Sparse Approximation via Polynomial Equations

Matija Tomić, Raphaël Widdershoven, and Lieven De Lathauwer

***Abstract*—We consider the problem of finding sparse solutions of an underdetermined linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$. In contrast to conventional approaches based on greedy algorithms or convex relaxation, we reformulate sparse approximation as a structured system of polynomial equations and connect with the literature on tensor methods. We develop an eigenvalue decomposition (EVD) based method that formally guarantees recovery of all sparse solutions if there is more than one. We also develop two optimization-based methods achieving favorable computational complexity. The new methods allow explicit control of the target sparsity. Numerical experiments illustrate the performance and compare to basis pursuit (denoising) and orthogonal matching pursuit.**

***Index Terms*—sparse approximation, sparse recovery, tensor decompositions, multilinear algebra, polynomial systems**

I. Introduction

MANY fundamental problems in signal processing and applied mathematics can be formulated as sparse approximation problems. The wide applicability of this framework is demonstrated across diverse domains, such as compressed sensing [1], statistics [2], dictionary learning [3], [4], image processing [5], [6], [7], [8], and data-driven discovery of governing equations [9].

From the viewpoint of signal processing, the goal of sparse approximation is to represent a measured signal as a linear combination of a few elementary signals. Sparse approximation aims to find sparse solutions $\mathbf{x}$ to possibly underdetermined systems $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A} \in \mathbb{R}^{m\times n}$ with $m < n$. The sparsity level s denotes the number of nonzero entries in $\mathbf{x}$.

A brute-force search over all $\binom{n}{s}$ subsets of columns in $\mathbf{A}$ is computationally intractable for large n, and the problem is NP-hard [10], [11]. Practical algorithms, such as greedy methods (e.g., Orthogonal Matching Pursuit (OMP) [12], [13], [14]) and convex relaxations (e.g., Basis Pursuit (BP), Basis Pursuit Denoising (BPDN), [15]), offer approximate solutions with theoretical guarantees under specific conditions; see [5], [16] and references therein.

In this paper, we present an alternative approach based on solving systems of polynomial equations. We first formulate sparse approximation as a polynomial system (Section II) and explain how tensors represent and solve such systems (Section III). The sparse solutions appear as the roots of a polynomial system, all of which can be recovered simultaneously by the Macaulay construction (Section IV). Additionally, we propose two computationally tractable optimization-based alternatives, SESP-D and SESP-P, and analyze their complexity (Section V). Finally, we experimentally verify their performance and compare them to BP/BPDN and OMP (Section VI).

Notation: A tensor generalizes scalars, vectors, and matrices to higher orders [17], [18]. Scalars are denoted by lowercase letters (a), vectors by bold lowercase letters ($\mathbf{a}$), matrices by bold uppercase letters ($\mathbf{A}$), and tensors by calligraphic letters ($\mathcal{A}$). Matrix transpose, inverse, and pseudoinverse are written as $\cdot^\top$, $\cdot^{-1}$, and $\cdot^\dagger$, respectively. The i-th column of $\mathbf{A}$ is denoted by $\mathbf{a}_i$, thus $\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2\ \cdots]$. A diagonal matrix with $\mathbf{a}$ on the diagonal is written as $\mathrm{diag}(\mathbf{a})$. The $\mathrm{vec}(\cdot)$ operator vectorizes tensors. The tensor entry $a_{i_1,i_2,\ldots,i_N}$ of $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N}$ corresponds to the entry of $\mathrm{vec}(\mathcal{A})$ with index $(i_N-1)I_{N-1}\cdots I_1+(i_{N-1}-1)I_{N-2}\cdots I_1+\ldots+(i_2-1)I_1+i_1$. Vectors and matrices of all zeros and all ones are denoted by $\mathbf{0}$ and $\mathbf{1}$, respectively, with shape inferred from context. The outer and Kronecker products are denoted by $\otimes$ and $\otimes$, respectively. The outer product of n vectors $\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \ldots, \mathbf{a}^{(n)}$ is the n-th order tensor $\mathcal{T} = \mathbf{a}^{(1)}\otimes\mathbf{a}^{(2)}\otimes\cdots\otimes\mathbf{a}^{(n)}$, whose entries are defined as $t_{i_1 i_2\ldots i_n} = a^{(1)}_{i_1} a^{(2)}_{i_2}\cdots a^{(n)}_{i_n}$. The Kronecker product of $\mathbf{A} \in \mathbb{R}^{I\times K}$ and $\mathbf{B} \in \mathbb{R}^{J\times L}$ is defined as

$$\mathbf{A}\otimes\mathbf{B} \triangleq \begin{bmatrix} a_{1,1}\mathbf{B} & \cdots & a_{1,K}\mathbf{B} \\ \vdots & & \vdots \\ a_{I,1}\mathbf{B} & \cdots & a_{I,K}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{IJ\times KL}.$$

II. Sparse Approximation and Polynomials

An s-sparse vector has the property that any product of $s+1$ distinct entries is zero. For example, if $\mathbf{x}$ has at most $s = 2$ nonzeros, then $x_i x_j x_k = 0$ for all distinct i, j, k. There are $\binom{n}{s+1}$ such monomial equations corresponding to all combinations of $s+1$ indices. Sparse approximation now amounts to solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ together with the sparsity-inducing monomial equations.

Before developing methods, we consider the geometric interpretation for a toy example. Consider

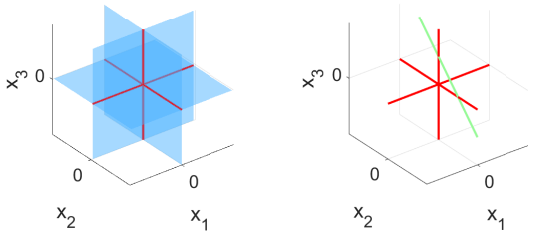
$$\underbrace{\begin{bmatrix} 1 & -1 & 1 \\ 0 & 1 & -1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} 1 \\ -1 \end{bmatrix}}_{\mathbf{b}}$$

with two 1-sparse solutions $\hat{\mathbf{x}}_1 = [0\ \ -1\ 0]^\top$ and $\hat{\mathbf{x}}_2 = [0\ 0\ 1]^\top$. The sparsity condition $s = 1$ gives three equations: $x_1x_2 = 0$, $x_2x_3 = 0$, and $x_1x_3 = 0$, overlapped in blue in Figure 1a. Their intersection, shown in red, characterizes vectors with at most one nonzero entry. More generally, for any n and sparsity level s, the solution set of the sparsity-inducing equations is a union of hyperplanes through the origin. This set contains all vectors $\mathbf{x}$ with sparsity at most s. Prior knowledge of the solution's sparsity is not strictly

M. Tomić was with the Department of Electrical Engineering (ESAT), KU Leuven, Leuven, Belgium. He is now with the Department of Mathematics, University of British Columbia, Vancouver, BC, Canada.

R. Widdershoven was with the Department of Electrical Engineering (ESAT), KU Leuven, Leuven, Belgium.

L. De Lathauwer is with the Department of Electrical Engineering (ESAT), KU Leuven, Leuven, Belgium, and also with Leuven.AI – KU Leuven institute for AI, B-3000 Leuven, Belgium.



(a) Sparsity-inducing equations and their intersection. (b) Sparse solutions as intersections with $\mathbf{Ax} = \mathbf{b}$.

Fig. 1: Solving the sparse approximation problem via systems of polynomial equations.

required, as this approach is also able to recover solutions that are sparser than the specified level s.

In our toy example, the sparsity-inducing equations define a set of vectors $\mathbf{x}$ with at most one nonzero entry, thus, the solution must lie on one of the red lines in Fig. 1. The solution must also satisfy the linear constraint $\mathbf{Ax} = \mathbf{b}$, which geometrically means finding the intersection points of the red lines with the green line representing $\mathbf{Ax} = \mathbf{b}$, as shown in Fig. 1b. The intersection points are the sparse solutions. This already highlights a key advantage of the polynomial formulation: it encodes *all* sparse solutions, i.e., the sparsest solution need not be unique.

III. Tensors and Polynomial Equations

A. Explicit Canonical Polyadic Decomposition

A rank-1 tensor of order N is the outer product of N nonzero vectors; see Notation in Section I. The rank of a tensor is the minimal number of rank-1 tensors required to represent it as a sum. This representation is called Canonical Polyadic Decomposition (CPD). Remarkably, the CPD is unique (up to permutation and scaling) under mild conditions [18, Section IV], unlike matrix rank decompositions, whose uniqueness typically requires conditions such as orthogonality.

Definition 1 (Canonical Polyadic Decomposition)**.** *The CPD of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_K}$ is given by*

$$\mathcal{A} = \sum_{r=1}^{R} \mathbf{u}_r^{(1)} \otimes \mathbf{u}_r^{(2)} \otimes \cdots \otimes \mathbf{u}_r^{(K)} \triangleq [\![\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(K)}]\!],$$

where R is the tensor rank, and $\mathbf{u}_r^{(k)} \in \mathbb{R}^{I_k}$ are the factor vectors for each mode k.

