Tensor-Based Multivariate Polynomial Optimization with Application in Blind Identification

Muzaffer Ayvaz*†, Lieven De Lathauwer*†
*Department of Electrical Engineering (ESAT), KU Leuven, Leuven, Belgium
†Group Science, Engineering and Technology, KU Leuven Kulak, Kortrijk, Belgium
{Muzaffer.Ayvaz, Lieven.DeLathauwer}@kuleuven.be

***Abstract*—Multivariate polynomial optimization problems are ubiquitous in signal processing, machine learning, and artificial intelligence. Examples include, but are not limited to, blind source separation, classification, multivariate polynomial regression, and tensor eigenvalue problems. Efficient algorithms for these problems have been studied in the case where the problem can be written as a best low rank tensor approximation. In the same spirit, we aim to extend these algorithms to a larger class of cost functions by representing multivariate polynomials using compact tensor models. This tensor-based multivariate polynomial optimization framework will allow us to tackle a broader range of problems than what is possible with existing methods. In this paper, we focus on the symmetric CPD format for representing multivariate polynomials, and show that exploiting this structure results in efficient numerical optimization-based algorithms. We demonstrate our approach for the blind deconvolution of constant modulus signals, outperforming state-of-the-art algorithms in computational time while maintaining similar accuracy.**

***Index Terms*—Blind deconvolution, blind identification, constant modulus, multivariate polynomial, numerical optimization, tensor decomposition**

I. Introduction

Multivariate polynomial optimization (MPO) is a fundamental problem in signal processing, machine learning, and artificial intelligence (AI). In the current information age, the main challenge is to efficiently process and represent large amounts of data from possibly heterogeneous sources, while addressing MPO problems. Existing approaches for MPO are computationally infeasible. For example, certain MPO-type problems such as fault detection and image classification can be written as semi-definite programming (SDP) with convex relaxations [1]. SDP-based frameworks such as CVX, SeDuMi, and SDPT3 do not scale well for large-scale problems [2]–[4]. We aim to tackle this challenge in this study.

For the processing and representation of high-order data, tensors (higher-order arrays) have become indispensable, because they preserve the inherent structure of higher-order data and can be decomposed uniquely under mild conditions [5]–[7]. In many applications, such as face recognition and epileptic seizure detection, optimization problems can be written as a (coupled) tensor decomposition problem, and are particular cases of MPO. Effective numerical strategies based on standard optimization algorithms have been developed for tensor decomposition problems and applied in diverse fields [8]–[11].

Our contribution is to generalize these existing numerical strategies to a broader class of problems beyond the decomposition of given data in matrix/tensor format. Examples include multivariate polynomial regression, multivariate logistic regression, and the tensor eigenvalue problem with applications to blind source separation, and magnetic resonance imaging [12]–[15]. In this study, we introduce a general framework called tensor-based multivariate polynomial optimization (TeMPO). First, we define the optimization problem and show how to exploit structure in the case where the data are represented by a symmetric canonical polyadic decomposition (CPD). Next, we illustrate the efficiency of our approach with the blind deconvolution of constant modulus (CM) signals, which is a basic signal processing problem arising in communication systems [16].

We describe notation and tensor background in the next section. Next, we explain our framework in Sec. III. In Sec. IV, we discuss the derivation and implementation details of the complex Gauss–Newton (GN) algorithm. We illustrate the framework with the blind deconvolution of CM signals in Sec. V. Lastly, we present numerical results in Sec. VI.

II. Notation and Definitions

A tensor is a higher-order generalization of a vector (first-order) and a matrix (second-order). Following established conventions in signal processing, we denote scalars, vectors, matrices and tensors by a, $\mathbf{a}$, $\mathbf{A}$, and $\mathcal{A}$, respectively. The complex conjugate, transpose, conjugated transpose, and inverse of matrix $\mathbf{A}$ are denoted as $\overline{\mathbf{A}}$, $\mathbf{A}^{\mathrm{T}}$, $\mathbf{A}^{\mathrm{H}}$, and $\mathbf{A}^{-1}$, respectively. The matrix created by removing the first row of $\mathbf{A}$ is denoted by $\widetilde{\mathbf{A}}$. Similarly, $\widetilde{\mathbf{a}}$ denotes a vector $\mathbf{a}$ whose first element is removed. A vector of length K with all entries equal to 1 is denoted by $\mathbf{1}_K$. The outer product, Kronecker product, Khatri–Rao

Research supported by: (1) Flemish Government: This work was supported by the Fonds de la Recherche Scientifique–FNRS and the Fonds Wetenschappelijk Onderzoek– Vlaanderen under EOS project no 30468160 (SeLMA) and by Artificiële Intelligentie (AI) Vlaanderen (3E190661); (2) KU Leuven Internal Funds C16/15/059 and ID-N project no 3E190402; (3) Leuven Institute for Artificial Intelligence (Leuven.ai).

