# New ALS Methods With Extrapolating Search Directions and Optimal Step Size for Complex-Valued Tensor Decompositions

Yannan Chen, Deren Han, and Liqun Qi

***Abstract*—In signal processing, data analysis and scientific computing, one often encounters the problem of decomposing a tensor into a sum of contributions. To solve such problems, both the search direction and the step size are two crucial elements in numerical algorithms, such as alternating least squares algorithm (ALS). Owing to the nonlinearity of the problem, the often used linear search direction is not always powerful enough. In this paper, we propose two higher-order search directions. The first one, geometric search direction, is constructed via a combination of two successive linear directions. The second one, algebraic search direction, is constructed via a quadratic approximation of three successive iterates. Then, in an enhanced line search along these directions, the optimal complex step size contains two arguments: modulus and phase. A current strategy is ELSCS that finds these two arguments alternately. So it may suffer from a local optimum. We broach a direct method, which determines these two arguments simultaneously, so as to obtain the global optimum. Finally, numerical comparisons on various search direction and step size schemes are reported in the context of blind separation-equalization of convolutive DS-CDMA mixtures. The results show that the new search directions have greatly improve the efficiency of ALS and the new step size strategy is competitive.**

***Index Terms*—Alternating least squares, block component model, CANDECOMP, CDMA, complex step size, PARAFAC, search direction, tensor decompositions.**

## I. Introduction

A $m$th-order tensor is a quantity of which the elements are addressed by $m$ indices. The problem of decomposing a higher-order tensor into a sum of products of lower-order tensors finds more and more important applications in signal processing, data analysis, and scientific computing [3], [10], [20], [26], [28]. This decomposition is a generalization of singular value decomposition (SVD) that gives a low-rank approximation to a matrix (a second-order tensor), while a direct generalization of SVD is nontrivial.

To describe the problem under consideration, we briefly introduce the CANDECOMP/PARAFAC (CP).

*Definition I.1: (CP):* A CP of a third-order tensor $\mathcal{Y} \in \mathbb{C}^{I\times J\times K}$ is a decomposition of $\mathcal{Y}$ as a linear combination of a minimal number of rank-1 tensors,

$$\mathcal{Y} = \sum_{r=1}^{R} \mathbf{h}_r \circ \mathbf{s}_r \circ \mathbf{a}_r,$$

where $\circ$ denotes the outer product, $\mathbf{h}_r$, $\mathbf{s}_r$ and $\mathbf{a}_r$ are the $r$th columns of three matrices $\mathbf{H} \in \mathbb{C}^{I\times R}$, $\mathbf{S} \in \mathbb{C}^{J\times R}$ and $\mathbf{A} \in \mathbb{C}^{K\times R}$, respectively.

Higher-order extension of CP is straightforward.

The CP can be dated back to the work of Hitchcock in 1927 [18], [19]. However it is out of the interests of researchers until the study of Carroll and Chang [4] and Harshman [17] in the fields of psychometries and phonetics in 1970, respectively. Recently, people find its applications in chemometrics [28] and independent component analysis [5], [11]. Moreover, Sidiropoulos *et al.* get its applications in wireless communication [27], where they propose a blind CP-based receiver for instantaneous CDMA mixtures impinging on an antenna array.

In several applications, a tensor $\mathcal{Y}$ might result from contributions that are not rank-1 tensors.

*Definition I.2: (Mode-n Product):* The mode-2 and mode-3 products of a third-order tensor $\mathcal{H} \in \mathbb{C}^{I\times L\times P}$ by the matrices $\mathbf{S} \in \mathbb{C}^{J\times L}$ and $\mathbf{A} \in \mathbb{C}^{K\times P}$, denoted by $\mathcal{H} \bullet_2 \mathbf{S}$ and $\mathcal{H} \bullet_3 \mathbf{A}$, result in an $(I \times J \times P)$ tensor and an $(I \times L \times K)$ tensor, respectively, with elements defined by

$$(\mathcal{H} \bullet_2 \mathbf{S})_{ijp} := \sum_{l=1}^{L} h_{ilp} s_{jl}, \quad (\mathcal{H} \bullet_3 \mathbf{A})_{ilk} := \sum_{p=1}^{P} h_{ilp} a_{kp}.$$

Nonrank-one contributions lead to another decomposition, block component model (BCM), which is more general than CP and is defined as follows.

*Definition I.3: (BCM):* A third-order tensor $\mathcal{Y} \in \mathbb{C}^{I\times J\times K}$ follows a BCM if it can be written as

$$\mathcal{Y} = \sum_{r=1}^{R} \mathcal{H}_r \bullet_2 \mathbf{S}_r \bullet_3 \mathbf{A}_r. \tag{1}$$

Manuscript received January 16, 2011; revised April 22, 2011 and July 28, 2011; accepted July 29, 2011. Date of publication August 15, 2011; date of current version November 16, 2011. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Hing Cheung So. The work of D. Han was supported by the NSFC Grants 11071122, 10871098 and by the Natural Science Foundation of Jiangsu Province Grant No. BK2009397. The work of L. Qi was supported by the Research Grant Council of Hong Kong.

Y. Chen is with the School of Mathematical Sciences and Key Laboratory for NSLSCS of Jiangsu Province, Nanjing Normal University, Nanjing 210046. He is also with the College of Science, Nanjing Forestry University, Nanjing 210037, China (e-mail: ynchen@njfu.edu.cn).

D. Han is with the School of Mathematical Sciences and Key Laboratory for NSLSCS of Jiangsu Province, Nanjing Normal University, Nanjing 210046, China (e-mail: handr00@hotmail.com).

L. Qi is with the Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong (e-mail: maqilq@polyu.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2011.2164911

The vectors $\mathbf{h}_r \in \mathbb{C}^{I\times 1}$, $\mathbf{s}_r \in \mathbb{C}^{J\times 1}$, and $\mathbf{a}_r \in \mathbb{C}^{K\times 1}$ of CP are now replaced by a tensor $\mathcal{H}_r \in \mathbb{C}^{I\times L\times P}$ and two matrices $\mathbf{S}_r \in \mathbb{C}^{J\times L}$ and $\mathbf{A}_r \in \mathbb{C}^{K\times P}$, respectively. This model is recently independently introduced by de Almeida *et al.* [8], [9] and by De Lathauwer [12], [13], [22], [25].

