a rank-deficient matrix," in *Proc. IEEE ISIT 2011*, Saint Petersburg, Russia, Aug. 2011, pp. 673-677.

[37] T. H Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. MIT Press 2001, Part II, Chapter 9.

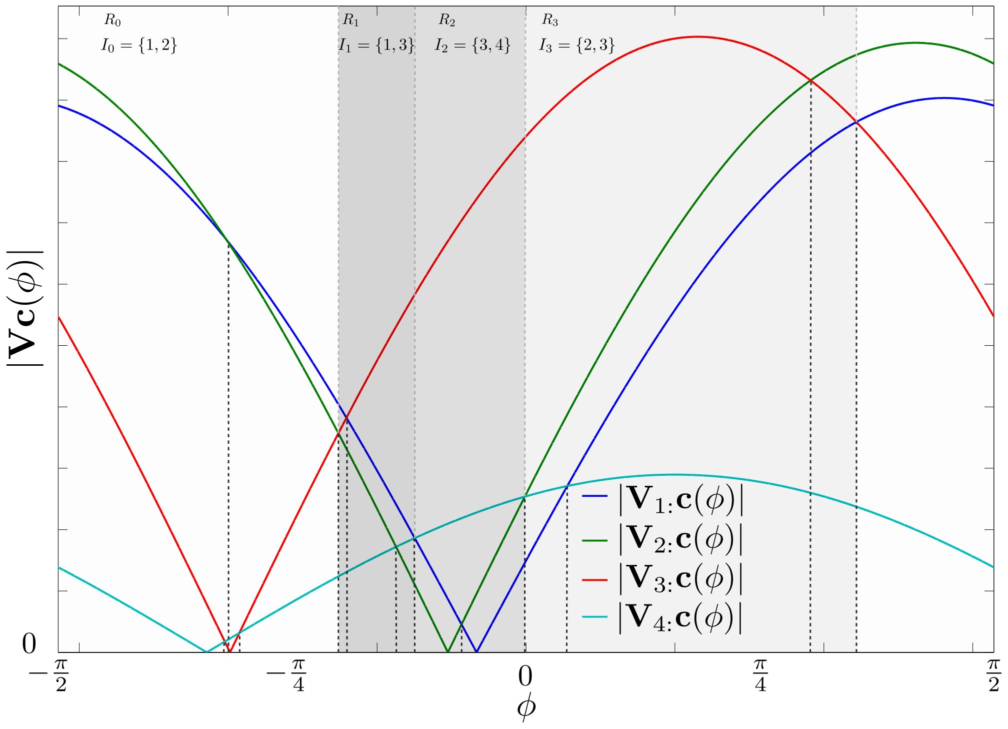

Fig. 1. The curves of $|\mathbf{Vc}(\phi)|$ for a randomly chosen matrix $\mathbf{V} \in \mathbb{R}^{4\times 2}$. Vertical dashed lines indicate the $2\binom{4}{2} = 12$ pairwise intersection points that partition Φ into intervals, within which the curve sorting does not change. For $K = 2$, we also illustrate the regions R_0 - R_3 (sets of adjacent intervals) where the candidate support set remains fixed.