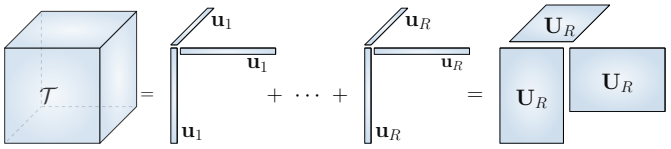

Fig. 1. Polyadic decomposition of a third order symmetric tensor $\mathcal{T}$. It is called canonical (CPD) if R is equal to the rank of $\mathcal{T}$, i.e., R is minimal. It allows compact representation of polynomials.

product and Hadamard product are denoted by $\otimes$, $\bigotimes$, $\odot$, and $*$, respectively. The mode-n product of a tensor $\mathcal{A} \in \mathbb{K}^{I_1 \times I_2 \times \ldots \times I_N}$ (with $\mathbb{K}$ meaning $\mathbb{R}$ or $\mathbb{C}$) and a vector $\mathbf{x} \in \mathbb{K}^{I_n}$ denoted by $\mathcal{A} \cdot_n \mathbf{x}^{\mathrm{T}}$, is defined element-wise as $(\mathcal{A} \cdot_n \mathbf{x}^{\mathrm{T}})_{i_1 i_2 \cdots i_{n-1} i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_N} a_{i_1 i_2 \cdots i_n \cdots i_N} x_{i_n}$. The mode-$n$ matricization of $\mathcal{A}$ is the matrix $\mathbf{A}_{(n)}$ collecting all the mode-n vectors as its columns following the ordering convention in [7]. A kth-order tensor $\mathcal{A} \in \mathbb{K}^{I \times \ldots \times I}$ is called symmetric if its entries do not change under the permutation of its k indices and the matrix representations of symmetric tensors in different modes are all equal. A rank-1 tensor of order N is defined as the outer product of N nonzero vectors. The rank of a tensor is equal to the minimal number of rank-1 terms that produce the tensor as their sum. The canonical polyadic decomposition writes a tensor as a sum of R rank-1 tensors and is denoted by $[\![\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}]\!]$, with its factor matrices $\mathbf{U}^{(n)}$, see Fig. 1, where R equals the rank of the tensor. For symmetric tensors, all the factor matrices are equal, i.e. $\mathcal{T} = [\![\mathbf{U}, \mathbf{U} \ldots, \mathbf{U}, \mathbf{c}^{\mathrm{T}}]\!]$, where $\mathbf{c} \in \mathbb{R}^R$ is a vector of weights which allows us to give minus signs to the factors for even-degree symmetric tensors.

III. Tensor-Based Multivariate Polynomial Optimization (TeMPO)

A general unconstrained optimization problem is:

$$\min_{\mathbf{z}} \quad f(\boldsymbol{\theta}, \mathbf{z}) + h(\mathbf{z}), \tag{1}$$

in which $f(\boldsymbol{\theta}, \mathbf{z})$ is the objective function, $h(\mathbf{z})$ is the regularization function, $\boldsymbol{\theta}$ denotes available data, and $\mathbf{z}$ contains the optimization variables. Generally, $f(\boldsymbol{\theta}, \mathbf{z})$ serves as the performance measure of the model to be optimized and $h(\mathbf{z})$ serves as the penalty term for the structural constraints of the model parameters. State-of-the-art numerical optimization-based algorithms and frameworks have been proposed in the case where $f(\boldsymbol{\theta}, \mathbf{z})$ can be expressed as a matrix/tensor decomposition [10], [11], [17]. Since all continuous functions can be approximated by polynomials, we can reasonably assume that $f(\boldsymbol{\theta}, \mathbf{z})$ is approximately a multivariate polynomial. This allows us to deal with a wider class of problems. Moreover, $\boldsymbol{\theta}$ is often available in structured tensor formats in applications due to preprocessing techniques, such as tensorization and data augmentation, applied a priori in the multiway analysis [12]–[14], [18], [19].