Two main approaches for the computation of CPD exist: *algebraic* methods and *optimization-based* methods. Algebraic methods, such as those based on the generalized EVD [18], are exact in the noiseless setting. In the presence of noise, the approximate solution of the algebraic algorithm can be used as an initial guess, which can be refined using optimization-based approaches. In this work, we opt to minimize a nonlinear least squares (NLS) objective to compute the CPD [19]. Specifically, given a target rank R, we fit the rank-R CPD model $[\![\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(K)}]\!]$ with factor matrices $\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(K)}$ to the data tensor $\mathcal{T}$ by solving

$$\min_{\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(K)}} f = \|[\![\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(K)}]\!] - \mathcal{T}\|_F^2. \quad (1)$$

Gauss–Newton (GN) algorithm is a possible choice to minimize the NLS objective function. We employ an existing optimization framework built into Tensorlab [20]. Only the key components need to be provided to the algorithm: the objective function, its gradient, and the Gramian of the Jacobian [19]. When paired with trust-region strategies, GN is globally convergent under mild conditions [21].

B. Implicit CPD

In the above, the tensor entries were explicitly given. However, sometimes the entries are only known implicitly. An example is the LS-CPD framework [22], where the tensor appears as the solution to a linear system subject to a CPD constraint:

$$\mathbf{Cy} = \mathbf{d}, \quad \mathbf{y} = \text{vec}(\mathcal{Y}), \quad \mathcal{Y} = [\![\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}]\!], \quad (2)$$

where $\mathbf{C} \in \mathbb{R}^{M \times K}$ and $\mathbf{d} \in \mathbb{R}^{M}$. $\mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is a rank-R tensor with factor matrices $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R}$. If the linear system is full rank, solving it yields a vectorized tensor of which the CPD can subsequently be computed. If the system is underdetermined, a CPD constrained solution can be obtained by minimizing the NLS objective

$$\min_{\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}} f = \|\mathbf{C}\,\text{vec}([\![\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}]\!]) - \mathbf{d}\|_F^2.$$

The computation is analogous to the explicit CPD case and requires expressions for the objective function, its gradient and the Gramian of the Jacobian.

C. Monomials and Rank-1 Tensors

The sparsity-promoting monomials $x_{i_1} \cdots x_{i_{s+1}} = 0$ with $i_1 \neq \cdots \neq i_{s+1}$ appear in the symmetric rank-1 tensor

$$\mathcal{X} = \underbrace{\mathbf{x} \otimes \cdots \otimes \mathbf{x}}_{s+1 \text{ times}}.$$

These monomials can be collected into a structured multilinear system of the form (2) with rank $R = 1$:

$$\mathbf{Z}\,\text{vec}(\mathcal{X}) = \mathbf{0}. \quad (3)$$

The matrix $\mathbf{Z}$ is defined as a selection matrix whose rows select the multilinear terms $x_{i_1} \cdots x_{i_{s+1}}$, with $i_1 \neq \cdots \neq i_{s+1}$ appearing in $\text{vec}(\mathcal{X}) = \otimes_{k=1}^{s+1} \mathbf{x}$. Equation (3) enforces all of these selected terms to be equal to zero.

We extend this LS-CPD system with the data fitting term $\mathbf{Ax} = \mathbf{b}$, leading to the formulation

$$\begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{Z} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \otimes_{i=1}^{s+1} \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix}. \quad (4)$$

The computation can follow the principles outlined above. In Section V, we work out two efficient computational variants.

IV. Polynomial Equations and Multiple Solutions

Formulation (4) describes a single solution $\mathbf{x}$ to a particular system of polynomial equations. We now show that all solutions of a polynomial system are described jointly by the null space of a matrix holding the polynomial coefficients.

A. Roots as Structured Null Space Vectors

Consider a system of p multivariate polynomials in n complex variables

$$\begin{cases} f_1(x_1, \ldots, x_n) = 0 \\ \qquad \vdots \\ f_p(x_1, \ldots, x_n) = 0 \end{cases} \tag{5}$$

with degrees $\deg(f_i) = d_i$ and maximal degree $d_0 = \max_i d_i$. The goal is to find the common roots $\mathbf{x}^{(q)} = [x_1^{(q)}, \ldots, x_n^{(q)}]^\top \in \mathbb{C}^n$, or, in the noisy case, the best fitting roots. We assume that (5) has Q distinct roots.

For each root $\mathbf{x}^{(q)}$, the vector of all distinct monomials of total degree up to d evaluated at this root is the multivariate Vandermonde vector

$$\mathbf{v}_q(d) = \begin{bmatrix} 1 & x_1^{(q)} & x_2^{(q)} & x_1^{(q)}x_2^{(q)} & \ldots & x_n^{(q)d} \end{bmatrix}^\top, \tag{6}$$

of length $N_d = \binom{n+d}{d}$. Every polynomial in (5) is linear in these monomials, so its coefficients can be collected in a matrix $\mathbf{M}_0 \in \mathbb{C}^{p \times N_{d_0}}$ such that

$$\mathbf{M}_0 \mathbf{v}_q(d_0) = \mathbf{0}, \qquad q = 1, \ldots, Q. \tag{7}$$

Stacking all Vandermonde vectors into the multivariate Vandermonde matrix $\mathbf{V}(d_0) = [\, \mathbf{v}_1(d_0) \ \cdots \ \mathbf{v}_Q(d_0) \,]$ gives

$$\mathbf{M}_0 \mathbf{V}(d_0) = \mathbf{0}. \tag{8}$$

With $d_0 = s + 1$, equation (4) can be expressed in the form (7) by dropping repeated monomials. Specifically, $\mathbf{b}$ has moved to the left-hand side, where it multiplies the leading entry 1 of $\mathbf{v}_q(d_0)$, and $\otimes_{i=1}^{d_0} \mathbf{x}$ is replaced by a vector $\mathbf{v}_q(d_0)$ holding a single copy of each monomial. The entries of $\mathbf{v}_q(d_0)$ are the distinct entries of the symmetric rank-1 tensor

$$\underbrace{\mathbf{x}_h^{(q)} \otimes \cdots \otimes \mathbf{x}_h^{(q)}}_{d_0 \text{ times}}, \quad \text{with} \quad \mathbf{x}_h^{(q)} = [1, x_1^{(q)}, \ldots, x_n^{(q)}]^\top. \tag{9}$$

Thus, equation (8) states that the null space of $\mathbf{M}_0$ contains Q such rank-1 tensors, one per root. Solving a polynomial system is in this sense a subspace variant of (4), and the task is to separate the Q rank-1 terms that live in the null space.

B. Recovering the Roots from the Null Space

Assume for now that the Q Vandermonde vectors are linearly independent and that

$$\dim \ker(\mathbf{M}_0) = Q, \tag{10}$$

so that the columns of $\mathbf{V}(d_0)$ form a basis of the null space of $\mathbf{M}_0$; the general case is treated in Section IV-C. We write $\mathbf{V}(d)$ with $d = d_0$ in this subsection.

In practice $\mathbf{V}(d)$ is not available, since it depends on the unknown roots. However, it can be obtained from a numerical (orthonormal) basis $\mathbf{K}(d) = [\mathbf{k}_1 \ \cdots \ \mathbf{k}_Q]$ for the null space of $\mathbf{M}_0$, obtained from its singular value decomposition. As $\mathbf{K}(d)$ and $\mathbf{V}(d)$ span the same subspace, they are related through an unknown nonsingular matrix $\mathbf{W} \in \mathbb{C}^{Q \times Q}$:

$$\mathbf{K}(d) = \mathbf{V}(d)\mathbf{W}^\top. \tag{11}$$

Every basis vector $\mathbf{k}_j = \sum_q w_{jq} \mathbf{v}_q(d)$ is a mixture of the same Q symmetric rank-1 tensors (9), and only the coefficients w_{jq} differ. Collecting these coefficients in an additional mode gives an explicit CPD whose rank-1 terms encode the roots. Indeed, with $\mathbf{X}_h = [\, \mathbf{x}_h^{(1)} \ \cdots \ \mathbf{x}_h^{(Q)} \,]$, the tensor $\mathcal{K} = [\![\mathbf{X}_h, \ \ldots, \ \mathbf{X}_h, \ \mathbf{W}]\!]$ can be obtained from $\mathbf{K}(d)$ by reshaping. The CPD is formally guaranteed to return all Q roots of (5) at once. To explain this, let us consider a further reshaping of $\mathcal{K}$ into a third-order tensor $\mathcal{Y} \in \mathbb{C}^{N_{d-1} \times (n+1) \times Q}$. It is easy to show that this tensor admits the CPD

$$\mathcal{Y} = [\![\mathbf{V}(d-1), \mathbf{V}(1), \mathbf{W}]\!], \tag{12}$$

whose third factor matrix contains the roots:

$$\mathbf{V}(1) = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(Q)} \\ \vdots & \vdots & & \vdots \\ x_n^{(1)} & x_n^{(2)} & \cdots & x_n^{(Q)} \end{bmatrix}. \tag{13}$$

Under (10), $\mathbf{V}(d-1)$ and $\mathbf{W}$ have full column rank Q, and the columns of $\mathbf{V}(1)$ are pairwise non-collinear because the Q toors are distinct. This guarantees the uniqueness of (12); moreover, the factors can be computed by means of a simple matrix EVD, see [18, Section IV] and references therein.

C. The Macaulay Construction

For a general polynomial system, assumption (10) fails. If $\dim \ker(\mathbf{M}_0) > Q$, the null space contains additional directions that do not correspond to any root, so the columns of $\mathbf{V}(d_0)$ no longer form a basis, and (11) does not hold.