Computationally, ALS is a classical and unsophisticated method for CP [3], [17], [20], [28]. However, ALS needs a large number of iterations to converge due to the fact that the convergent rate of many iterations is almost null. To accelerate the rate of convergence of ALS, the line search along the incremental direction of an old iterate to a new one is proposed by Harshman [17]. For real-valued tensors, the optimal step size can be calculated because it is a solution of a polynomial equation with a single argument. This method is called "enhanced line search" (ELS) [6], [29], [30]. Since BCM is a generalization of CP, ALS for CP is generalized to compute BCM immediately [9], [14], [22], [25].

More recently, for complex-valued tensors that follow CP and BCM, Nion and De Lathauwer propose a line search scheme that is named as "enhanced line search with complex step" (ELSCS) [23], [24], where a complex step size factor $\rho := me^{i\theta}$ contains two independent arguments: modulus $m$ and phase $\theta$. In order to find the optimal step size, ELSCS proceeds in an alternating minimization manner, which consists of two steps: firstly, find the optimal modulus $m$ for a fixed phase $\theta$; secondly, find the optimal $\theta$ for a fixed $m$. That is to say, ELSCS needs to solve two polynomial equations alternately to update $m$ and $\theta$. After several iterations, a local optimal step size factor appears.

ELS and ELSCS can enhance the performance of ALS, which is verified numerically in [23], [24], [29], and [30]. Nonetheless, the current search direction in ELS and ELSCS is the difference between two successive previous iterates, which sometimes, due to the nonlinearity of the cost function in ALS, is not suitable to generate search directions that accelerate the rate of convergence greatly. Comon *et al.* give an example in Section V-B of [6] to illustrate the poorness of linear direction, and they assert that: "if the directions provided by the algorithm are bad, ELS cannot improve much on the convergence."

From the viewpoint of numerical optimization, the higher-order information is necessary to construct efficient search directions. Some gradient based optimization methods, including nonlinear conjugate-gradient, limited-memory BFGS and truncated Newton's method, are employed to exploit higher-order search directions for CP [1], [2]. However, if the gradient of the cost function is unknown or expensive to compute, such as the one considered in Section V where an argument matrix $\mathbf{S}_r$ has a special Toeplitz structure, these optimization methods may lose their advantages.

Based on ALS, we propose two extrapolating search directions containing higher-order information. The obtained search directions enjoy the advantages of preserving the structure of contributions and requiring slight additional costs of computation. At the same time the efficiency of the corresponding ALS methods can be enhanced greatly.

The first direction is called "geometric search direction", which is formed by adding the transformation of directions to the often used linear direction. The resulting direction is a linear combination of three previous iterates. When the combination parameter is set to zero, the geometric search direction reduces to a linear direction. The freedom in setting the combination parameter provides us the possibility of obtaining better search directions.

The second direction is named "algebraic search direction", where the higher-order information is collected by exploiting the quadratic extrapolation technique. The resulting direction is a convex combination of the quadratic extrapolating direction and the linear direction, and it is also a linear combination of three previous iterates.

After setting down the search direction, we proceed to find the modulus and phase of the optimal complex step size factor. Instead of ELSCS that finds them alternately, we broach a direct method that solves a system of two polynomial equations with two arguments simultaneously. Using the results of algebraic geometry [7], such system of polynomials has solutions if and only if its resultant vanishes, where the resultant is a determinant of a single argument polynomial matrix. We give a novel method to compute all roots of the resultant. Then, all the solutions of the polynomial system can be found immediately. Finally, the optimal step size factor is the one that minimizes the cost function restricted in the search direction.

This paper is organized as follows. The geometric and the algebraic search directions are presented in Section II. In Section III, we broach the technique of computing the optimal complex step size factor. The new ALS algorithm is described in Section IV. Some preliminary simulation results are reported in Section V. Finally, some concluding remarks are drawn in Section VI.

## II. Search Directions

### A. Preliminaries and the Linear Search Direction

Given a tensor $\mathcal{Y} \in \mathbb{C}^{I\times J\times K}$, a BCM of $\mathcal{Y}$ consists of the estimation of two matrices $\mathbf{S}_r \in \mathbb{C}^{J\times L}$, $\mathbf{A}_r \in \mathbb{C}^{K\times P}$ and a tensor $\mathcal{H}_r \in \mathbb{C}^{I\times L\times P}$ for $r = 1, \ldots, R$ such that definition (1) holds. Denote $\mathbf{S}$ and $\mathbf{A}$ as the $J \times RL$ and the $K \times RP$ matrices resulted from the concatenation of the $R$ matrices $\mathbf{S}_r$ and $\mathbf{A}_r$, respectively. Denote $\mathbf{H}$ as the $I \times RLP$ matrix whose elements are the elements of the tensors $\mathcal{H}_r$, which are stacked as follows $\mathbf{H}_{i,(r-1)LP+(l-1)P+p} := \mathcal{H}_r(i,l,p)$. Thus, a BCM of $\mathcal{Y}$ can be formulated by finding $\mathbf{S}$, $\mathbf{A}$ and $\mathbf{H}$ such that

$$\mathbf{Y} = (\mathbf{S} \odot_R \mathbf{A})\mathbf{H}^\top$$

where $\mathbf{Y}$ is the $JK \times I$ matrix representation of $\mathcal{Y}$ with elements defined as $\mathbf{Y}_{(j-1)K+k,i} := y_{ijk}$, and $\odot_R$ denotes the partition-wise Kronecker product of $\mathbf{S}$ and $\mathbf{A}$. This product results in a $JK \times RLP$ matrix defined by $\mathbf{S} \odot_R \mathbf{A} := [\mathbf{S}_1 \otimes \mathbf{A}_1, \ldots, \mathbf{S}_R \otimes \mathbf{A}_R]$, and $\otimes$ denotes the Kronecker product.

Computationally, the task is to find an estimator $\hat{\mathcal{Y}}$ of $\mathcal{Y}$, which builds from the estimators $\hat{\mathbf{S}}$, $\hat{\mathbf{A}}$, and $\hat{\mathbf{H}}$ of $\mathbf{S}$, $\mathbf{A}$, and $\mathbf{H}$, such that the following cost function

$$\phi = ||\mathcal{Y} - \hat{\mathcal{Y}}||_F^2 = ||\mathbf{Y} - (\hat{\mathbf{S}} \odot_R \hat{\mathbf{A}})\hat{\mathbf{H}}^\top||_F^2 \qquad (2)$$

is minimized. Here, $\|\cdot\|_F$ is the Frobenius norm. When $L = P = 1$, a BCM reduces to a CP, and the partition-wise Kronecker product $\odot_R$ is replaced by the Khatri-Rao product $\odot$ that is also called a column-wise Kronecker product.

Owing to the multilinearity of BCM in three estimators, people consider one estimator at a time in minimizing the cost function, while fixing the other two. Obviously, this is a linear least squares problem that is easy to solve. Then, three estimators are found by alternately solving these linear least squares problems. In this way, ALS algorithm appears.