We use tensors to represent multivariate polynomials. An mth-order multivariate homogeneous polynomial $p(\mathbf{z})$ can be written by using mth-order tensor $\mathcal{T}_m$ as [18],

$$p(\mathbf{z}) = \mathcal{T} \cdot_1 \mathbf{z}^{\mathrm{T}} \cdot_2 \mathbf{z}^{\mathrm{T}} \ldots \cdot_m \mathbf{z}^{\mathrm{T}} \stackrel{\text{def}}{=} \mathcal{T}_m \mathbf{z}^m.$$

Since all polynomials can be written as the sum of homogeneous polynomials, any polynomial of order n can be written by using tensors of orders up to n. Note that in the tensor representation of polynomials, all tensors can be assumed to be symmetric without loss of generality[1].

We consider a simple MPO problem to put more emphasis on the advantages of tensor representation:

$$\min_{\mathbf{z}} \quad f(\mathbf{z}) = \min_{\mathbf{z}} \quad \sum_{k=0}^{n} \mathcal{T}_k \mathbf{z}^k,$$

where $\mathcal{T}_k$ denotes a kth-order symmetric tensor. The representation of polynomials by tensors allows for new insights. In many applications from signal processing, these tensors are available in structured formats, such as CPD, low multilinear rank (LMRA), tensor trains (TT) and hierarchical Tucker (HT) [18]–[20], and the main advantages of the proposed method lie here. The tensor contraction $\mathcal{T}_k \mathbf{z}^k$ is the core computation of TeMPO in the numerical optimization setting and can efficiently be computed for large-scale tensors when the structure of the tensor is exploited [21]. Indeed, even if the tensors $\mathcal{T}_k$ are dense, contractions can be computed in a memory efficient way by storing them in a compact or sparse format [22]. Additionally, in the case of larger order polynomials, efficient stochastic and/or randomized algorithms can be integrated into the TeMPO framework, see, e.g., [23], [24].

Numerical optimization algorithms often require expressions for the gradient, Gramian, Gramian-vector product and/or Hessian. In this paper, we limit ourselves to the GN algorithm and only derive the first-order derivative of $\mathcal{T}_m \mathbf{z}^m$ for symmetric tensors and tensors in symmetric CPD formats. The second-order derivatives can be computed using a similar approach.

A. Derivative of $\mathcal{T}_m \mathbf{z}^m$ for Dense Symmetric Tensors

By using the definition of the gradient of a vector-valued function and the mode-n product, one can easily obtain the derivative of the function $f(\mathbf{z}) = \mathcal{T}_m \mathbf{z}^m$ w.r.t. $\mathbf{z}$ for an mth-order symmetric tensor $\mathcal{T}_m \in \mathbb{K}^{I \times \cdots \times I}$ and $\mathbf{z} \in \mathbb{K}^I$ as:

$$\nabla f(\mathbf{z}) = m \mathcal{T} \mathbf{z}^{m-1}. \tag{2}$$

B. Derivatives of $\mathcal{T} \mathbf{z}^m$ for Symmetric CPD Tensors

The CPD of an mth-order symmetric tensor $\mathcal{T}_m \in \mathbb{K}^{I \times \ldots \times I}$ in matricized format can be written as:

$$\mathbf{T}_{(n)} = \mathbf{U}(\mathbf{U} \odot \mathbf{U} \odot \ldots \odot \mathbf{U})^{\mathrm{T}} \stackrel{\text{def}}{=} \mathbf{U}(\mathbf{U}^{\odot(m-1)})^{\mathrm{T}}.$$

[1]To see this, let us consider the homogeneous polynomial $p(x, y) = x^3 + \alpha x^2 y + \beta x y^2 + y^3$. A third order symmetric tensor $\mathcal{T}$ with the elements $a_{111} = 1$, $a_{222} = 1$, $a_{112} = a_{121} = a_{211} = \alpha/3$, $a_{221} = a_{212} = a_{122} = \beta/3$ can represent the polynomial.

Using (2) and utilizing the property $(\mathbf{A} \odot \mathbf{B})^{\mathrm{T}}(\mathbf{C} \odot \mathbf{D}) = (\mathbf{A}^{\mathrm{T}}\mathbf{C}) * (\mathbf{B}^{\mathrm{T}}\mathbf{D})$, we obtain the derivatives of the multilinear forms for symmetric tensors in CPD format as:

$$\nabla f(\mathbf{z}) = m\mathbf{U}(\mathbf{U}^{\mathrm{T}}\mathbf{z})^{*(m-1)}. \tag{3}$$

Exploiting symmetric CPD structure significantly reduces both the computational complexity and storage requirements. We only need to store $\mathbf{U}$ instead of the full tensor $\mathcal{T}_m$ and matrix-vector operations, as is clear in (3). This is where our main speed-up is realized. A more detailed description will be given in Sec. V.