The spurious directions can be removed by adding rows that every root still satisfies. Since $f_i(\mathbf{x}^{(q)}) = 0$ implies $\mathbf{x}^{(q)\boldsymbol{\alpha}} f_i(\mathbf{x}^{(q)}) = 0$ for every monomial $\mathbf{x}^{\boldsymbol{\alpha}}$, the shifted polynomials $\mathbf{x}^{\boldsymbol{\alpha}} f_i$ vanish at all Q roots, so their coefficients give extra rows that constrain $\mathbf{v}_q(d)$ without excluding any root.

For a degree $d \geq d_0$, the Macaulay matrix $\mathbf{M}(d)$ collects, in its rows, the coefficients of all shifted polynomials $\mathbf{x}^{\boldsymbol{\alpha}} f_i$ with $\deg(\mathbf{x}^{\boldsymbol{\alpha}}) \in \{0, 1, \ldots, d - d_i\}$, expressed in the monomials of $\mathbf{v}_q(d)$ [23]. Its rows include those of $\mathbf{M}_0$, obtained for $\boldsymbol{\alpha} = \mathbf{0}$, and every root still satisfies

$$\mathbf{M}(d)\mathbf{v}_q(d) = \mathbf{0}, \qquad \mathbf{M}(d)\mathbf{V}(d) = \mathbf{0}. \tag{14}$$

The structured vectors of interest therefore remain in the null space at every degree, while the added rows keep removing directions that do not correspond to roots.

If the system has a finite number of roots, the dimension of the null space of $\mathbf{M}(d)$ stabilizes for $d \geq d^*$. The so-called degree of regularity d^* is the lowest degree for which the null space is spanned only by vectors corresponding to the roots [23], [24], [25], that is,

$$\ker(\mathbf{M}(d)) = \operatorname{span}\{\mathbf{v}_1(d), \ldots, \mathbf{v}_Q(d)\}, \qquad d \geq d^*. \tag{15}$$

At the degree of regularity the situation of Section IV-B is restored, with $\mathbf{M}(d^*)$ in the role of $\mathbf{M}_0$, and the roots follow from a variant of (12) constructed for $d \geq d^*$. In practice neither Q nor d^* is known beforehand, so d is increased until the dimension of the null space of $\mathbf{M}(d)$ stabilizes, and that dimension is taken as the number of roots.

For sparse approximation, the polynomial system consists of $\mathbf{Ax} = \mathbf{b}$ and the $\binom{n}{s+1}$ sparsity-inducing equations. The latter turn the infinitely many solutions of the underdetermined $\mathbf{Ax} = \mathbf{b}$ into a finite set: if every s columns of $\mathbf{A}$ are linearly independent, each support of size at most s carries at most one solution, so there are at most $\binom{n}{s}$ roots [16]. Computing $\mathbf{K}(d)$ at the degree of regularity and decomposing (12) therefore returns *all* of them at once in $\mathbf{V}(1)$. Solutions that are sparser than s are included as well. The limitation is the computational cost. The Macaulay matrix has $N_d = \binom{n+d}{d}$ columns and a comparable number of rows, so computing $\mathbf{K}(d)$ may be expensive. As a speed-up, the columns of $\mathbf{M}(d)$ corresponding to monomials that need to be zero, can be dropped, and the null space of the resulting smaller matrix computed.

V. Efficient Optimization-Based Algorithms

In LS-CPD problems, it is generally advantageous to exploit the structure of the system matrix, see [26] and references therein. This is also true in the present setting. The matrix $\mathbf{Z}$ in our core equation (4) is a large matrix of size $(m + n(n-1)\cdots(n-s)) \times (n + n^{s+1})$; however, as a selection matrix it is just a part of an identity matrix. Exploiting this fact, the GN quantities for the NLS optimization can be expressed in terms of elementary symmetric polynomials (ESPs). The latter can be evaluated cheaply, leading to computationally efficient implementations; see Section V-B.

Definition 2 (Elementary Symmetric Polynomial [27])**.** *ESP of degree d in n variables is obtained by summing together all monomials of degree d in n variables:*

$$e_d(z_1, \ldots, z_n) = \sum_{1 \leq i_1 < \cdots < i_d \leq n} z_{i_1} \cdots z_{i_d},$$

with $e_0(z_1, \ldots, z_n) = 1$ and $e_d(z_1, \ldots, z_n) = 0$ for $d > n$.

In particular, we are interested in the following ESPs:

$$\begin{aligned} E_{s+1} &:= e_{s+1}(x_1^2, \ldots, x_n^2), \\ E_s^{(i)} &:= e_s(x_1^2, \ldots, x_{i-1}^2, x_{i+1}^2, \ldots, x_n^2), \\ E_{s-1}^{(i,j)} &:= e_{s-1}(x_1^2, \ldots, x_{i-1}^2, x_{i+1}^2, \ldots, x_{j-1}^2, x_{j+1}^2, \ldots, x_n^2). \end{aligned} \tag{16}$$

A. Algorithms

We propose two algorithms based on the formulation in (4). The first, SESP-D, joinlty optimizes data fitting and sparsity promotion. The second, SESP-P, seeks the sparsest solution that exactly fits the data. Both algorithms return a single solution for a given initialization; when multiple sparse solutions exist, different solutions may be obtained by reinitializing.

Sparsification via ESP - Direct: SESP-D applies the NLS approach to:

$$\begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \lambda \mathbf{Z} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \otimes_{i=1}^{s+1} \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix}. \tag{17}$$

The scalar λ is introduced to balance data fitting and sparsity promotion. By exploiting the structure of $\mathbf{Z}$ (Appendix A), the following expressions for the objective function, gradient, and Jacobian Gramian are derived in Appendix B:

$$\begin{aligned} f &= \tfrac{1}{2}\|\mathbf{Ax} - \mathbf{b}\|_2^2 + \tfrac{1}{2}\lambda^2 (s+1)! E_{s+1} \\ \mathbf{J}^\top \mathbf{r} &= \mathbf{A}^\top(\mathbf{Ax} - \mathbf{b}) + \lambda^2 (s+1)! \sum_{i=1}^{n} E_s^{(i)} x_i \mathbf{e}_i \\ \mathbf{J}^\top \mathbf{J} &= \mathbf{A}^\top \mathbf{A} + \lambda^2 (s+1)! \Big(\sum_{i=1}^{n} E_s^{(i)} \mathbf{e}_i \mathbf{e}_i^\top \\ &\quad + \sum_{\substack{i,j=1 \\ i \neq j}}^{n} E_{s-1}^{(i,j)} x_i x_j \mathbf{e}_i \mathbf{e}_j^\top \Big), \end{aligned} \tag{18}$$

Sparsification via ESP - Projected: The linear system $\mathbf{Ax} = \mathbf{b}$ is first solved to obtain a particular solution $\mathbf{x}_p$. The null space $\mathbf{V}$ of $\mathbf{A}$ is then computed, yielding the parametrization $\mathbf{x} = \mathbf{x}_p + \mathbf{Vy}$. The goal is to determine $\mathbf{y}$ such that the resulting solution $\mathbf{x}$ is sparse. Sparsity is enforced through the matrix $\mathbf{Z}$, leading to

$$\mathbf{Z} \operatorname{vec}(\mathcal{X}) = \mathbf{0}, \qquad \operatorname{vec}(\mathcal{X}) = \bigotimes_{i=1}^{s+1} (\mathbf{x}_p + \mathbf{Vy}).$$

The resulting NLS problem is solved using GN algorithm.

The optimization variable is $\mathbf{y}$, with the solution recovered through $\mathbf{x} = \mathbf{x}_p + \mathbf{Vy}$. For notational convenience, however, the objective function, gradient, and Jacobian Gramian are expressed in terms of $\mathbf{x}$, which can be readily computed from $\mathbf{y}$ at each iteration. Appendix C derives the expressions

$$\begin{aligned} f &= \tfrac{1}{2}(s+1)! E_{s+1} \\ \mathbf{J}^\top \mathbf{r} &= (s+1)! \mathbf{V}^\top \sum_{i=1}^{n} E_s^{(i)} x_i \mathbf{e}_i \\ \mathbf{J}^\top \mathbf{J} &= (s+1)! \mathbf{V}^\top \Big(\sum_{i=1}^{n} E_s^{(i)} \mathbf{e}_i \mathbf{e}_i^\top + \sum_{\substack{i,j=1 \\ i \neq j}}^{n} E_{s-1}^{(i,j)} x_i x_j \mathbf{e}_i \mathbf{e}_j^\top \Big) \mathbf{V}. \end{aligned} \tag{19}$$

B. Computational Complexity of the LS-CPD-Based Methods

ESPs appearing in (18) and (19) involve sums with up to $\binom{n}{s+1}$ terms, and their naive evaluation requires $\mathcal{O}(n^{s+1})$ operations. However, ESPs have a structure which we exploit to reduce the complexity. The proof of Lemma 1 (Appendix D) establishes a recursion which, when applied, evaluates the objective functions of SESP-D and SESP-P in $\mathcal{O}(ns)$ time.

Lemma 1. *$e_d(z_1, \ldots, z_n)$ can be computed in $O(nd)$ time.*

Applying the same recursion directly to evaluate $\mathbf{J}^\top \mathbf{r}$ and $\mathbf{J}^\top \mathbf{J}$ would require $\mathcal{O}(n^2 s)$ and $\mathcal{O}(n^3 s)$ operations, respectively. Lemma 2 and Lemma 3 reduce these complexities to $\mathcal{O}(ns)$ and $\mathcal{O}(n^2 s)$, respectively. The proofs in Appendices E and F are constructive.