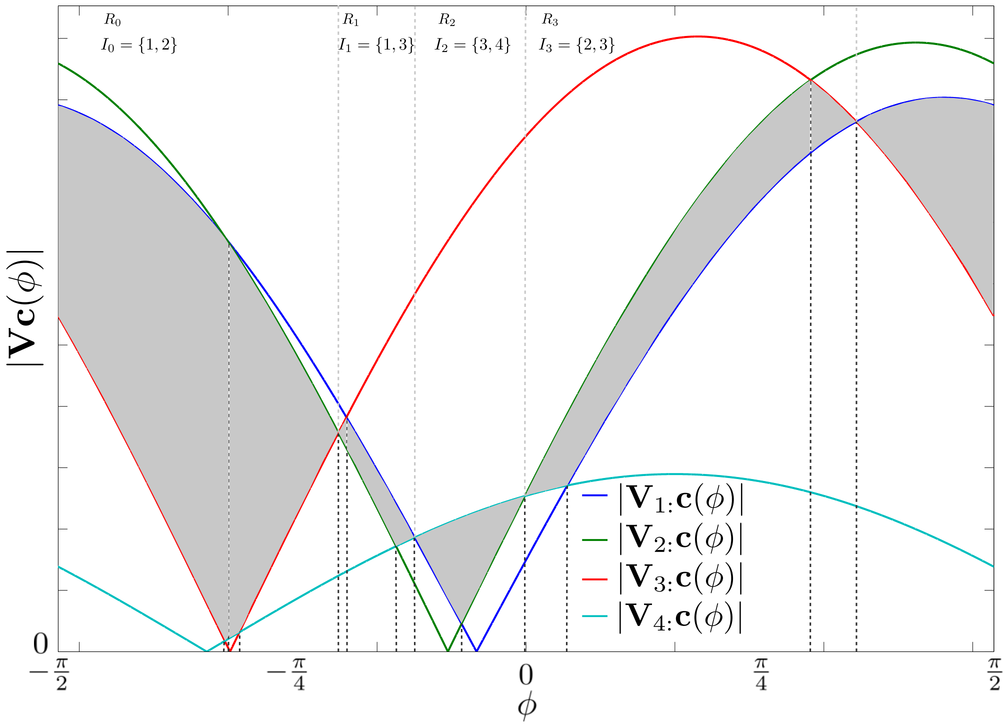

Fig. 2. Cells generated by the rows of a randomly chosen matrix $\mathbf{V} \in \mathbb{R}^{4\times 2}$. The shaded cells are the normal ones.

```
Input: $\mathbf{V} \in \mathbb{R}^{N \times D}$, $K$.
  $\mathcal{S} \leftarrow \{\}$, (Set of candidate supports).
  $\mathcal{B} \leftarrow \{b_1, \ldots, b_{D-1}\} \in \{\pm 1\}^{d-1}$
  for all $\binom{N}{D}$ sets $(i_1, \ldots, i_D) \subseteq [N]$ do
      for all sequences $(b_1, \ldots, b_{D-1}) \in \mathcal{B}$ do
          $\mathbf{c} \leftarrow$ nullspace of $\begin{bmatrix} \mathbf{V}_{i_1,:} - b_1 \mathbf{V}_{i_2,:} \\ \vdots \\ \mathbf{V}_{i_1,:} - b_{D-1} \mathbf{V}_{i_D,:} \end{bmatrix}$
          $\mathcal{I} \leftarrow \text{top}_K(\mathbf{Vc})$      ▷ If multiple entries are equal to the $K$th order element, include all in $\mathcal{I}$.
          if $|\mathcal{I}| = K$ then      ▷ No ambiguity: either all $D$ intersecting curves in $\mathcal{I}$, or none.
              $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{I}$
          else
              $\mathcal{T} \leftarrow \mathcal{I} \backslash \{i_1, \ldots, i_D\}$.
              $r \leftarrow |\mathcal{T}|$.
              for all $\binom{D}{r}$ $r$-subsets $\mathcal{M}$ from $(i_1, \ldots, i_D)$ do
                  $\hat{\mathcal{I}} \leftarrow \mathcal{T} \cup \mathcal{M}$
                  $\mathcal{S} \leftarrow \mathcal{S} \cup \hat{\mathcal{I}}$
              end for
          end if
      end for
  end for
  $\mathcal{I}_{\text{opt}} = \arg\max_{\mathcal{I} \in \mathcal{S}} \sigma_{\max}(\mathbf{V}_{\mathcal{I}_{\text{opt}},:})$.
  return $\mathcal{I}_{\text{opt}}$ & principal left singular vector of $\mathbf{V}_{\mathcal{I}_{\text{opt}},:}$.
```

Fig. 3. The algorithm for the computation of the sparse principal component of a rank-D matrix with complexity $\mathcal{O}\left(N^{D+1}\right)$.