IV. Complex GN Algorithm for TeMPO

Complex-valued signals are of fundamental interest in signal processing. Optimization of real-valued functions in complex variables is one of the fundamental approaches to exploit the impropriety and non-circularity of signals [25]. To this end, generalizations of standard numerical optimization algorithms have been generalized to the complex domain [17], [26]. Also, the GN algorithm is often preferred when the cost function is a sum of squares thanks to its convergence properties [27]. Here, we will briefly explain the complex generalization of the GN algorithm, which we will use in our illustrative example in Sec. V. The complex Gauss–Newton (cGN) algortihm solves the unconstrained nonlinear least-squares problem:

$$\min_{\mathbf{z},\overline{\mathbf{z}}} \quad f(\mathbf{z},\overline{\mathbf{z}}) = \min_{\mathbf{z},\overline{\mathbf{z}}} \quad \frac{1}{2}\left|\left|\mathbf{r}(\mathbf{z},\overline{\mathbf{z}})\right|\right|_2^2 \tag{4}$$

where $f(\mathbf{z},\overline{\mathbf{z}})$ is a real-valued function and $\mathbf{r}(\mathbf{z},\overline{\mathbf{z}})$ is called residual function. The Hessian for the cGN algorithm is a Hermitian matrix defined as:

$$\mathbf{H} = \begin{bmatrix} \mathbf{U}_{\mathbf{zz}} & \mathbf{U}_{\overline{\mathbf{z}}\mathbf{z}} \\ \mathbf{U}_{\mathbf{z}\overline{\mathbf{z}}} & \mathbf{U}_{\overline{\mathbf{z}}\overline{\mathbf{z}}} \end{bmatrix}, \text{where } \mathbf{U}_{zz} = \overline{\mathbf{U}_{\overline{z}\overline{z}}} \text{ and } \mathbf{U}_{\overline{z}z} = \overline{\mathbf{U}_{z\overline{z}}}.$$

The blocks of the Hessian can be obtained by only using the first-order information of $\mathbf{r}(\mathbf{z},\overline{\mathbf{z}})$ as:

$$\mathbf{U}_{\mathbf{zz}} = \frac{1}{2}\left(\mathbf{J}_{\mathbf{z}}{}^{\mathrm{H}}\mathbf{J}_{\mathbf{z}} + \overline{\mathbf{J}_{\overline{\mathbf{z}}}{}^{\mathrm{H}}\mathbf{J}_{\overline{\mathbf{z}}}}\right), \mathbf{U}_{\overline{\mathbf{z}}\mathbf{z}} = \frac{1}{2}\left(\mathbf{J}_{\mathbf{z}}{}^{\mathrm{H}}\mathbf{J}_{\overline{\mathbf{z}}} + \overline{\mathbf{J}_{\overline{\mathbf{z}}}{}^{\mathrm{H}}\mathbf{J}_{\mathbf{z}}}\right), \tag{5}$$

where $\mathbf{J}_{\mathbf{z}}$ and $\mathbf{J}_{\overline{\mathbf{z}}}$ are the Jacobian matrices of $r(\mathbf{z},\overline{\mathbf{z}})$ w.r.t. $\mathbf{z}$ and $\overline{\mathbf{z}}$, respectively. The cGN step $\mathbf{p}_k$ at the kth iteration for the optimization variable $\mathbf{c} = [\mathbf{z};\overline{\mathbf{z}}]$ is obtained by solving the following linear system of equations,

$$\mathbf{H}\mathbf{p}_k = -\mathbf{g}_k, \text{ where } \mathbf{g}_k = \left[\frac{\partial f}{\partial \overline{\mathbf{z}}}(\mathbf{z}_k), \frac{\partial f}{\partial \mathbf{z}}(\mathbf{z}_k)\right]^{\mathrm{T}}. \tag{6}$$

Here, $\mathbf{g}_k$ contains the gradient of the objective function (4) w.r.t. $\overline{\mathbf{z}}$ and $\mathbf{z}$, respectively and can be written as:

$$\overline{\frac{\partial f}{\partial \mathbf{z}}^{\mathrm{H}}} = \frac{\partial f}{\partial \overline{\mathbf{z}}}^{\mathrm{H}} = \frac{1}{2}\left(\mathbf{J}_{\mathbf{z}}{}^{\mathrm{H}}\mathbf{r} + \mathbf{J}_{\overline{\mathbf{z}}}{}^{\mathrm{H}}\mathbf{r}\right). \tag{7}$$

The most computationally expensive part of the algorithm is to solve (6). We use the conjugate gradient Steihaug method, which is implemented in TensorLab [9], [28] to solve (6). This method does not require matrix inversion. To reduce the overall computational complexity, it is crucial to reduce the number of CG iterations using preconditioning techniques. Here, we use the block-Jacobi preconditioning in view of the quasi-GN algorithm [26]. The preconditioner is obtained by setting the off-diagonal blocks of Hessian to zero, i.e. $\mathbf{U}_{\overline{\mathbf{z}}\mathbf{z}} = \mathbf{U}_{\mathbf{z}\overline{\mathbf{z}}} = \mathbf{0}$. The overall algorithm is summarized in Algorithm 1.