Lemma 2. *Let $i = 1, \ldots, n$ and $s \leq n$. All n values $E_s^{(i)}$ can be computed simultaneously in $O(ns)$ time.*

Lemma 3. *Let $1 \leq i < j \leq n$ and $s \leq n$. All $\binom{n}{2}$ values $E_{s-1}^{(i,j)}$ can be computed simultaneously in $O(n^2 s)$ time.*

Table I reports per-iteration complexities of the GN algorithm, excluding iteration-independent quantities such as $\mathbf{V}$ and $\mathbf{A}^\top\mathbf{A}$, which are precomputed once. The remaining terms arise as follows: ns and n^2s from evaluating the ESPs; nm from computing the residual $\mathbf{Ax}-\mathbf{b}$ in f and the product $\mathbf{A}^\top(\mathbf{Ax}-\mathbf{b})$ in $\mathbf{J}^\top\mathbf{r}$; $n(n-m)$ from the substitution $\mathbf{x}=\mathbf{x}_p+\mathbf{Vy}$ and the multiplication by $\mathbf{V}^\top$ in $\mathbf{J}^\top\mathbf{r}$; and $n^2(n-m)$ from the multiplications by $\mathbf{V}^\top$ and $\mathbf{V}$ in $\mathbf{J}^\top\mathbf{J}$.

TABLE I: Per-iteration computational complexities of SESP-D and SESP-P

	Calls per Iteration	SESP-D	SESP-P
f	$1+it_{TR}$	$nm+ns$	$n(n-m)+ns$
$\mathbf{J}^\top\mathbf{r}$	1	$nm+ns$	$n(n-m)+ns$
$\mathbf{J}^\top\mathbf{J}$	1	n^2s	$n^2(n-m)+n^2s$

Since the dogleg trust-region GN method is used, the objective f is evaluated $1+it_{\mathrm{TR}}$ times per iteration, where it_{TR} denotes the number of trust-region sub-iterations. At each iteration, the descent step $\mathbf{p}$ is obtained by solving $\mathbf{J}^\top\mathbf{J}\,\mathbf{p}=\mathbf{J}^\top\mathbf{r}$ via the pseudoinverse,

$$\mathbf{p}=(\mathbf{J}^\top\mathbf{J})^\dagger(\mathbf{J}^\top\mathbf{r}),$$

which has computational complexity $\mathcal{O}(n^3)$ [28].

In Table I, the terms ns and n^2s correspond to enforcing sparsity, while nm, $n(n-m)$, and $n^2(n-m)$ correspond to enforcing $\mathbf{Ax}=\mathbf{b}$. Since $s\ll m$, the sparsity-enforcing terms do not dominate the overall complexity, confirming that polynomials can be used to design computationally efficient sparse approximation algorithms.

VI. Experiments

This section compares the proposed methods with BP/BPDN and OMP. Section VI-A describes the MATLAB implementations and stopping criteria. Sections VI-B, VI-C, and VI-D detail the experimental setup, initialization strategies, and the choice of regularization parameter, respectively. Recovery accuracy and average runtime are evaluated for both the noiseless (Section VI-E) and noisy (Section VI-F) settings. The algebraic framework allows explicit control over the target sparsity s and enables the recovery of multiple valid sparse representations. This is illustrated in Section VI-G. All computations were performed in MATLAB R2024b on a Lenovo ThinkPad T480s equipped with an Intel® Core™ i5-8350U CPU @ 1.70 GHz and 8 GB of RAM.

A. MATLAB Implementation

1) SESP-D and SESP-P: The Tensorlab [20] implementation of the GN-NLS dogleg trust-region algorithm is used for both SESP-D and SESP-P. The function and step tolerances are ε^2 and ε, respectively, where ε denotes machine precision. A maximum of 2000 iterations is used.

At each iteration, the following stopping criterion based on support-restricted least squares is evaluated. Let $\mathbf{x}_k$ denote the current iterate, and let $S_k=\mathrm{supp}_s(\mathbf{x}_k)$ denote the index set of its s largest-magnitude entries, where s is the target sparsity supplied to the algorithm. Define the support-restricted least-squares solution $\mathbf{z}\in\mathbb{R}^n$ by

$$z_j=\begin{cases}\left(\mathbf{A}_{S_k}^\dagger\mathbf{b}\right)_j & j\in S_k,\\ 0 & j\notin S_k,\end{cases}$$

where $\mathbf{A}_{S_k}\in\mathbb{R}^{m\times s}$ is the submatrix of $\mathbf{A}$ formed by retaining only the columns indexed by S_k. The algorithm terminates when $\|\mathbf{Az}-\mathbf{b}\|_2\le\tau$, where $\tau=10^{-10}$ in the noiseless case and $\tau=\sigma\sqrt{m}$ in the noisy case. The latter threshold matches the expectation $\mathbb{E}[\|\mathbf{n}\|_2]=\sigma\sqrt{m}$ for additive Gaussian noise $\mathbf{n}\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I}_m)$. The algorithms return $\hat{\mathbf{x}}=\mathbf{z}$ as the computed solution to the sparse approximation problem.

2) BP/BPDN: In the noiseless case, the BP problem

$$\min_{\mathbf{x}}\|\mathbf{x}\|_1\quad\text{subject to}\quad\mathbf{Ax}=\mathbf{b}$$

is solved following [15] using MATLAB's `linprog` interior-point solver with optimality tolerance 10^{-8}, constraint tolerance 10^{-6}, and a maximum of 2000 iterations. In the noisy setting, where σ denotes the standard deviation of the additive noise, the BPDN problem

$$\min_{\mathbf{x}}\|\mathbf{x}\|_1\quad\text{subject to}\quad\|\mathbf{Ax}-\mathbf{b}\|_2\le\sigma\sqrt{m}$$

is solved using SPGL1 [29], [30].

3) OMP: OMP [31] greedily selects at each iteration the column of $\mathbf{A}$ most correlated with the current residual, incrementally building a sparse approximation of $\mathbf{b}$. The algorithm terminates when the residual norm falls below a prescribed threshold: $\|\mathbf{Ax}-\mathbf{b}\|_2\le10^{-10}$ in the noiseless case, and $\|\mathbf{Ax}-\mathbf{b}\|_2\le\sigma\sqrt{m}$ in the noisy case.

B. General Experimental Setup

The matrix $\mathbf{A}\in\mathbb{R}^{m\times n}$ has i.i.d. entries drawn from the standard normal distribution, with columns subsequently normalized to unit ℓ_2 norm. Unless otherwise specified, $m=32$ and $n=64$. The vector $\mathbf{x}_{\mathrm{true}}\in\mathbb{R}^n$ has s nonzero entries, with support chosen uniformly at random and nonzero values drawn independently from the standard normal distribution.

In the noiseless setting, the measurement vector $\mathbf{b}$ is computed as $\mathbf{b}=\mathbf{Ax}_{\mathrm{true}}$. A computed solution $\hat{\mathbf{x}}$ is declared successful if $\|\hat{\mathbf{x}}-\mathbf{x}_{\mathrm{true}}\|_2/\|\mathbf{x}_{\mathrm{true}}\|_2<10^{-5}$. The success rate is defined as the ratio of successful recoveries over 100 trials.

In the noisy setting, the measurement vector is given by $\mathbf{b}=\mathbf{Ax}_{\mathrm{true}}+\mathbf{n}$, with additive Gaussian noise $\mathbf{n}\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})$. The signal-to-noise ratio is defined as

$$\mathrm{SNR}=10\log_{10}\left(\frac{\|\mathbf{Ax}_{\mathrm{true}}\|_2^2/m}{\sigma^2}\right).$$

The relative error $\|\hat{\mathbf{x}}-\mathbf{x}_{\mathrm{true}}\|_2/\|\mathbf{x}_{\mathrm{true}}\|_2$ is reported instead of the success rate. All algorithms are evaluated on the same randomly generated problem instance, i.e., the same $\mathbf{A}$, $\mathbf{x}_{\mathrm{true}}$, and $\mathbf{b}$ within each trial. The sparsity level s is assumed known and provided to the algorithms. For SESP-D, λ is fixed at 10^{-4}; Section VI-D motivates this choice. For SESP-P, we use the particular solution $\mathbf{x}_p$ satisfying $\mathbf{Ax}_p=\mathbf{b}$ with noiseless or noisy measurements $\mathbf{b}$, depending on the setting.

C. Initialization

SESP-D and SESP-P solve nonconvex optimization problems, and the GN solver is a local method. The choice of initialization may therefore affect the outcome. We briefly discuss the initialization strategy used for each method.

1) SESP-D: Let $\mathbf{x}_0 = \alpha \mathbf{z}$, where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n)$ and $\alpha > 0$ is a scaling parameter to be determined. To ensure that the data-fitting term $f_1 = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$ and the sparsity-promoting term $f_2 = \frac{1}{2}(s+1)!\,\lambda\, E_{s+1}(\mathbf{x}^{\cdot 2})$ in (18) contribute equally at initialization, we choose α such that their expected values are equal. A direct calculation yields

$$\mathbb{E}[f_1] = \frac{\alpha^2 n + s}{2}, \qquad \mathbb{E}[f_2] = \alpha^{2(s+1)} \prod_{j=0}^{s} (n-j),$$

where the expectations are taken over $\mathbf{z}$. Solving $\mathbb{E}[f_1] = \mathbb{E}[f_2]$ for α yields the initialization scale. Fig. 2 shows the effect of increasing the number of random restarts on success rate: a single initialization suffices at low sparsity, while for higher sparsity reinitializations may improve the result.