Harshman observes via simulation that sometimes a large number of ALS iterates have almost null convergence rate and the estimators here increase gradually along fixed directions [17]. To accelerate the rate of convergence, at the $n$th iteration, he takes the increment in estimators as the linear search directions

$$\mathbf{G}_{\mathbf{lin}_S}^{(n)} := \hat{\mathbf{S}}^{(n-1)} - \hat{\mathbf{S}}^{(n-2)} \tag{3}$$

$$\mathbf{G}_{\mathbf{lin}_A}^{(n)} := \hat{\mathbf{A}}^{(n-1)} - \hat{\mathbf{A}}^{(n-2)} \tag{4}$$

$$\mathbf{G}_{\mathbf{lin}_H}^{(n)} := \hat{\mathbf{H}}^{(n-1)} - \hat{\mathbf{H}}^{(n-2)} \tag{5}$$

where $\hat{\mathbf{S}}^{(n-1)}$, $\hat{\mathbf{A}}^{(n-1)}$, and $\hat{\mathbf{H}}^{(n-1)}$ are the estimators of $\hat{\mathbf{S}}$, $\hat{\mathbf{A}}$, and $\hat{\mathbf{H}}$, respectively, obtained from the $(n-1)$th ALS iteration and $\hat{\mathbf{S}}^{(n-2)}$, $\hat{\mathbf{A}}^{(n-2)}$, and $\hat{\mathbf{H}}^{(n-2)}$ are the estimators obtained from the $(n-2)$th ALS iteration. These linear directions are widely used.

### B. Geometric Search Direction

Since the cost function (2) is obviously nonlinear, the search directions (3)–(5) based on linearization are not good enough to accelerate the rate of convergence of ALS. Now, we add some higher-order information to the linear search directions and get the new geometric extrapolating search directions.

We take an estimator $\hat{\mathbf{S}}$ for example, and the geometric search directions in the other two estimators $\hat{\mathbf{A}}$ and $\hat{\mathbf{H}}$ could be obtained in a similar way. Suppose at iteration $n(n \geq 3)$, three previous iterates $\hat{\mathbf{S}}^{(n-1)}$, $\hat{\mathbf{S}}^{(n-2)}$ and $\hat{\mathbf{S}}^{(n-3)}$ are known. At iterate $\hat{\mathbf{S}}^{(n-2)}$, $\mathbf{G}_{\mathbf{lin}_S}^{(n)} = \hat{\mathbf{S}}^{(n-1)} - \hat{\mathbf{S}}^{(n-2)}$ is a forward linear search direction and $\mathbf{G}_{\mathbf{lin}_S}^{(n-1)} = \hat{\mathbf{S}}^{(n-2)} - \hat{\mathbf{S}}^{(n-3)}$ is a backward linear search direction. From this point of view, the change from the backward direction to the forward direction is $\mathbf{G}_{\mathbf{lin}_S}^{(n)} - \mathbf{G}_{\mathbf{lin}_S}^{(n-1)}$. When the rate of convergence is slow, $\hat{\mathbf{S}}^{(n-1)}$ is not far from $\hat{\mathbf{S}}^{(n-2)}$. Therefore, we regard that the variation trend of the directions remains hold. So we add some variational information to the linear direction (3) and get the new geometric extrapolating search direction

$$\begin{aligned}\mathbf{G}_{\mathbf{geo}_S}^{(n)} &:= \mathbf{G}_{\mathbf{lin}_S}^{(n)} + \tau_1\left(\mathbf{G}_{\mathbf{lin}_S}^{(n)} - \mathbf{G}_{\mathbf{lin}_S}^{(n-1)}\right)\\ &= (1+\tau_1)\hat{\mathbf{S}}^{(n-1)} - (1+2\tau_1)\hat{\mathbf{S}}^{(n-2)}\\ &\quad + \tau_1\hat{\mathbf{S}}^{(n-3)}\end{aligned} \tag{6}$$

where $\tau_1$ is a positive parameter. When $\tau_1$ tends to zero, the geometric direction tends to the linear direction. Thus, a small value of $\tau_1$ is preferable. It is worthwhile to note that the geometric direction preserves the Toeplitz structure in an argument matrix $\mathbf{S}_r$.

In the same way, we can get the geometric directions of estimators $\hat{\mathbf{A}}$ and $\hat{\mathbf{H}}$

$$\begin{aligned}\mathbf{G}_{\mathbf{geo}_A}^{(n)} &:= (1+\tau_1)\hat{\mathbf{A}}^{(n-1)} - (1+2\tau_1)\hat{\mathbf{A}}^{(n-2)}\\ &\quad + \tau_1\hat{\mathbf{A}}^{(n-3)}\end{aligned} \tag{7}$$

$$\begin{aligned}\mathbf{G}_{\mathbf{geo}_H}^{(n)} &:= (1+\tau_1)\hat{\mathbf{H}}^{(n-1)} - (1+2\tau_1)\hat{\mathbf{H}}^{(n-2)}\\ &\quad + \tau_1\hat{\mathbf{H}}^{(n-3)}.\end{aligned} \tag{8}$$

### C. Algebraic Search Direction

Let us consider using the quadratic interpolation technique to extrapolate the new iterates and form the new algebraic extrapolating search directions.

We also take $\hat{\mathbf{S}}$ for example. Consider $\hat{\mathbf{S}}^{(n)}$ as a sample from a continuous path $\hat{\mathbf{S}}(t)$ in an unknown argument $t$, and the sequence of iterates $\{\hat{\mathbf{S}}^{(n)}\}$ locate in $\{t^{(n)}\}$ such that $\hat{\mathbf{S}}^{(n)} = \hat{\mathbf{S}}(t^{(n)})$. If we know the information about $t^{(n-1)}$, $t^{(n-2)}$, and $t^{(n-3)}$, we can form a quadratic approximation $\hat{\mathbf{S}}_q(t)$ to $\hat{\mathbf{S}}(t)$ by interpolating the three pieces of information available to obtain

$$\begin{aligned}\hat{\mathbf{S}}_q(t) := {}& \hat{\mathbf{S}}^{(n-1)}\frac{(t-t^{(n-2)})(t-t^{(n-3)})}{(t^{(n-1)}-t^{(n-2)})(t^{(n-1)}-t^{(n-3)})}\\ &+ \hat{\mathbf{S}}^{(n-2)}\frac{(t-t^{(n-1)})(t-t^{(n-3)})}{(t^{(n-2)}-t^{(n-1)})(t^{(n-2)}-t^{(n-3)})}\\ &+ \hat{\mathbf{S}}^{(n-3)}\frac{(t-t^{(n-1)})(t-t^{(n-2)})}{(t^{(n-3)}-t^{(n-1)})(t^{(n-3)}-t^{(n-2)})}.\end{aligned}$$