Algorithm 1: cGN algorithm with (PC)CG-Steihaug

Input : $\mathbf{r}(\mathbf{z},\overline{\mathbf{z}})$ – function to compute residual
$\mathbf{z}_0$ – initial condition
Output : $\mathbf{z}$ – the point where algorithm converged

```
while not converged do
    Cache repeated variables
    Compute r(z_k, z̄_k)
    Compute g_k using (7)
    Solve (6) using (preconditioned) CG-Steihaug
    Update z_k
end
```

V. Blind Deconvolution of CM Signals

In digital communication systems, the recovery of the transmitted input signals from the received output signals, which have been altered by the medium is one of the core problems. In blind deconvolution, one attempts to find the original input signal by only observing the output signal. Thus, constraints on signals and/or channel have to be imposed to obtain interpretable results. The constant modulus (CM) criterion is a widely used input constraint [29]. In this section, we apply our framework to the blind deconvolution of CM signals by formulating as an MPO problem. We limit ourselves to an autoregressive single-input single-output (SISO) system [30], given by

$$\sum_{l=0}^{L} w_l \cdot y[k-l] = s[k] + n[k], \text{ for } k = 1,\ldots,K, \tag{8}$$

where $y[k]$, $s[k]$, and $n[k]$ are the measured output signal, the input signal and the noise at the kth measurement, respectively. w_l denotes the lth filter coefficient. By ignoring the noise for ease of derivation, (8) can be written as:

$$\mathbf{Y}^{\mathrm{T}}\mathbf{w} = \mathbf{s}, \tag{9}$$

where $\mathbf{Y} \in \mathbb{C}^{L\times K}$ is a Toeplitz matrix and its rows are the subsequent observations under the assumption that we have $K+L-1$ samples $y[-L+1],\ldots,y[K]$. Also, the vector $\mathbf{w} \in \mathbb{K}^L$ contains the filter coefficients and the kth entry of the source vector $\mathbf{s} \in \mathbb{C}^K$ is the input signal at the kth time instance, i.e. $s_k = s[k]$. The aim of the blind deconvolution problem is to determine the filter coefficients $\mathbf{w}$, using only the values $\mathbf{Y}$. The CM property, which holds for phase- or frequency-modulated signals [31], [32] can be written as:

$$|s_k|^2 = c, \text{ for } k = 1,2,\ldots,K. \tag{10}$$

Here, c is a constant scalar which is known a priori. By substituting s_k defined in (9) into (10), we obtain,

$$\left(\mathbf{Y} \odot \overline{\mathbf{Y}}\right)^{\mathrm{T}} (\mathbf{w} \otimes \overline{\mathbf{w}}) = c \cdot \mathbf{1}_K. \tag{11}$$

Following the same intuition in [33], by multiplying (11) from the left with a Householder reflector $\mathbf{Q}$, generated for $c \cdot \mathbf{1}_K$, and removing the first equation[2], we obtain

$$\mathbf{M}(\mathbf{w} \otimes \overline{\mathbf{w}}) = \mathbf{0}. \tag{12}$$

Here, $\mathbf{M} = \widetilde{\mathbf{Q}}\left(\mathbf{Y} \odot \overline{\mathbf{Y}}\right)^{\mathrm{T}}$. In applications, $\mathbf{M}(\mathbf{w} \otimes \overline{\mathbf{w}})$ will not vanish exactly due to the noise. Hence, we look for the solution which minimizes its norm. Thus, we obtain the following MPO problem,

$$\min_{\mathbf{w},\overline{\mathbf{w}}} \; f(\mathbf{w}, \overline{\mathbf{w}}) = \min_{\mathbf{w},\overline{\mathbf{w}}} \; \frac{1}{2} \left|\left|\mathbf{M}(\mathbf{w} \otimes \overline{\mathbf{w}})\right|\right|_2^2, \tag{13}$$

under the normalization constraint $||\mathbf{w}|| = 1$.