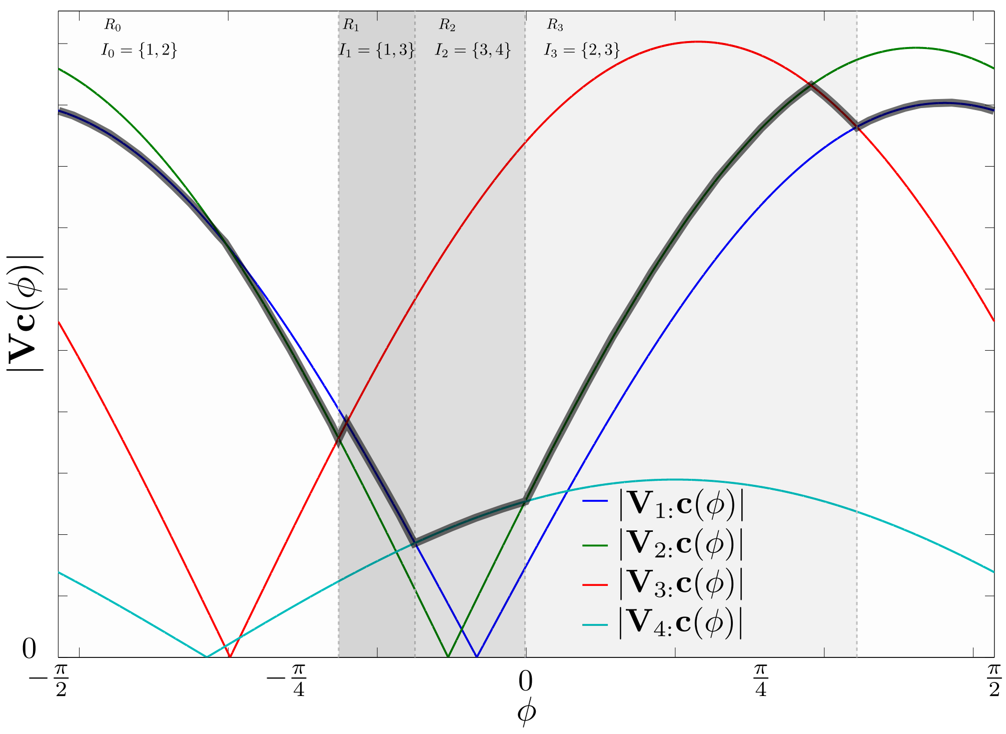

Fig. 4. Execution of the serial algorithm on a rank-2 example with $N = 4$ curves, i.e., $\mathbf{V} \in \mathbb{R}^{4\times 2}$. We seek the optimal 2-sparse solution, i.e., $K = 2$. The algorithm, starts at $\phi = -\pi/2$ and scans Φ, keeping track of the Kth order curve, highlighted with a black solid line. Vertical dashed lines indicate the intersections points at which the support of the optimal $\mathbf{x}$ changes. Regions of Φ corresponding to different support have a different background color. The optimal support in each region is depicted in the top of the figure. Note that 2 of the $\binom{N}{2} = 6$ support sets need not be considered: sets $\{1,4\}$ and $\{2,4\}$ do not appear for any ϕ.

```
**Input:** $\mathbf{V} \in \mathbb{R}^{N\times 2}$, $K$.
  $\mathcal{S} \leftarrow \{\}$, (Set of candidate supports).
  $\boldsymbol{\phi} \leftarrow$ sort$\left(\text{all } 2\binom{N}{2} \text{ pairwise intersection points}\right)$. (Maintain information on the pair $(i,j)$ associated with each intersection).
  $\mathcal{I}_0 \leftarrow \{$the indices of the top $K$ elements of $|\mathbf{V}\mathbf{c}(\phi_1)|\}$.
  **for** $t = 2 : \text{length}(\boldsymbol{\phi})$ **do**
    $(i,j) \leftarrow$ curves intersecting at $\phi_t$.
    Determine $\mathcal{I}_t$ by applying $O(1)$ changes on $\mathcal{I}_{t-1}$ as follows:
    **if** $i$ and $j$ are both included or excluded from $\mathcal{I}_{t-1}$, **then** $\mathcal{I}_t = \mathcal{I}_{t-1}$,
    **else if** only $i$ ($j$) is included in $\mathcal{I}_{t-1}$, **then** substitute $i$ ($j$) by $j$ ($i$) in $\mathcal{I}_t$.
    **end if**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{I}_t$.
  **end for**
  $\mathcal{I}_{\text{opt}} = \arg\max_{\mathcal{I}\in\mathcal{S}} \sigma_{\max}(\mathbf{V}_{\mathcal{I}_{\text{opt}},:})$.
  **return** $\mathcal{I}_{\text{opt}}$ & principal left singular vector of $\mathbf{V}_{\mathcal{I}_{\text{opt}},:}$.
```

Fig. 5. The serial rank-2 algorithm for the computation of the sparse principal component with complexity $\mathcal{O}\left(N^2 \log N\right)$.