For simplicity, we make an additional assumption that the prediction of the new iterate $t^{(\mathrm{new})}$ and three previous iterates $t^{(n-1)}, t^{(n-2)}, t^{(n-3)}$ have the identical distances, that is to say, $t^{(\mathrm{new})} - t^{(n-1)} = t^{(n-1)} - t^{(n-2)} = t^{(n-2)} - t^{(n-3)}$. So the expression of $\hat{\mathbf{S}}_q(t^{(\mathrm{new})})$ is not dependent on $t$. Then, by some calculation, we have $\hat{\mathbf{S}}_q(t^{(\mathrm{new})}) = 3\hat{\mathbf{S}}^{(n-1)} - 3\hat{\mathbf{S}}^{(n-2)} + \hat{\mathbf{S}}^{(n-3)}$. The displacement from $\hat{\mathbf{S}}^{(n-2)}$ to $\hat{\mathbf{S}}_q(t^{(\mathrm{new})})$ forms the quadratic approximate search direction

$$\begin{aligned}\mathbf{G}_{\mathbf{qua}_S}^{(n)} &:= \hat{\mathbf{S}}_q(t^{(\mathrm{new})}) - \hat{\mathbf{S}}^{(n-2)}\\ &= 3\hat{\mathbf{S}}^{(n-1)} - 4\hat{\mathbf{S}}^{(n-2)} + \hat{\mathbf{S}}^{(n-3)}.\end{aligned} \tag{9}$$

The algebraic extrapolating search direction is the convex combination of the linear search direction (3) and the quadratic one (9)

$$\begin{aligned}\mathbf{G}_{\mathbf{alg}_S}^{(n)} &:= (1-\tau_2)\mathbf{G}_{\mathbf{lin}_S}^{(n)} + \tau_2\mathbf{G}_{\mathbf{qua}_S}^{(n)}\\ &= (1+2\tau_2)\hat{\mathbf{S}}^{(n-1)} - (1+3\tau_2)\hat{\mathbf{S}}^{(n-2)}\\ &\quad + \tau_2\hat{\mathbf{S}}^{(n-3)}\end{aligned} \tag{10}$$

where $\tau_2 \in [0,1]$. The Toeplitz structure in $\mathbf{S}_r$ is also preserved.

In the same way, we get the algebraic extrapolating search directions of $\hat{\mathbf{A}}$ and $\hat{\mathbf{H}}$,

$$\begin{aligned}\mathbf{G}_{\mathbf{alg}_A}^{(n)} &:= (1+2\tau_2)\hat{\mathbf{A}}^{(n-1)} - (1+3\tau_2)\hat{\mathbf{A}}^{(n-2)}\\ &\quad + \tau_2\hat{\mathbf{A}}^{(n-3)}\end{aligned} \tag{11}$$

$$\begin{aligned}\mathbf{G}_{\mathbf{alg}_H}^{(n)} &:= (1+2\tau_2)\hat{\mathbf{H}}^{(n-1)} - (1+3\tau_2)\hat{\mathbf{H}}^{(n-2)}\\ &\quad + \tau_2\hat{\mathbf{H}}^{(n-3)}.\end{aligned} \tag{12}$$

Note that, the geometric direction (6) is a linear combination of the linear direction (3) and the quadratic direction (9) with coefficients $(1-2\tau_1)$ and $\tau_1$, respectively. Therefore, both the geometric directions and the algebraic ones have some second-order information. Moreover, the computational cost of them is ignorable in numerical algorithms.

## III. The Optimal Step Size

At iteration $n$, when the search directions $\mathbf{G}_S^{(n)}$, $\mathbf{G}_A^{(n)}$ and $\mathbf{G}_H^{(n)}$ are determined, we perform a line search process

$$\hat{\mathbf{S}}^{(\text{new})} = \hat{\mathbf{S}}^{(n-2)} + \rho \mathbf{G}_S^{(n)} \tag{13}$$

$$\hat{\mathbf{A}}^{(\text{new})} = \hat{\mathbf{A}}^{(n-2)} + \rho \mathbf{G}_A^{(n)} \tag{14}$$

$$\hat{\mathbf{H}}^{(\text{new})} = \hat{\mathbf{H}}^{(n-2)} + \rho \mathbf{G}_H^{(n)} \tag{15}$$

where $\rho$ is a step size factor and the directions here can be the geometric search directions (6)–(8) or the algebraic search directions (10)–(12). Then, started from $\hat{\mathbf{S}}^{(\text{new})}$, $\hat{\mathbf{A}}^{(\text{new})}$, and $\hat{\mathbf{H}}^{(\text{new})}$, ALS iterates are performed to get the next estimators $\hat{\mathbf{S}}^{(n)}$, $\hat{\mathbf{A}}^{(n)}$ and $\hat{\mathbf{H}}^{(n)}$.

In the line search framework, the step size scheme affects the rate of convergence of a numerical algorithm. Harshman choose a fixed value between 1.2 and 1.3 to be the step size factor [17]. For real-valued tensors, Rajih *et al.* introduce the ELS that calculate the optimal step size factor by rooting a polynomial [29], [30]. For the complex-valued tensors, Nion and De Lathauwer propose ELSCS that is an alternating polynomial rooting algorithm to approximate the optimal complex step size factor [23], [24].

The optimal complex step size factor is the minimum of the following objective function in a single complex argument $\rho$

$$\begin{aligned}\phi^{(n)} &:= \|\mathbf{Y} - (\hat{\mathbf{S}}^{(\text{new})} \odot_R \hat{\mathbf{A}}^{(\text{new})})\hat{\mathbf{H}}^{(\text{new})^\top}\|_F^2 \\ &= \|\mathbf{Y} - \left((\hat{\mathbf{S}}^{(n-2)} + \rho \mathbf{G}_S^{(n)}) \odot_R (\hat{\mathbf{A}}^{(n-2)} + \rho \mathbf{G}_A^{(n)})\right) \\ &\quad \cdot (\hat{\mathbf{H}}^{(n-2)} + \rho \mathbf{G}_H^{(n)})^\top\|_F^2. \end{aligned} \tag{16}$$