A. Computation of the Objective Function

Computation of the objective function requires the computation of the norm of the residual vector,

$$\mathbf{r}(\mathbf{w}, \overline{\mathbf{w}}) = \widetilde{\mathbf{Q}}\left(\mathbf{Y} \odot \overline{\mathbf{Y}}\right)^{\mathrm{T}} (\mathbf{w} \otimes \overline{\mathbf{w}}) / \left|\left|\mathbf{w}\right|\right|^2,$$

where $\mathbf{w}$ is normalized to be able to utilize algorithms for unconstrained problems. As described in Sec. III-B, we only need to compute $\widetilde{\mathbf{Q}}(\mathbf{v} * \overline{\mathbf{v}})$, where $\mathbf{v} = \mathbf{Y}^{\mathrm{T}}\mathbf{w}$. The multiplication of the reflector $\mathbf{Q}$ by a vector needs only a single vector-vector multiplication, and $(\mathbf{v} * \overline{\mathbf{v}})$ needs only elementwise products. Hence, the computation is dominated by the computation of $\mathbf{v}$. The explicit computation of $\mathbf{v}$ requires $L \times K$ multiplications, which can be efficient when the problem size is small, i.e. $L \ll K$. For large-scale problems, exploiting the Toeplitz structure of $\mathbf{Y}$ through the DFT algorithm reduces the computational complexity to $\mathcal{O}\left(2N \log(N) + N\right)$ where $N = L + K$.

B. Jacobian and Jacobian-vector Product

The Jacobian of $\mathbf{r}(\mathbf{z}, \overline{\mathbf{z}})$ w.r.t. $\mathbf{z}$ is given by

$$\mathbf{J}_{\mathbf{z}} = \frac{\partial \mathbf{r}}{\partial \mathbf{z}} = \widetilde{\mathbf{Q}}\mathbf{D}(\mathbf{v})\mathbf{Y}^{\mathrm{T}} / \left|\left|\mathbf{z}\right|\right|^2 - \widetilde{\mathbf{r}}\overline{\mathbf{z}}^{\mathrm{T}} / \left|\left|\mathbf{z}\right|\right|^4,$$

using the derivations in Sec. III-B. Here, $\mathbf{D}(\overline{\mathbf{v}})$ is a diagonal matrix composed from the elements of $\overline{\mathbf{v}}$. $\mathbf{J}_{\overline{\mathbf{z}}}$ can be derived in a similar way. Note that $\mathbf{v}$ is both needed in the computation of the Jacobian and the residual, and therefore it will be computed only once per iteration. Similar to Sec. V-A, the computation of $\mathbf{J}_{\mathbf{z}}\mathbf{x}$ is dominated by $\mathbf{Y}^{\mathrm{T}}\mathbf{x} = \mathbf{u}$. The other operations, $\widetilde{\mathbf{Q}}\mathbf{D}(\mathbf{v})\mathbf{u}$ and $\overline{\mathbf{z}}^{\mathrm{T}}\mathbf{x}$, can be done in $\mathcal{O}\left(K\right)$ and $\mathcal{O}\left(L\right)$ operations, respectively.

[2]The first equation is only a normalization constraint.

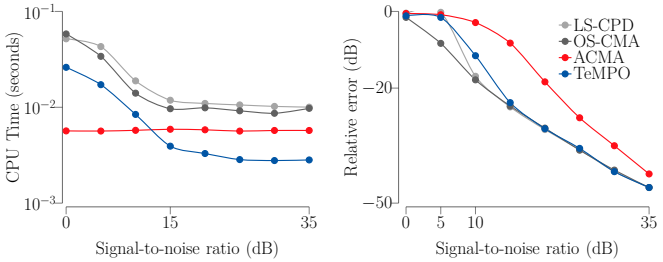

Fig. 2. TeMPO is faster than other algorithms for SNR > 10(dB), while obtaining more accurate results than ACMA.

C. Gradient and Gramian-vector Products

The computation of the gradient can be done using the Jacobian-vector products $\mathbf{J}_{\mathbf{z}}\mathbf{r}$ and $\mathbf{J}_{\overline{\mathbf{z}}}\mathbf{r}$, see (7). Thanks to the symmetry in (13), $\mathbf{J}_{\overline{z}} = \overline{\mathbf{J}_{\mathbf{z}}}$. Thus, only two Jacobian-vector products are sufficient for the computation of the gradient. As mentioned in Sec. IV, we solve (6) using (preconditioned) CG, which only requires the repeated Gramian-vector products of the form $\mathbf{J}_{\mathbf{z}}{}^{\mathrm{H}}\mathbf{J}_{\mathbf{z}}\mathbf{x}$ and $\mathbf{J}_{\overline{\mathbf{z}}}{}^{\mathrm{H}}\mathbf{J}_{\mathbf{z}}\mathbf{x}$, see (5). By using the equality of $\mathbf{J}_{\overline{z}} = \overline{\mathbf{J}_{\mathbf{z}}}$, we need four Jacobian-vector products, which require $\mathcal{O}\left(8N \log(N) + 4N\right)$ additional operations, when the DFT algorithm is utilized. As a result, the per-iteration complexity of cGN algorithm for the blind deconvolution of CM signals is logarithmic and particular operations are listed in Table I.