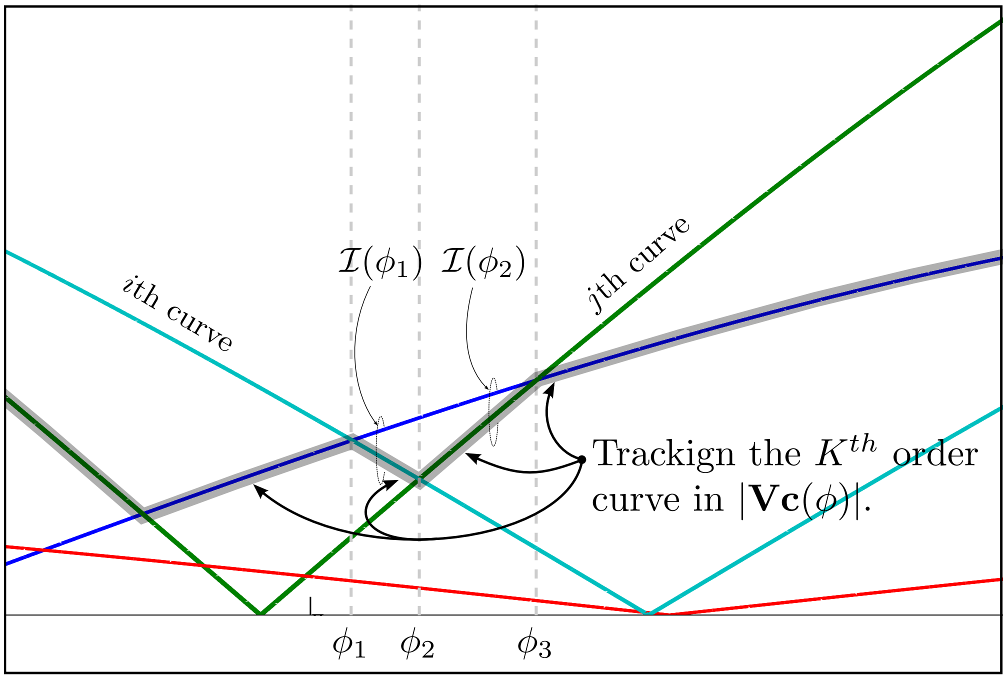

Fig. 6. Curves in $|\mathbf{u}(\phi)|$ in a constrained region of Φ for a randomly chosen matrix $\mathbf{V} \in \mathbb{R}^{4\times 2}$. We highlight the evolution of the Kth order curve, for $K = 2$. Angles ϕ_1, ϕ_2, and ϕ_3 are three consecutive intersection points. The candidate support sets in $[\phi_1, \phi_2)$ and $[\phi_2, \phi_3)$ differ by exactly one element. The Kth order curve changes over the intersection point ϕ_2.

```
Input: $\mathbf{V} \in \mathbb{R}^{N\times 2}$, $K$.
  $\phi_0 \leftarrow -\frac{\pi}{2}$, $t \leftarrow 0$,
  $\mathcal{S} \leftarrow \{\}$, (Set of candidate supports.)
  $\mathcal{I}_0 \leftarrow$ The set of indices of the $K$ absolutely largest elements of $\mathbf{Vc}(\phi_0)$.
  $i_k \leftarrow$ The index of the smallest element in $\mathcal{I}_0$.
  while $\phi_t < \frac{\pi}{2}$ do
    $t \leftarrow t + 1$.
    $\phi_t = \min\{\phi \in \left(\phi_{t-1}, \frac{\pi}{2}\right] : \exists j \in [N]\backslash\{i_k\}$ s.t. $|\mathbf{V}_{i_k,:}\mathbf{c}(\phi)| = |\mathbf{V}_{j,:}\mathbf{c}(\phi)|\}$,
        i.e., $\phi_t$ is the closest intersection of the $i_k$th curve following $\phi_t$.
    $j^* \leftarrow$ the index $j$ for which $|\mathbf{V}_{i_k,:}\mathbf{c}(\phi_t)| = |\mathbf{V}_{j,:}\mathbf{c}(\phi_t)|$
    if $j^* \in \mathcal{I}_{t-1}$ then.
      $\mathcal{I}_t \leftarrow \mathcal{I}_{t-1}$,
    else
      $\mathcal{I}_t \leftarrow (\mathcal{I}_{t-1}\backslash\{i_k\}) \cup j^*$.
    end if
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{I}_t$.
    $i_k \leftarrow j^*$.
  end while
  $\mathcal{I}_{\text{opt}} = \arg\max_{\mathcal{I}\in\mathcal{S}} \sigma_{\max}(\mathbf{V}_{\mathcal{I}_{\text{opt}},:})$.
  return $\mathcal{I}_{\text{opt}}$ & principal left singular vector of $\mathbf{V}_{\mathcal{I}_{\text{opt}},:}$.
```