Denoting the $JK \times I$ matrices $\mathbf{T}_3$, $\mathbf{T}_2$, $\mathbf{T}_1$, and $\mathbf{T}_0$ as (where the superscripts $(n)$ and $(n-2)$ are omitted for convenience of notation)

$$\begin{cases} \mathbf{T}_3 := (\mathbf{G}_S \odot_R \mathbf{G}_A)\mathbf{G}_H^\top \\ \mathbf{T}_2 := (\mathbf{S} \odot_R \mathbf{G}_A + \mathbf{G}_S \odot_R \mathbf{A})\mathbf{G}_H^\top + (\mathbf{G}_S \odot_R \mathbf{G}_A)\mathbf{H}^\top \\ \mathbf{T}_1 := (\mathbf{S} \odot_R \mathbf{A})\mathbf{G}_H^\top + (\mathbf{S} \odot_R \mathbf{G}_A + \mathbf{G}_S \odot_R \mathbf{A})\mathbf{H}^\top \\ \mathbf{T}_0 := (\mathbf{S} \odot_R \mathbf{A})\mathbf{H}^\top - \mathbf{Y}. \end{cases} \tag{17}$$

Then the objective function (16) can be rewritten in a compact form as

$$\begin{aligned}\phi^{(n)} &= \|\rho^3\mathbf{T}_3 + \rho^2\mathbf{T}_2 + \rho\mathbf{T}_1 + \mathbf{T}_0\|_F^2 \\ &= \|\mathbf{T}\mathbf{u}\|_F^2 = \mathbf{u}^*\mathbf{T}^*\mathbf{T}\mathbf{u} \end{aligned} \tag{18}$$

where

$$\mathbf{T} := [\text{Vec}(\mathbf{T}_3), \text{Vec}(\mathbf{T}_2), \text{Vec}(\mathbf{T}_1), \text{Vec}(\mathbf{T}_0)] \tag{19}$$

is an $IJK \times 4$ matrix, Vec is the operator that writes a matrix $\mathbf{A} \in \mathbb{C}^{I\times J}$ in a vector format by concatenation of the columns such that $[\text{Vec}(\mathbf{A})]_{i+(j-1)I} := \mathbf{A}_{ij}$, $\mathbf{u} := (\rho^3, \rho^2, \rho, 1)^\top$ and the superscript $^*$ denotes the conjugate transpose.

Define a $4 \times 4$ matrix $\Delta := \mathbf{T}^*\mathbf{T}$. Then it can be represented by a componentwise form $\Delta_{mn} := \alpha_{mn} + i\beta_{mn}$ for $m, n = 1, 2, 3, 4$, where $\alpha_{mn} = \alpha_{nm}$ and $\beta_{mn} = -\beta_{nm}$.

Let $\rho := me^{i\theta} = m(\cos\theta + i\sin\theta)$ be a complex step size factor, where $m$ and $\theta$ are its modulus and phase, respectively. Define $t := \tan\left(\frac{\theta}{2}\right)$, then $\cos\theta = \frac{(1-t^2)}{(1+t^2)}$ and $\sin\theta = \frac{2t}{(1+t^2)}$. The task is now to find the optimal $m$ and $\theta$ that minimize the objective function (18).

The partial derivative of $\phi^{(n)}$ with respect to $m$ can be expressed as

$$\frac{\partial\phi^{(n)}}{\partial m}\Big|_{\tan(\frac{\theta}{2})=t} = \frac{\mathbf{u}_t^\top \mathbf{C} \mathbf{u}_m}{(1+t^2)^3} \tag{20}$$

where $\mathbf{u}_t := (t^6, t^5, \ldots, 1)^\top$, $\mathbf{u}_m := (m^5, m^4, \ldots, 1)^\top$ and $\mathbf{C}$ is a $7 \times 6$ matrix (21), shown at the bottom of the page. On the other hand, the partial derivative of $\phi^{(n)}$ with respect to $\theta$ can be expressed as

$$\frac{\partial\phi^{(n)}}{\partial \theta}\Big|_{\tan(\frac{\theta}{2})=t} = \frac{\mathbf{u}_t^\top \mathbf{D} \mathbf{u}_m}{(1+t^2)^3} \tag{22}$$

where $\mathbf{D}$ is a $7 \times 6$ matrix, shown in (23) at the bottom of the next page.

$$\mathbf{C} := \begin{pmatrix} 6\alpha_{11} & -10\alpha_{12} & 8\alpha_{13}+4\alpha_{22} & -6\alpha_{14}-6\alpha_{23} & 4\alpha_{24}+2\alpha_{33} & -2\alpha_{34} \\ 0 & 20\beta_{12} & -32\beta_{13} & 36\beta_{14}+12\beta_{23} & -16\beta_{24} & 4\beta_{34} \\ 18\alpha_{11} & -10\alpha_{12} & -40\alpha_{13}+12\alpha_{22} & 90\alpha_{14}-6\alpha_{23} & -20\alpha_{24}+6\alpha_{33} & -2\alpha_{34} \\ 0 & 40\beta_{12} & 0 & -120\beta_{14}+24\beta_{23} & 0 & 8\beta_{34} \\ 18\alpha_{11} & 10\alpha_{12} & -40\alpha_{13}+12\alpha_{22} & -90\alpha_{14}+6\alpha_{23} & -20\alpha_{24}+6\alpha_{33} & 2\alpha_{34} \\ 0 & 20\beta_{12} & 32\beta_{13} & 36\beta_{14}+12\beta_{23} & 16\beta_{24} & 4\beta_{34} \\ 6\alpha_{11} & 10\alpha_{12} & 8\alpha_{13}+4\alpha_{22} & 6\alpha_{14}+6\alpha_{23} & 4\alpha_{24}+2\alpha_{33} & 2\alpha_{34} \end{pmatrix}. \tag{21}$$

The optimal condition for a complex $\rho = me^{i\theta}$ to be the optimal step size factor is that

$$\frac{\partial \phi^{(n)}}{\partial m} = \frac{\partial \phi^{(n)}}{\partial \theta} = 0.$$

From the partial derivatives (20) and (22), the above optimal condition holds if and only if $(m, t)$ is a solution of the following system of two polynomial equations:

$$\begin{cases} f(m,t) := \mathbf{u}_t^\top \mathbf{C}\mathbf{u}_m = 0 \\ g(m,t) := \mathbf{u}_t^\top \mathbf{D}\mathbf{u}_m = 0. \end{cases} \tag{24}$$

Then, we can regard $f$ and $g$ as polynomials in $m$ with some polynomials in $t$ as its coefficients. That is to say, we rewrite (24) as

$$\begin{cases} f(m,t) = c_1(t)m^5 + c_2(t)m^4 + \cdots + c_6(t) = 0 \\ g(m,t) = d_1(t)m^5 + d_2(t)m^4 + \cdots + d_6(t) = 0 \end{cases} \tag{25}$$

where $c_k(t) := \mathbf{u}_t^\top \mathbf{C}\mathbf{e}_k$ and $d_k(t) := \mathbf{u}_t^\top \mathbf{D}\mathbf{e}_k$ are sixth-order polynomials in $t$, and $\mathbf{e}_k$ is the $k$th column of an identity matrix.