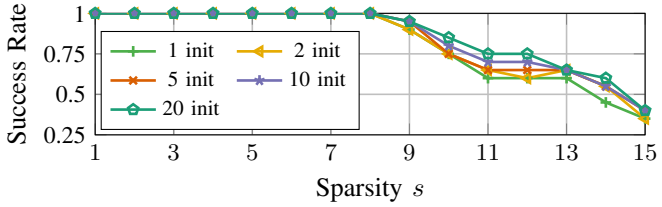


Fig. 2: Success rate versus sparsity for SESP-D with different numbers of random initializations.

2) SESP-P: The SESP-P algorithm minimizes $E_{s+1}(\mathbf{x}^{\cdot 2})$ over the affine solution set via GN iterations on the null-space coordinate $\mathbf{y}$, where $\mathbf{x} = \mathbf{x}_p + \mathbf{V}\mathbf{y}$. Because the objective $E_{s+1}(\mathbf{x}_0^{\cdot 2})$ increases with the magnitude of each entry of $\mathbf{x}_0$, we initialize by minimizing $\|\mathbf{x}_p + \mathbf{V}\mathbf{y}_0\|_2$, which gives $\mathbf{y}_0 = -\mathbf{V}^\dagger \mathbf{x}_p$. Since the matrix $\mathbf{V}$ contains an orthonormal basis for the null space of $\mathbf{A}$, this simplifies to $\mathbf{y}_0 = -\mathbf{V}^\top \mathbf{x}_p$.

3) Warm Start from BP/BPDN: The BP/BPDN solution $\mathbf{x}_{\mathrm{BP/BPDN}}$ can be used to warm-start SESP-D and SESP-P. For SESP-D, we set $\mathbf{x}_0 = \mathbf{x}_{\mathrm{BP/BPDN}}$. For SESP-P, we set $\mathbf{y}_0 = \mathbf{V}^\top(\mathbf{x}_{\mathrm{BP/BPDN}} - \mathbf{x}_p)$. This strategy provides a starting point that is optimal in the ℓ_1 sense.

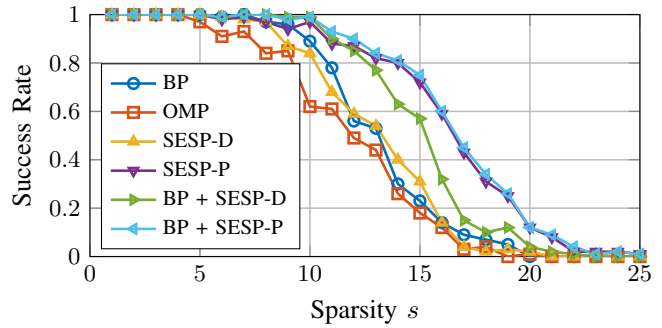


Fig. 3: Success rate versus sparsity over 100 trials in the noiseless setting. SESP-D uses 1 random initialization.

The recovery performance for different initialization strategies in the noiseless setting is reported in Fig. 3, alongside BP and OMP for reference. All methods achieve perfect recovery at low sparsity; as sparsity increases, SESP-D performs similarly to BP, while SESP-P outperforms it, showing that the polynomial-based formulation can surpass ℓ_1-based recovery on its own. Warm-starting SESP-D and SESP-P from the BP solution (BP + ELS and BP + Null) further improves performance beyond BP alone. Since BP finds the ℓ_1 optimal solution to $\mathbf{A}\mathbf{x} = \mathbf{b}$ instead of the ℓ_0, our polynomial-based refinement further sparsify that solution.

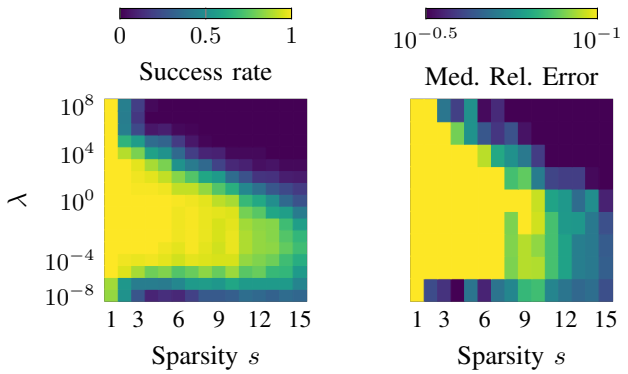
D. Regularization Parameter

The SESP-D formulation (17) includes the regularization parameter λ, which controls the relative weight of the sparsity-promoting term in the objective.

In the noiseless setting, the success rate is insensitive to the choice of λ over several orders of magnitude for low-to-medium sparsity. At higher sparsity, the choice of λ is not very critical either, but quite small values of λ tend to be preferable. They place relatively more weight on the data-fitting term compared to the sparsity-promoting term. Fig. 4a shows the success rate as a function of both s and λ; for each (λ, s) pair, 5 random initializations are used across 20 trials, and a trial is declared successful if at least one initialization succeeds. Yellow regions indicate successful recovery, while dark blue regions indicate failure across all 5 initializations.

In the noisy setting, λ has a similar effect. Fig. 4b shows the median error over 20 trials as a function of s and λ, for additive Gaussian noise with SNR $= 20$ dB. A recovery with relative error below 10^{-1} is considered successful at this SNR.

Based on Fig. 4, $\lambda = 10^{-4}$ is used for all experiments.



(a) Noiseless setting. (b) Noisy setting, SNR $= 20$ dB.

Fig. 4: Recovery performance of SESP-D as a function of sparsity level s and regularization parameter λ.

E. Recovery and Runtime in the Noiseless Case

Recovery performance of warm-started SESP-D and SESP-P in the noiseless case is compared to BP and OMP in Figs. 5 and 6. Fig. 5 displays the success rate as a function of the number of measurements m for a fixed sparsity level s over 100 trials. SESP-D and SESP-P require fewer measurements than BP and OMP to successfully recover an s-sparse signal.

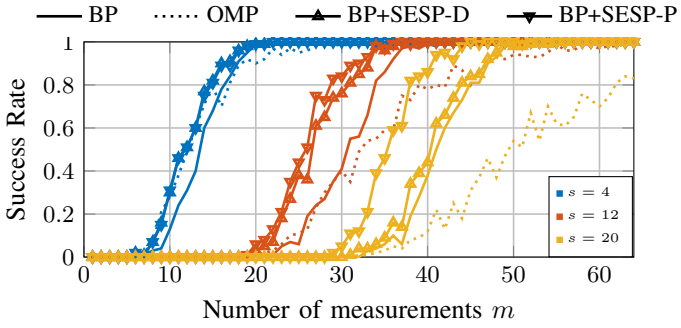


Fig. 5: Effect of measurement count on recovery.

Fig. 6 displays the success rate as a function of the sparsity level s for a fixed number of measurements m over 100 trials. With $m = 32$ measurements, SESP-D and SESP-P recover a 12-sparse signal in approximately 90% of trials, compared to roughly 50% for BP and OMP.

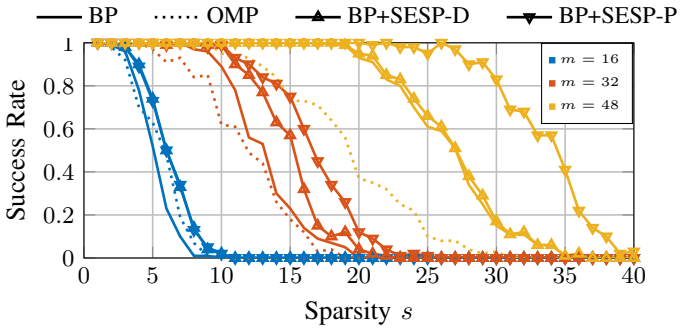
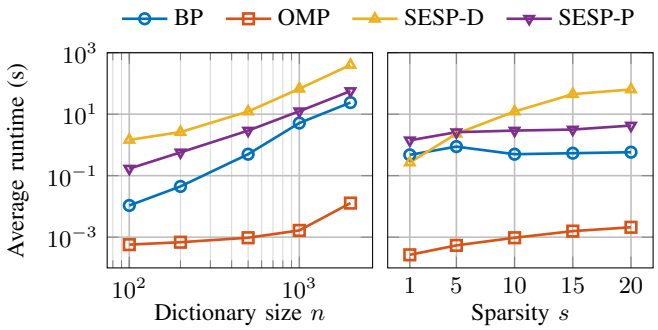


Fig. 6: Recovery with a fixed measurement count.

Average runtime as a function of matrix size n or sparsity level s is reported in Fig. 7, with all timings measured using MATLAB timing functions and averaged over 20 trials. Since BP achieves perfect recovery for all n and s considered, warm-starting SESP-D and SESP-P from the BP solution would be redundant here and is omitted.



(a) Fixed sparsity $s = 10$. (b) Fixed size $n = 500$.

Fig. 7: Average runtime comparison in the noiseless case.

Fig. 7a shows that BP, SESP-D, and SESP-P scale similarly with matrix dimension at fixed sparsity. Fig. 7b shows that SESP-D runtime grows with sparsity for fixed n, while BP and SESP-P remain largely unaffected. The per-iteration cost of SESP-D is comparable to that of BP and SESP-P; the increase instead reflects more iterations needed to converge at higher sparsity levels.