VI. Experimental Results

A number of algorithms have been developed to solve (11) and (12). The analytical CM algorithm (ACMA) [32] writes (12) as a generalized matrix eigenvalue problem. Gradient descent and stochastic gradient descent algorithms have also been proposed for the minimization of the expected value of $\{(|\mathbf{y_n}^{\mathrm{T}}\mathbf{w}| - c)^2\}$. The optimal step-size CMA (OSCMA) [34] algorithm uses a gradient descent algorithm, which computes the step size algebraically.The problem in (12) can also be interpreted as a linear system with a rank-1 constrained solution, which fits the LS-CPD framework in [35]. LS-CPD solves (11) by relaxing complex parameters and utilizing a second-order GN algorithm. We compare with these algorithms in terms of computation time and accuracy for a simple example.

We consider an autoregressive model of degree L = 10 with uniformly distributed coefficients between zero and one, sample length K = 300, and c = 1. We add scaled Gaussian noise to the measurements to obtain a particular signal-to-noise ratio (SNR). We run 50 experiments starting

TABLE I
The computational complexity of the cGN algorithm

	Calls per iteration	Complexity
Objective function	1	$\mathcal{O}\left(3N \log(N) + N\right)$
Gradient	1	$\mathcal{O}\left(4N \log(N) + 2N\right)$
Gramian-vector	it_{CG}	$\mathcal{O}\left(8N \log(N) + 4N\right)$

from the algebraic solution [35] for the SNR between 0-35 (dB). Fig. 2(a) shows the median CPU time in seconds. TeMPO is faster than ACMA, OS-CMA and LS-CPD for SNR > 10(dB) by exploiting the structure of the data. Fig. 2(b) shows the median relative error on $\mathbf{w}$. Clearly, TeMPO obtains similar accuracy as LS-CPD and OS-CMA, which are more accurate than ACMA.

VII. Conclusion and Further Work

We introduced the TeMPO framework for MPO. We demonstrated the efficiency of TeMPO for multivariate polynomials represented by symmetric CPD-structured tensors. We illustrated the efficiency of TeMPO with a blind deconvolution problem and showed that it outperforms state-of-the-art algorithms in computation time and achieves similar accuracy. In future work, we will investigate LMRA, HT and TT structured tensors. We will apply TeMPO to a range of problems in signal processing, machine learning, and data analysis.

Acknowledgment

The authors would like to thank M. Boussé and S. Hendrikx for proofreading the manuscript and N. Vervliet for valuable discussions.

References

[1] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Rev.*, vol. 38, p. 49–95, 3 1996.

[2] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," 3 2014.

[3] J. F. Sturm, "Using SEDUMI 1.02, a MATLAB* toolbox for optimization over symmetric cones," 2001.

[4] R. H. Tütüncü, K. C. Toh, and M. J. Todd, "Solving semidefinite-quadratic-linear programs using sdpt3," *Math. Program.*, vol. 95, pp. 189–217, 2003.

[5] N. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Trans. Signal Process.*, vol. 65, pp. 3551–3582, 7 2017.

[6] A. Cichocki, D. P. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. F. Caiafa, and A.-H. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Process. Mag.*, vol. 32, pp. 145–163, 3 2015.

[7] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, pp. 455–500, 8 2009.

[8] N. Vervliet, O. Debals, L. Sorber, and L. De Lathauwer, "Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis," *IEEE Signal Process. Mag.*, vol. 31, pp. 71–79, 9 2014.

[9] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer, "Tensorlab 3.0," 3 2016.

[10] N. Vervliet and L. De Lathauwer, "Numerical optimization based algorithms for data fusion," in *Data Fusion Methodology and Applications* (M. Cocchi, ed.), vol. 31, ch. 4, pp. 81–128, Elsevier, 2019.

[11] N. Vervliet, *Compressed sensing approaches to large-scale tensor decompositions*. PhD thesis, KU Leuven, 5 2018.

[12] W. Guo, I. Kotsia, and I. Patras, "Tensor learning for regression," *IEEE Trans. Image Process.*, vol. 21, pp. 816–827, 2 2012.