By the Sylvester theorem [7], (25) has solutions if and only if its resultant vanishes. The resultant here is a determinant of the following $10 \times 10$ polynomial matrix:

$$\begin{pmatrix} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & 0 & 0 & 0 & 0 \\ 0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & 0 & 0 & 0 \\ 0 & 0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & 0 & 0 \\ 0 & 0 & 0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & 0 \\ 0 & 0 & 0 & 0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \\ d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & 0 & 0 & 0 & 0 \\ 0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & 0 & 0 & 0 \\ 0 & 0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & 0 & 0 \\ 0 & 0 & 0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & 0 \\ 0 & 0 & 0 & 0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \end{pmatrix}.$$

The useful roots of this resultant are solved as follows. First, notice that $d_6 = 0$. Then

$$t = \frac{\beta_{34} \pm \sqrt{\beta_{34}^2 + \alpha_{34}^2}}{\alpha_{34}}, \qquad m = 0$$

are solutions of (25) with objective function value $\alpha_{44}$. These two roots are useless, since their modului are zeros. The other roots of the resultant can be found via solving a determinant of the following $9 \times 9$ polynomial matrix

$$\Upsilon := \begin{pmatrix} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & 0 & 0 & 0 \\ 0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & 0 & 0 \\ 0 & 0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & 0 \\ 0 & 0 & 0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \\ d_1 & d_2 & d_3 & d_4 & d_5 & 0 & 0 & 0 & 0 \\ 0 & d_1 & d_2 & d_3 & d_4 & d_5 & 0 & 0 & 0 \\ 0 & 0 & d_1 & d_2 & d_3 & d_4 & d_5 & 0 & 0 \\ 0 & 0 & 0 & d_1 & d_2 & d_3 & d_4 & d_5 & 0 \\ 0 & 0 & 0 & 0 & d_1 & d_2 & d_3 & d_4 & d_5 \end{pmatrix}. \tag{26}$$

Obviously, the determinant $\det(\Upsilon)$ is a 54th-order polynomial in $t$, due to the polynomials $c_k$ for $k = 1, \ldots, 6$ and $d_k$ for $k = 1, \ldots, 5$ are all of order six. The 54th-order polynomial here suffers from a trouble that its coefficients are complicated and unobtainable.

We give a novel method to compute all the roots of the determinant $\det(\Upsilon)$. Since all the elements of the matrix $\Upsilon$ are sixth-order polynomials in $t$ except some zeros, we rewrite $\Upsilon$ as

$$\Upsilon = \Gamma_1 t^6 + \Gamma_2 t^5 + \Gamma_3 t^4 + \Gamma_4 t^3 + \Gamma_5 t^2 + \Gamma_6 t + \Gamma_7$$

where the elements of $\Gamma_k$ are the coefficients of $(7-k)$th-order term of the corresponding elements of $\Upsilon$. That is to say

$$\Gamma_k := \begin{pmatrix} C_{k1} & C_{k2} & C_{k3} & C_{k4} & C_{k5} & C_{k6} & 0 & 0 & 0 \\ 0 & C_{k1} & C_{k2} & C_{k3} & C_{k4} & C_{k5} & C_{k6} & 0 & 0 \\ 0 & 0 & C_{k1} & C_{k2} & C_{k3} & C_{k4} & C_{k5} & C_{k6} & 0 \\ 0 & 0 & 0 & C_{k1} & C_{k2} & C_{k3} & C_{k4} & C_{k5} & C_{k6} \\ D_{k1} & D_{k2} & D_{k3} & D_{k4} & D_{k5} & 0 & 0 & 0 & 0 \\ 0 & D_{k1} & D_{k2} & D_{k3} & D_{k4} & D_{k5} & 0 & 0 & 0 \\ 0 & 0 & D_{k1} & D_{k2} & D_{k3} & D_{k4} & D_{k5} & 0 & 0 \\ 0 & 0 & 0 & D_{k1} & D_{k2} & D_{k3} & D_{k4} & D_{k5} & 0 \\ 0 & 0 & 0 & 0 & D_{k1} & D_{k2} & D_{k3} & D_{k4} & D_{k5} \end{pmatrix} \tag{27}$$

for $k = 1, 2, \ldots, 7$.

Then, we have the following important theorem, which is an extension of the method in [15]. And it is the support theory of a Matlab function roots when $M \equiv 1$ [21].

$$\mathbf{D} := \begin{pmatrix} -2\beta_{12} & 4\beta_{13} & -6\beta_{14} - 2\beta_{23} & 4\beta_{24} & -2\beta_{34} & 0 \\ -4\alpha_{12} & 16\alpha_{13} & -36\alpha_{14} - 4\alpha_{23} & 16\alpha_{24} & -4\alpha_{34} & 0 \\ -2\beta_{12} & -20\beta_{13} & 90\beta_{14} - 2\beta_{23} & -20\beta_{24} & -2\beta_{34} & 0 \\ -8\alpha_{12} & 0 & 120\alpha_{14} - 8\alpha_{23} & 0 & -8\alpha_{34} & 0 \\ 2\beta_{12} & -20\beta_{13} & -90\beta_{14} + 2\beta_{23} & -20\beta_{24} & 2\beta_{34} & 0 \\ -4\alpha_{12} & -16\alpha_{13} & -36\alpha_{14} - 4\alpha_{23} & -16\alpha_{24} & -4\alpha_{34} & 0 \\ 2\beta_{12} & 4\beta_{13} & 6\beta_{14} + 2\beta_{23} & 4\beta_{24} & 2\beta_{34} & 0 \end{pmatrix}. \tag{23}$$

*Theorem III.1:* Suppose $\Gamma_k$ for $k = 1, 2, \ldots, N+1$ are $M \times M$ matrices, and $\Gamma_1$ is nonsingular. Then all the roots of the $MN$th-order polynomial equation

$$\det\left(\Gamma_1 t^N + \Gamma_2 t^{N-1} + \cdots + \Gamma_N t + \Gamma_{N+1}\right) = 0$$

are exactly the eigenvalues of the following $MN \times MN$ matrix

$$\begin{pmatrix} -\Gamma_1^{-1}\Gamma_2 & -\Gamma_1^{-1}\Gamma_3 & \cdots & -\Gamma_1^{-1}\Gamma_N & -\Gamma_1^{-1}\Gamma_{N+1} \\ I & 0 & \cdots & 0 & 0 \\ 0 & I & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & I & 0 \end{pmatrix} \tag{28}$$

where $I$ is a $M \times M$ identity matrix.