F. Recovery and Runtime in the Noisy Case

Fig. 8 reports recovery performance under additive Gaussian noise, with 100 independent trials per SNR level and error bars indicating the 25th and 75th percentiles. The oracle estimator, which solves the least-squares problem restricted to the true support S, lower-bounds the achievable error for any method that must identify the support. Since SESP-D and SESP-P estimate the support before applying a support-restricted least-squares correction (Section VI-A), the same debiasing step is applied to BPDN and OMP for a fair comparison.

Fig. 8 shows that, unlike BP and OMP, SESP-D and SESP-P remain close to the oracle bound, demonstrating accurate support recovery even at higher sparsity levels. Initializing SESP-D and SESP-P with the BPDN solution yields essentially the same reconstruction accuracy as the other initialization strategies in Section VI-A.

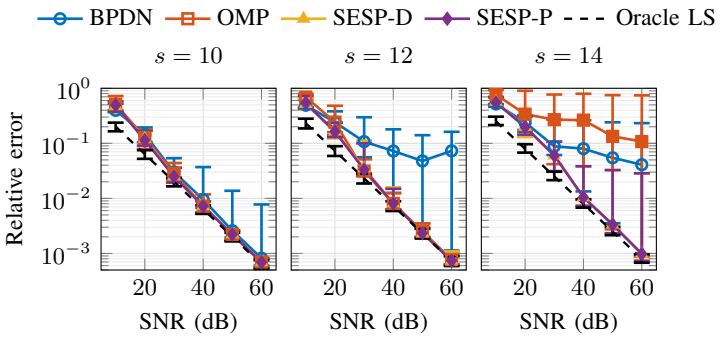
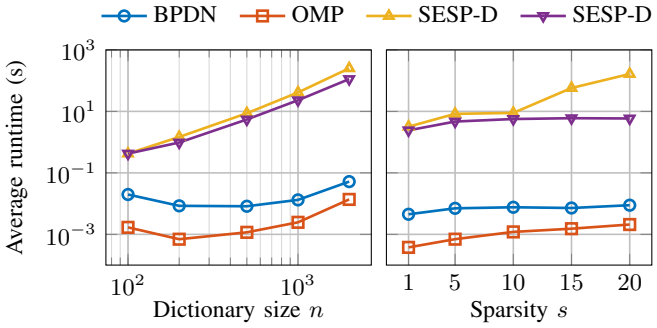


Fig. 8: Median relative error versus SNR. Error bars indicate the 25th and 75th percentiles over 100 trials. The oracle lower bound corresponds to least-squares on the true support.

The improved robustness to measurement noise comes at increased computational cost: Fig. 9 shows SESP-D and SESP-P requiring longer runtimes than BPDN and OMP, with a larger gap than in the noiseless case (Fig. 7).



(a) Fixed sparsity $s = 10$ (b) Fixed size $n = 500$

Fig. 9: Average runtime comparison in the noisy case.

Runtimes for warm-starting SESP-D and SESP-P from the BPDN solution are omitted in Fig. 9, since BPDN alone

already achieves sufficient accuracy for all n and s considered. In regimes where BPDN accuracy is insufficient, however, it can serve as an inexpensive initialization: a low-cost BPDN estimate is first computed, then refined with SESP-D or SESP-P, trading additional computation for improved accuracy.

G. Sparse Modeling with Equivalent Representations

We next illustrate the ability of the proposed methods to recover multiple equivalent sparse representations of the same signal while explicitly controlling the target sparsity level.

The proposed methods may return sparse solutions with equal sparsity but different supports. Hence, performance is no longer evaluated via the relative error on $\hat{\mathbf{x}}$. Instead, a computed solution $\hat{\mathbf{x}}$ is declared successful if it satisfies $\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}\|_2 < 10^{-10}$ and $\|\hat{\mathbf{x}}\|_0 \leq s$, where entries with $|\hat{x}_i| < 10^{-8}$ are treated as zero when computing $\|\hat{\mathbf{x}}\|_0$.

Let $\mathbf{t}$ be the time vector with entries $t_i = i$, for $i = 0, 1, \ldots, 10$. We construct the matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{1} & \mathbf{t} & \mathbf{t}-\mathbf{1} & \mathbf{t}+\mathbf{1} & \mathbf{t}^{\cdot 2} & \mathbf{t}^{\cdot 3} \end{bmatrix},$$

where $\mathbf{t}^{\cdot k}$ denotes the entrywise power and $\mathbf{1}$ is the all-ones vector. Since $\mathbf{A}$ has linearly dependent columns, the same signal may admit multiple sparse representations with different supports and different sparsity levels.

Consider the signal $\mathbf{b} = \frac{2}{3} \cdot \mathbf{1} + \frac{1}{3} \cdot \mathbf{t}^{\cdot 3}$, which admits the equivalent representations

$$\begin{aligned}
\hat{\mathbf{x}}_1 &= \begin{bmatrix} \frac{2}{3} & 0 & 0 & 0 & 0 & \frac{1}{3} \end{bmatrix}^\top, && \|\hat{\mathbf{x}}_1\|_0 = 2, \\
\hat{\mathbf{x}}_2 &= \begin{bmatrix} 0 & 0 & -\frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \end{bmatrix}^\top, && \|\hat{\mathbf{x}}_2\|_0 = 3, \\
\hat{\mathbf{x}}_3 &= \begin{bmatrix} 0 & \frac{2}{3} & -\frac{2}{3} & 0 & 0 & \frac{1}{3} \end{bmatrix}^\top, && \|\hat{\mathbf{x}}_3\|_0 = 3, \\
\hat{\mathbf{x}}_4 &= \begin{bmatrix} 0 & -\frac{2}{3} & 0 & \frac{2}{3} & 0 & \frac{1}{3} \end{bmatrix}^\top, && \|\hat{\mathbf{x}}_4\|_0 = 3.
\end{aligned}$$

BP seeks a single minimum-ℓ_1 solution and cannot find multiple sparse representations. OMP follows a deterministic greedy path and is similarly limited to a single solution. In contrast, SESP-D and SESP-P recover all four solutions.

SESP-D and SESP-P require specifying a target sparsity level s. The multilinear sparsity constraints define an algebraic variety containing all vectors with $\|\mathbf{x}\|_0 \leq s$; consequently, even when the target sparsity exceeds the true sparsity, lower-sparsity solutions remain recoverable. In this example, setting $s = 2$ always yields $\hat{\mathbf{x}}_1$, while setting $s = 3$ yields $\hat{\mathbf{x}}_2$, $\hat{\mathbf{x}}_3$, or $\hat{\mathbf{x}}_4$, but also $\hat{\mathbf{x}}_1$, depending on the initialization.

SESP-D is initialized with a vector drawn from the normal distribution $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \gamma^2 \mathbf{I}_n)$, where $\gamma > 0$. SESP-P is initialized with $\mathbf{y}_0 = \mathbf{V}^\top(\mathbf{x}_0 - \mathbf{x}_p)$. We vary γ to examine how initialization scale affects the recovery frequency of each of the four solutions, testing 1000 random initializations per value of γ and reporting the resulting percentages in Fig. 10. The results show that even when the target sparsity s overestimates the true minimal sparsity, the recovered solution is most often the one of smaller sparsity ($\hat{\mathbf{x}}_1$) rather than the higher-sparsity alternatives ($\hat{\mathbf{x}}_2$, $\hat{\mathbf{x}}_3$, $\hat{\mathbf{x}}_4$).

The Macaulay method is also evaluated on this example. Unlike the previous experiments, where its computational cost rendered it intractable, the limited problem size here makes it feasible. In contrast to SESP-D and SESP-P, the Macaulay method is formally guaranteed to recover all solutions at once. Specifically, $\hat{\mathbf{x}}_2$, $\hat{\mathbf{x}}_3$, and $\hat{\mathbf{x}}_4$ are recovered each with multiplicity one, and $\hat{\mathbf{x}}_1$ with multiplicity three.

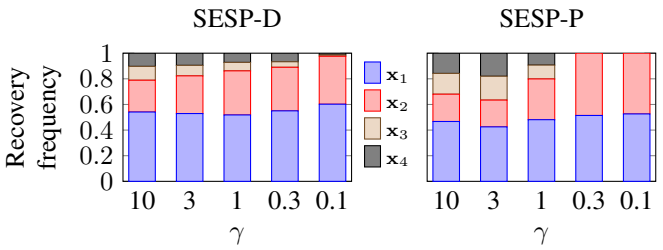


Fig. 10: Recovery frequency by component for ELS-CPD (left) and Nullspace (right) as a function of γ.

The tensor constructed in (12) is decomposed using the CPD-NLS algorithm in Tensorlab with GEVD-based initialization. Function and step tolerances are set to ε^2 and ε, respectively, and a maximum of 500 iterations is used. The solutions at infinity that the method finds are discarded.

VII. Conclusion and Future Work

We reformulated sparse approximation as a structured system of polynomial equations. The Macaulay construction is guaranteed to algebraically recover all s-sparse solutions at once, but is tractable only for small problems. We also proposed two optimization-based methods, SESP-D and SESP-P, which exploit the structure of the sparsity constraints via inexpensive ESP estimation and keep the per-iteration cost comparable to that of BP.

In our experiments, SESP-D and SESP-P recovered sparse signals from fewer measurements than BP and OMP, succeeded at higher sparsity levels, and remained close to the oracle bound under additive noise. This accuracy comes at a higher computational cost, which is most pronounced in the noisy setting. When BP or BPDN already succeeds, it remains the cheaper choice; when it does not, its solution can serve as an inexpensive initialization for the proposed methods.