[13] H. Zhou, L. Li, and H. Zhu, "Tensor regression with applications in neuroimaging data analysis," *J. Amer. Statist. Assoc.*, vol. 108, pp. 540–552, 1 2013.

[14] X. Li, D. Xu, H. Zhou, and L. Li, "Tucker tensor regression and neuroimaging analysis," *Stat. Biosci.*, vol. 10, pp. 520–545, 3 2018.

[15] S. Hendrikx, M. Boussé, N. Vervliet, and L. De Lathauwer, "Algebraic and optimization based algorithms for multivariate regression using symmetric tensor decomposition," in *Proc. 2019 IEEE Int. Workshop on CAMSAP, Le Gosier, Guadeloupe, West Indies, France)*, pp. 475–479, 12 2019.

[16] K. Abed-Meraim, W. Qiu, and Y. Hua, "Blind system identification," *Proc. IEEE*, vol. 85, pp. 1310–1322, 8 1997.

[17] L. Sorber, M. Van Barel, and L. De Lathauwer, "Unconstrained optimization of real functions in complex variables," *SIAM J. Optim.*, vol. 22, pp. 879–898, 7 2012.

[18] O. Debals, *Tensorization and applications in blind source separation*. PhD thesis, KU Leuven, 8 2017.

[19] A. Cichocki, A.-H. Phan, Q. Zhao, N. Lee, I. Oseledets, M. Sugiyama, and D. P. Mandic, "Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives," *Found. Trends Mach. Learn.*, vol. 9, no. 6, pp. 431–673, 2017.

[20] M. Boussé, O. Debals, and L. De Lathauwer, "A tensor-based method for large-scale blind source separation using segmentation," *IEEE Trans. Signal Process.*, vol. 65, pp. 346–358, 1 2017.

[21] N. Vervliet, O. Debals, and L. De Lathauwer, "Exploiting efficient representations in large-scale tensor decompositions," *SIAM J. Sci. Comput.*, vol. 41, pp. A789–A815, 3 2019.

[22] M. D. Schatz, T.-M. Low, R. A. van de Geijn, and T. G. Kolda, "Exploiting symmetry in tensors for high performance," *SIAM J. Sci. Comput.*, vol. 36, pp. C453–C479, September 2014.

[23] X. Fu, S. Ibrahim, H.-T. Wai, C. Gao, and K. Huang, "Block-randomized stochastic proximal gradient for low-rank tensor factorization," *IEEE Trans. Signal Process.*, vol. 68, pp. 2170–2185, 2020.

[24] N. Vervliet and L. De Lathauwer, "A randomized block sampling approach to canonical polyadic decomposition of large-scale tensors," *IEEE J. Sel. Topics Signal Process.*, vol. 10, pp. 284–295, 2016.

[25] T. Adali, P. J. Schreier, and L. L. Scharf, "Complex-valued signal processing: The proper way to deal with impropriety," *IEEE Trans. Signal Process.*, vol. 59, no. 11, pp. 5101–5125, 2011.

[26] K. Kreutz-Delgado, "The complex gradient operator and the $\mathbb{CR}$-calculus [online]," 06 2009. arXiv:0906.4835.

[27] J. Nocedal and S. Wright, *Numerical optimization*. Springer New York, 7 2006.

[28] L. Sorber, M. Van Barel, and L. De Lathauwer, "Complex optimization toolbox v1.," 2 2013.

[29] R. Johnson, P. Schniter, T. J. Endres, J. D. Behm, D. R. Brown, and R. A. Casas, "Blind equalization using the constant modulus criterion: a review," *Proc. IEEE*, vol. 86, no. 10, pp. 1927–1950, 1998.

[30] L. Ljung, *System identification: Theory for the user*. Prentice hall, second ed., 1999.

[31] A.-J. van der Veen, "Algebraic methods for deterministic blind beamforming," *Proc. IEEE*, vol. 86, pp. 1987–2008, 10 1998.

[32] A.-J. van der Veen and A. Paulraj, "An analytical constant modulus algorithm," *IEEE Trans. Signal Process.*, vol. 44, pp. 1136–1155, 5 1996.

[33] L. De Lathauwer, "Algebraic techniques for the blind deconvolution of constant modulus signals," in *12th EUSIPCO Vienna, Austria*, pp. 225–228, 9 2004.

[34] V. Zarzoso and P. Comon, "Optimal step-size constant modulus algorithm," *IEEE Trans. Commun.*, vol. 56, no. 1, 2008.

[35] 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, p. e2190, 8 2018.