From this theorem, all the roots in $t$ of the determinant $\det(\Upsilon)$ can be found via computing all the eigenvalues of the matrix (28) with $M = 9$ and $N = 6$. Hence, it is stable. Then, we ignore complex-valued roots since they are meaningless. For each real-valued root $t$, we substitute it into the polynomials system (25) and solve the system to get the corresponding modului $m$. After this, we find all the critical points of the objective function $\phi^{(n)}$. Finally, the optimal step size factor is the one that gives the smallest objective function value.

To state succinctly, we sum up an algorithm.

---

Subroutine for the optimal complex step size.

---

**Given**: iterates $\hat{\mathbf{S}}^{(n-2)}$, $\hat{\mathbf{A}}^{(n-2)}$ and $\hat{\mathbf{H}}^{(n-2)}$, and directions $\mathbf{G}_S^{(n)}$, $\mathbf{G}_A^{(n)}$ and $\mathbf{G}_H^{(n)}$.

**S1.** Compute $\Delta = \mathbf{T}^*\mathbf{T}$, where $\mathbf{T}$ is from **(17)** and **(19)**.

**S2.** Calculate $\mathbf{C}$ and $\mathbf{D}$ in **(21)** and **(23)**, respectively.

**S3.** Form $\Gamma_p$ for $p = 1, \ldots, 7$ in **(27)** and build **(28)**.

**S4.** Compute all the eigenvalues $t$ of **(28)**.

**S5.** For each real $t$, find the corresponding real $m$ by **(25)**. Then, the optimal step size is the one with the minimal objective function value.

The matrix $\Delta$ is necessary for both ELSCS and the method presented here. Computing $\mathbf{T}_i$ for $i = 3, 2, 1, 0$ in (17) requires $4JKRLP + 12IJKRLP$ flops. Then, computing $\Delta$ requires $20IJK$ flops because of symmetry. Therefore, the total number of operations for $\Delta$ is $\mathcal{O}(IJKRLP)$.

The computational cost after $\Delta$ of the optimal complex step size method here comes mainly from the computation of all the eigenvalues of a $54 \times 54$ matrix (28), where about $10 \cdot 54^3$ flops are required theoretically [16].

On the other hand, ELSCS requires a few alternating minimizations between $m$ and $t$ to get a step size factor. Thus, its cost is computing the roots of some fifth- and sixth-order polynomials, which needs $\mathcal{O}(10 \cdot 6^3)$ flops. It's cheap.

We give a heuristic strategy to compute an approximate optimal step size factor, since computing all the eigenvalues of the $54 \times 54$ matrix (28) is time-consuming when compared with ELSCS. In the context of line search, we believe the phase $\theta$ is close to zero, so the parameter $t = \tan\left(\frac{\theta}{2}\right)$ is also close to zero. Omitting some higher order terms of polynomial $\det(\Upsilon)$ should not greatly affect the optimum. Hence, for the purpose of easily computing and resulting in a real-valued root, we truncate the polynomial to a linear function. In this case, we only need to compute the constant and linear terms of $\det(\Upsilon)$. The constant term is a determinant $\det(\Gamma_7)$. The linear term is a sum of determinants $\det(D_i)$ for $i = 1, 2, \ldots, 9$, where $D_i$ is a $9 \times 9$ matrix obtained by replacing the elements of $i$th row of $\Gamma_7$ by the corresponding elements of $\Gamma_6$. Hence, the computational cost is ten $9 \times 9$ determinants that needs $10 \cdot \frac{2}{3}9^3$ flops. Whenever the parameter $t$ is determined, we compute the optimal modulus $m$ by ELS that is used in ELSCS. We argue that this heuristic scheme is as cheap as ELSCS at least.

## IV. The New ALS Algorithm

Based on the analysis above, we give the new ALS algorithm with extrapolating search direction and optimal step size schemes.

---

New ALS

---

**Initialize**: Choose three iterates $\hat{\mathbf{S}}^{(0)}$, $\hat{\mathbf{S}}^{(1)}$, $\hat{\mathbf{S}}^{(2)}$, $\hat{\mathbf{A}}^{(0)}$, $\hat{\mathbf{A}}^{(1)}$, $\hat{\mathbf{A}}^{(2)}$, $\hat{\mathbf{H}}^{(0)}$, $\hat{\mathbf{H}}^{(1)}$, $\hat{\mathbf{H}}^{(2)}$, and set $n = 3$.

**while** $\|\hat{\mathcal{Y}}^{(n-1)} - \hat{\mathcal{Y}}^{(n-2)}\|_F > \epsilon$ (e.g., $\epsilon = 10^{-6}$) **do**

**S1.** Compute directions **(6)**–**(8)** or **(10)**–**(12)**.

**S2.** Find the optimal step size factor $\rho^{(n)}$.

**S3.If** $\phi^{(n)}(\rho^{(n)}) \leq \phi^{(n)}(1)$,

build $\hat{\mathbf{A}}^{(\mathrm{new})}$ and $\hat{\mathbf{H}}^{(\mathrm{new})}$ from **(14)** and **(15)**;

**else**

set $\hat{\mathbf{A}}^{(\mathrm{new})} = \hat{\mathbf{A}}^{(n-1)}$ and $\hat{\mathbf{H}}^{(\mathrm{new})} = \hat{\mathbf{H}}^{(n-1)}$.

**S4.** Update $\hat{\mathbf{S}}^{(n)}$ by $\hat{\mathbf{A}}^{(\mathrm{new})}$ and $\hat{\mathbf{H}}^{(\mathrm{new})}$.

**S5.** Update $\hat{\mathbf{A}}^{(n)}$ by $\hat{\mathbf{S}}^{(n)}$ and $\hat{\mathbf{H}}^{(\mathrm{new})}$.

**S6.** Update $\hat{\mathbf{H}}^{(n)}$ by $\hat{\mathbf{S}}^{(n)}$ and $\hat{\mathbf{A}}^{(n)}$.

**S7.** Set $n := n + 1$.

**end**

The ALS iterations including **S4**-**S6** are the same as the ones presented in [22] and [25]. In the remainder of this section, we go to the details and analyze their computational complexity. For the convenience of notation, the superscripts are omitted.