The formulation offers two further properties of interest. First, several equivalent sparse representations of the same signal can be recovered, which neither BP nor OMP provides. Second, unlike in BP and BPDN, where sparsity is obtained indirectly through a regularization parameter, the target sparsity is specified directly and acts as an upper bound rather than an exact requirement.

Several directions remain open. First, the connection with EVD and CPD makes it in principle possible to derive upper bounds on the estimation error in the noisy case. Second, our approach extends to constrained sparse approximation, since any constraint that can be written polynomially can be appended to the system. Example of such constrained variants are: the unit-norm constraint, structured sparsity patterns, and nonnegativity. Third, the approach may be extended to parametrized dictionaries if the dependence on the unknown parameters is polynomial, as those parameters may be treated as additional variables. Fourth, faster Macaulay solvers would make the simultaneous recovery of all solutions practical for larger problems.

APPENDIX A
STRUCTURE OF $\mathbf{Z}^\top\mathbf{Z}$

Let $[n] = \{1, 2, \ldots, n\}$. We index the standard basis of $\mathbb{R}^{n^{s+1}}$ by multi-indices $(i_1, \ldots, i_{s+1}) \in [n]^{s+1}$, with corresponding basis vectors $\mathbf{e}_{i_1,\ldots,i_{s+1}} = \mathbf{e}_{i_1} \otimes \mathbf{e}_{i_2} \otimes \cdots \otimes \mathbf{e}_{i_{s+1}}$ where $\mathbf{e}_{i_j}$ are basis vectors of $\mathbb{R}^n$.

Since $\mathbf{Z}$ consists of one-hot rows, the matrix $\mathbf{Z}^\top\mathbf{Z}$ is diagonal with diagonal entries in $\{0, 1\}$. The matrix $\mathbf{Z}^\top\mathbf{Z}$ acts as a projector onto the subspace of $\mathbb{R}^{n^{s+1}}$ spanned by basis vectors whose indices are pairwise distinct. Hence

$$\mathbf{Z}^\top\mathbf{Z} = \sum_{\substack{(i_1,\ldots,i_{s+1})\in[n]^{s+1} \\ i_1\neq\cdots\neq i_{s+1}}} \mathbf{e}_{i_1,\ldots,i_{s+1}}\mathbf{e}_{i_1,\ldots,i_{s+1}}^\top.$$

Equivalently, we obtain the Kronecker representation

$$\mathbf{Z}^\top\mathbf{Z} = \sum_{\substack{(i_1,\ldots,i_{s+1})\in[n]^{s+1} \\ i_1\neq\cdots\neq i_{s+1}}} \mathbf{e}_{i_1}\mathbf{e}_{i_1}^\top \otimes \cdots \otimes \mathbf{e}_{i_{s+1}}\mathbf{e}_{i_{s+1}}^\top. \tag{20}$$

APPENDIX B
SESP-D DERIVATION

Objective Function The objective function f is obtained from the residual $\mathbf{r}$ as $f = \frac{1}{2}\mathbf{r}^\top\mathbf{r}$. From the formulation of SESP-D (17), it follows

$$\mathbf{r} = \begin{bmatrix} \mathbf{r}^{(1)} \\ \lambda\mathbf{r}^{(2)} \end{bmatrix} = \begin{bmatrix} \mathbf{A}\mathbf{x} - \mathbf{b} \\ \lambda\mathbf{Z}\left(\otimes_{i=1}^{s+1}\mathbf{x}\right) \end{bmatrix}. \tag{21}$$

Thus, $f = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \frac{1}{2}\lambda^2\mathbf{r}^{(2)\top}\mathbf{r}^{(2)}$.

Using the mixed-product property of the Kronecker product, we obtain $(\mathbf{x}\otimes\mathbf{x})^\top(\mathbf{e}_i\mathbf{e}_i^\top \otimes \mathbf{e}_j\mathbf{e}_j^\top)(\mathbf{x}\otimes\mathbf{x}) = (\mathbf{x}^\top\mathbf{e}_i\mathbf{e}_i^\top\mathbf{x}) \otimes (\mathbf{x}^\top\mathbf{e}_j\mathbf{e}_j^\top\mathbf{x}) = x_i^2x_j^2$ Combining this identity with the structure of $\mathbf{Z}^\top\mathbf{Z}$ in (20) yields

$$\begin{aligned}\mathbf{r}^{(2)\top}\mathbf{r}^{(2)} &= \lambda^2 \left(\bigotimes_{k=1}^{s+1}\mathbf{x}\right)^\top \mathbf{Z}^\top\mathbf{Z} \left(\bigotimes_{l=1}^{s+1}\mathbf{x}\right) \\ &= \lambda^2 \sum_{\substack{(i_1,\ldots,i_{s+1})\in[n]^{s+1} \\ i_1\neq\ldots\neq i_{s+1}}} \prod_{k=1}^{s+1} x_{i_k}^2.\end{aligned}$$

Each ordered tuple $(i_1, \ldots, i_{s+1})$ with distinct elements corresponds to a permutation of a subset $S \subset [n]$ with $|S| = s+1$. For each subset there are $(s+1)!$ such permutations. Grouping terms by subsets therefore yields

$$\mathbf{r}^{(2)\top}\mathbf{r}^{(2)} = (s+1)! \sum_{\substack{S\subset[n] \\ |S|=s+1}} \prod_{i\in S} x_i^2.$$

Expanding the sum over subsets gives

$$\sum_{\substack{S\subset[n] \\ |S|=s+1}} \prod_{i\in S} x_i^2 = \sum_{\substack{1\le i_1<\cdots \\ \cdots<i_{s+1}\le n}} x_{i_1}^2\cdots x_{i_{s+1}}^2 \overset{\text{Def. 2}}{=} E_{s+1}(\mathbf{x}^{\cdot 2}).$$

Therefore, the expression for the objective function is

$$f = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \frac{1}{2}\lambda^2(s+1)!\, E_{s+1}(\mathbf{x}^{\cdot 2}).$$

Gradient: The gradient splits as

$$\mathbf{J}^\top\mathbf{r} = \nabla f = \frac{1}{2}\nabla\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \frac{1}{2}\lambda^2(s+1)!\nabla E_{s+1}$$

where $\nabla\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 = 2\mathbf{A}^\top(\mathbf{A}\mathbf{x} - \mathbf{b})$ and $\nabla E_{s+1}(\mathbf{x}^{\cdot 2})$ is obtained using the fact that $\frac{\partial}{\partial x_i}E_{k+1} = 2x_iE_k^{(i)}$. Thus, $\nabla E_{s+1} = 2\sum_{i=1}^n E_s^{(i)}x_i\mathbf{e}_i$.

Gramian of the Jacobians: The Gramian of the Jacobians splits as

$$\mathbf{J}^\top\mathbf{J} = \mathbf{A}^\top\mathbf{A} + \lambda^2\mathbf{J}^{(2)\top}\mathbf{J}^{(2)},$$

where $\mathbf{J}^{(1)} = \mathbf{A}$ follows from $\mathbf{r}^{(1)} = \mathbf{A}\mathbf{x}-\mathbf{b}$, and $\mathbf{J}^{(2)}$ denotes the Jacobian of $\mathbf{r}^{(2)}$ (21). We derive $\mathbf{J}^{(2)\top}\mathbf{J}^{(2)}$ as follows.

The entries of $\mathbf{r}^{(2)}$ are indexed by ordered tuples $(i_1, \ldots, i_{s+1}) \in [n]^{s+1}$ with pairwise distinct indices, and the corresponding entry is $r_{(i_1,\ldots,i_{s+1})}(\mathbf{x}) = x_{i_1}\cdots x_{i_{s+1}}$. Every permutation of $S \subset [n]$ with $|S| = s+1$ yields the same value

$$r_S(\mathbf{x}) = \prod_{i\in S} x_i,$$

so each subset S contributes exactly $(s+1)!$ entries to $\mathbf{r}^{(2)}$, all equal to $r_S(\mathbf{x})$ and hence all sharing the same gradient ∇r_S. Using $\mathbf{J}^{(2)\top}\mathbf{J}^{(2)} = \sum_k(\nabla r_k^{(2)})(\nabla r_k^{(2)})^\top$ and grouping terms by subset gives

$$\mathbf{J}^{(2)\top}\mathbf{J}^{(2)} = (s+1)! \sum_{\substack{S\subset[n] \\ |S|=s+1}} (\nabla r_S)(\nabla r_S)^\top. \tag{22}$$

Since $\nabla r_S = \sum_{i\in S}\prod_{k\in S\setminus\{i\}} x_k\mathbf{e}_i$,, it follows that

$$(\nabla r_S)(\nabla r_S)^\top = \sum_{i,j\in S}\prod_{k\in S\setminus\{i\}} x_k \prod_{l\in S\setminus\{j\}} x_l\mathbf{e}_i\mathbf{e}_j^\top.$$

If $i = j$, the coefficient simplifies to $\prod_{k\in S\setminus\{i\}} x_k^2$. Summing over all subsets S in (22) yields $E_s^{(i)}$ at the position $\mathbf{e}_i\mathbf{e}_i^\top$ in the matrix $\mathbf{J}^{(2)\top}\mathbf{J}^{(2)}$.