Fig. 7. The modified serial rank-2 algorithm.

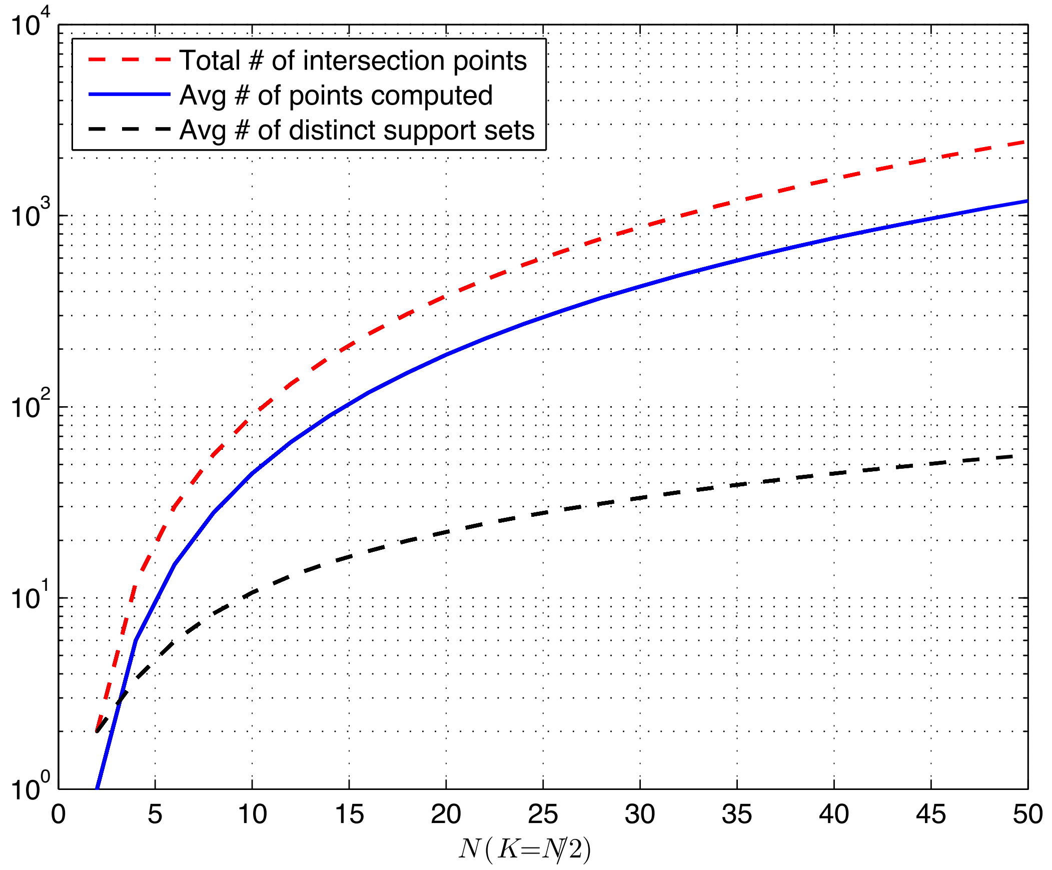

Fig. 8. The solid curve depicts the total number of intersection points computed by the modified serial rank-2 algorithm depicted in Fig. 7 as a function of N, for sparsity $K = N/2$. The number of points for a single value of N is averaged over 10^4 experiments with entries of $\mathbf{V} \in \mathbb{R}^{N\times 2}$ drawn from the standard normal distribution. The algorithm visits a subset of these points to keep track of the Kth order curve. The top dashed curve depicts the total number of intersection points, $2\binom{N}{2}$. The lower dashed curve depicts the average total number of distinct candidate support sets.