If $i \neq j$, the coeficient factorises as $x_ix_j\prod_{k\in S\setminus\{i,j\}} x_k^2$. Summing over all subsets S in (22) yields $x_ix_jE_{s-1}^{(i,j)}$ at the positions $\mathbf{e}_i\mathbf{e}_j^\top$ and $\mathbf{e}_j\mathbf{e}_i^\top$ in the matrix $\mathbf{J}^{(2)\top}\mathbf{J}^{(2)}$.

Therefore,

$$\mathbf{J}^{(2)\top}\mathbf{J}^{(2)} = (s+1)!\sum_{i=1}^n E_s^{(i)}\,\mathbf{e}_i\mathbf{e}_i^\top + \sum_{\substack{i,j=1 \\ i\neq j}}^n E_{s-1}^{(i,j)}\,x_ix_j\,\mathbf{e}_i\mathbf{e}_j^\top.$$

APPENDIX C
SESP-P DERIVATION

The residual $\mathbf{r} = \mathbf{Z}\left(\otimes_{i=1}^{s+1}\mathbf{x}\right) = \mathbf{Z}\left(\otimes_{i=1}^{s+1}(\mathbf{x}_p + \mathbf{V}\mathbf{y})\right)$ is of the same form as $\mathbf{r}^{(2)}$ in (21). Thus, the objective

$$f = \frac{1}{2}(s+1)!\, E_{s+1}(\mathbf{x}^{\cdot 2})$$

is obtained in the same way as in Appendix B.

An important difference is that SESP-P optimizes over $\mathbf{y}$ instead of $\mathbf{x}$. Since $\mathbf{x} = \mathbf{x}_p + \mathbf{V}\mathbf{y}$, the Jacobian of $\mathbf{x}$ with respect to $\mathbf{y}$ is $\frac{\partial\mathbf{x}}{\partial\mathbf{y}} = \mathbf{V}$. Thus, using the chain rule,

$$\mathbf{J}^\top\mathbf{r} = \left(\frac{\partial\mathbf{x}}{\partial\mathbf{y}}\right)^\top \mathbf{J}_\mathbf{x}^\top\mathbf{r} = (s+1)!\,\mathbf{V}^\top\sum_{i=1}^n E_s(\mathbf{x}_{\setminus\{i\}}^{\cdot 2})x_i\mathbf{e}_i,$$

$$\mathbf{J}^\top\mathbf{J} = (s+1)!\mathbf{V}^\top\Big(\sum_i E_s^{(i)}\mathbf{e}_i\mathbf{e}_i^\top + \sum_{i,j} E_{s-1}^{(i,j)}x_ix_j\mathbf{e}_i\mathbf{e}_j^\top\Big)\mathbf{V}.$$

APPENDIX D
PROOF OF LEMMA 1

Proof. Partition $e_d(z_1, \ldots, z_k)$, with $d \leq k$, into two parts: one that does not contain z_k and one that does. The former is $e_d(z_1, \ldots, z_{k-1}) = \sum_{1 \leq i_1 < \cdots < i_d \leq k-1} z_{i_1} \cdots z_{i_d}$. The latter is $z_k e_{d-1}(z_1, \ldots, z_{k-1}) = \sum_{1 \leq i_1 < \cdots < i_{d-1} \leq k-1} z_{i_1} \cdots z_{i_{d-1}} z_k$. Therefore, for every $j = 1, \ldots, \min(k, d)$,

$$e_j(z_1, \ldots, z_k) = e_j(z_1, \ldots, z_{k-1}) + z_k e_{j-1}(z_1, \ldots, z_{k-1}). \tag{23}$$

The recursion is initialized by $e_0(z_1, \ldots z_k) = 1$. Evaluating the recursion for $k = 2, \ldots, n$ requires $\mathcal{O}(nd)$ operations and yields $e_d(z_1, \ldots, z_n)$. □

APPENDIX E
PROOF OF LEMMA 2

Proof. Compute left and right partial ESPs, for $k = 0, \ldots, s$,

$$E_k^{\mathrm{L}}(i) := e_k\big(x_1^2, \ldots, x_i^2\big), \quad E_k^{\mathrm{R}}(i) := e_k\big(x_i^2, \ldots, x_n^2\big),$$

with boundary conventions $E_0^{\mathrm{L}}(0) = 1$ and $E_0^{\mathrm{R}}(n+1) = 1$. Both tables are computed via (23), sweeping left-to-right and right-to-left respectively, each in $O(ns)$ time.

Partition each size-s subset of $\{1, \ldots, n\} \setminus \{i\}$ according to how many elements lie strictly to the left versus strictly to the right of i. A subset contributing j elements from $\{1, \ldots, i-1\}$ and $s - j$ elements from $\{i+1, \ldots, n\}$ contributes a term $E_j^{\mathrm{L}}(i-1)\, E_{s-j}^{\mathrm{R}}(i+1)$ to $E_s^{(i)}$. Thus, summing over j gives

$$E_s^{(i)} = \sum_{j=0}^{s} E_j^{\mathrm{L}}(i-1)\, E_{s-j}^{\mathrm{R}}(i+1),$$

which costs $O(s)$. Evaluating all n values costs $O(ns)$. Together with the $O(ns)$ left and right table construction, the total complexity is $O(ns)$. □

APPENDIX F
PROOF OF LEMMA 3

Proof. Fix $i \in \{1, \ldots, n\}$ and let $S^{(i)} := \{x_1^2, \ldots, x_n^2\} \setminus \{x_i^2\}$ denote the set of $n-1$ variables with x_i^2 removed. By Lemma 2 applied to $S^{(i)}$, all $n-1$ values $E_{s-1}^{(i,j)}$, $j \neq i$, can be computed simultaneously in $\mathcal{O}(ns)$ time. Repeating for each $i = 1, \ldots, n$ yields all $\binom{n}{2}$ values $E_{s-1}^{(i,j)}$ in $\mathcal{O}(n^2 s)$ time in total. □

ACKNOWLEDGMENT

This work was supported by the Flemish Government's AI Research Program and KU Leuven Internal Funds (iBOF/23/064, C14/22/096).

REFERENCES

[1] D. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[2] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. R. Stat. Soc. Ser. B*, vol. 58, no. 1, pp. 267–288, 1996.

[3] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, p. 19–60, Mar. 2010.

[4] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, 2006.

[5] M. Elad, *Sparse and Redundant Representations : From Theory to Applications in Signal and Image Processing*, 1st ed. New York, NY: Springer New York, 2010.

[6] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, 2006.

[7] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2861–2873, 2010.

[8] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, 2009.

[9] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 113, no. 15, pp. 3932–3937, 2016.

[10] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM J. Comput.*, vol. 24, no. 2, pp. 227–234, 1995.

[11] A. Çivril, "Sparse approximation is provably hard under coherent dictionaries," *J. Comput. Syst. Sci.*, vol. 84, pp. 32–43, 2017.

[12] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, 1993, pp. 40–44 vol.1.

[13] G. Davis, S. Mallat, and M. Avellaneda, "Adaptive greedy approximations," *Constr. Approx.*, vol. 13, no. 1, pp. 57–98, 1997.

[14] A. J. Miller, *Subset Selection in Regression*, 2nd ed. London, U.K.: Chapman and Hall, 2002.

[15] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1998.

[16] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*, 1st ed., ser. Applied and Numerical Harmonic Analysis. New York, NY: Birkhäuser, 2013.

[17] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.

[18] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, 2017.

[19] N. Vervliet and L. De Lathauwer, "Numerical optimization-based algorithms for data fusion," in *Data handling in science and technology*. Elsevier, 2019, vol. 31, pp. 81–128.

[20] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer. (2016) Tensorlab 3.0. [Online]. Available: https://www.tensorlab.net

[21] J. Nocedal and S. Wright, *Numerical optimization*, 2nd ed., ser. Springer series in operations research and financial engineering. New York, NY: Springer, 2006.

[22] M. Boussé, N. Vervliet, I. Domanov, O. Debals, and L. De Lathauwer, "Linear systems with a canonical polyadic decomposition constrained solution: Algorithms and applications," *Numer. Linear Algebra Appl.*, vol. 25, no. 6, pp. 1–18, Aug. 2018.

[23] K. Batselier, P. Dreesen, and B. De Moor, "On the null spaces of the Macaulay matrix," *Linear Algebra Appl.*, vol. 460, pp. 259–289, 2014.

[24] J. Vanderstukken and L. De Lathauwer, "Systems of polynomial equations, higher-order tensor decompositions, and multidimensional harmonic retrieval: A unifying framework. Part I: The canonical polyadic decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 42, no. 2, pp. 883–912, 2021.

[25] D. A. Cox, J. B. Little, and D. O'Shea, *Using Algebraic Geometry*, ser. Graduate Texts in Mathematics. New York, NY, USA: Springer, 1998, vol. 185.

[26] S. Hendrikx and L. De Lathauwer, "Block row kronecker-structured linear systems with a low-rank tensor solution," *Front. Appl. Math. Stat.*, vol. 8, p. 832883, 2022.

[27] I. G. Macdonald, *Symmetric Functions and Hall Polynomials*. Oxford University Press, 1995.

[28] L. N. Trefethen and D. Bau, III, *Numerical Linear Algebra*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1997.

[29] E. van den Berg and M. P. Friedlander, "SPGL1: A solver for large-scale sparse reconstruction," December 2019, https://friedlander.io/spgl1.

[30] ——, "Probing the pareto frontier for basis pursuit solutions," *SIAM J. Sci. Comput.*, vol. 31, no. 2, pp. 890–912, 2008.

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