First, update $\mathbf{S}$, when $\mathbf{A}$ and $\mathbf{H}$ are known. To preserve the Toeplitz structure, we update the generator vector $\mathbf{s}_r \in \mathbb{C}^{J+L-1}$ of matrix $\mathbf{S}_r \in \mathbb{C}^{J \times L}$. From (1), $\mathcal{Y} = \sum_{r=1}^{R} \mathcal{H}_r \bullet_2 \mathbf{S}_r \bullet_3 \mathbf{A}_r$, where $\mathcal{Y} \in \mathbb{C}^{I \times J \times K}$, $\mathcal{H}_r \in \mathbb{C}^{I \times L \times P}$, and $\mathbf{A}_r \in \mathbb{C}^{K \times P}$. Computing $\mathcal{G}_r := \mathcal{H}_r \bullet_3 \mathbf{A}_r \in \mathbb{C}^{I \times L \times K}$ requires $\mathcal{O}(PILK)$ flops. Then $\mathcal{Y} = \sum_{r=1}^{R} \mathcal{G}_r \bullet_2 \mathbf{S}_r$. Define $\mathbf{Y}_k := \mathcal{Y}(:, :, k) \in \mathbb{C}^{I \times J}$ and $\mathbf{G}_{rk} := \mathcal{G}_r(:, :, k) \in \mathbb{C}^{I \times L}$ as the frontal slice of the correspond tensors, then $\mathbf{Y}_k = \sum_{r=1}^{R} \mathbf{G}_{rk}\mathbf{S}_r^\top$. Rearrange $\mathbf{G}_{rk}$ to get $\mathbf{M}_{rk}$ such that $\mathrm{Vec}(\mathbf{G}_{rk}\mathbf{S}_r^\top) = \mathbf{M}_{rk}\mathbf{s}_r$, where $\mathbf{M}_{rk} \in$

$\mathbb{C}^{JI\times(J+L-1)}$ is a sparse matrix with at most $LI$ nonzeros in each column. Define

$$\mathbf{M} := \begin{pmatrix} \mathbf{M}_{11} & \cdots & \mathbf{M}_{R1} \\ \vdots & & \vdots \\ \mathbf{M}_{1K} & \cdots & \mathbf{M}_{RK} \end{pmatrix}, \quad \mathbf{s} := \begin{pmatrix} \mathbf{s}_1 \\ \vdots \\ \mathbf{s}_R \end{pmatrix}$$

and $\mathbf{Y}^{(KJI\times 1)}$ with $[\mathbf{Y}^{(KJI\times 1)}]_{i+(j-1)I+(k-1)JI} := \mathcal{Y}_{ijk}$. Then

$$\mathbf{Y}^{(KJI\times 1)} = \mathbf{M}\mathbf{s}$$

where $\mathbf{M} \in \mathbb{C}^{KJI\times R(J+L-1)}$ is a sparse matrix with at most $KLI$ nonzeros in each column. Solving the equation by sparse matrix algorithm requires $\mathcal{O}\left(KIL\cdot R^2(J+L-1)^2\right)$ flops. Therefore, the total number of operations for updating $\mathbf{S}$ is $\mathcal{O}\left(RLPIK + R^2LI(J+L-1)^2K\right)$.

Second, update $\mathbf{A}$, where $\mathbf{S}$ and $\mathbf{H}$ are known. Computing $\mathcal{Q}_r := \mathcal{H}_r \bullet_2 \mathbf{S}_r \in \mathbb{C}^{I\times J\times P}$ requires $\mathcal{O}(LIJP)$ flops. Then $\mathcal{Y} = \sum_{r=1}^{R} \mathcal{Q}_r \bullet_3 \mathbf{A}_r$. Rearranging the tensors here as a matrix form, we get $\mathbf{Y}^{(JI\times K)} = \sum_{r=1}^{R} \mathbf{Q}_r^{(JI\times P)}\mathbf{A}_r^\top = \mathbf{Q}\mathbf{A}^\top$, where $[\mathbf{Y}^{(JI\times K)}]_{i+(j-1)I,k} := \mathcal{Y}_{ijk}$, $[\mathbf{Q}_r^{(JI\times P)}]_{i+(j-1)I,p} := [\mathcal{Q}_r]_{ijp}$ and $\mathbf{Q} := (\mathbf{Q}_1,\ldots,\mathbf{Q}_R) \in \mathbb{C}^{JI\times RP}$. Then $\mathbf{A} = (\mathbf{Q}^\dagger\mathbf{Y}^{(JI\times K)})^\top$. Computing the pseudoinverse matrix $\mathbf{Q}^\dagger$ and the matrix multiplication require $\mathcal{O}\left((JI)^2RP\right)$ and $\mathcal{O}(RPIJK)$ flops, respectively. Thus, there are $\mathcal{O}\left((JI)^2RP + RPIJK\right)$ operations.

Finally, update $\mathbf{H}$, where $\mathbf{S}$ and $\mathbf{A}$ are known. Since $\mathbf{Y}^{(JK\times I)} = (\mathbf{S}\odot_R\mathbf{A})\mathbf{H}^\top$, computing $\mathbf{Z} := \mathbf{S}\odot_R\mathbf{A} \in \mathbb{C}^{JK\times RLP}$ and its pseudoinverse matrix require $\mathcal{O}(RLPJK)$ and $\mathcal{O}\left((JK)^2RLP\right)$ flops, respectively. Then computing $\mathbf{H} = (\mathbf{Z}^\dagger\mathbf{Y}^{(JK\times I)})^\top$ requires $\mathcal{O}(RLPIJK)$ flops. Totally, $\mathcal{O}\left((JK)^2RLP + RLPIJK\right)$ flops are required.

## V. Simulation Results

We are going to examine the performance of the new search directions and the new step size. For these purpose, we apply these techniques to solve the problem of blind separation-equalization of convolutive DS-CDMA mixtures received by an antenna array after multipath propagation, and compare them with the basic ALS and the ALS with ELSCS.

We make the same assumptions as those in [22], [24], and [25], i.e., we assume that the signal of the $r$th user is subject to inter-symbol-interference over $L$ consecutive symbols and that this signal arrives at antenna array via $P$ specular paths. For user $r$, $r = 1,\ldots,R$, the $I\times L$ frontal slice $\mathcal{H}_r(:,:,p)$ of $\mathcal{H}_r$ collects samples of the convolved spreading waveform associated to the $p$th path, $p = 1,\ldots,P$. The $J\times L$ matrix $\mathbf{S}_r$ holds the $J$ transmitted symbols and has a Toeplitz structure. The $K\times P$ matrix $\mathbf{A}_r$ collects the response of the $K$ antennas according to the angles of arrival of the $P$ paths.

In our experiments, we consider $R = 4$ users, pseudorandom spreading codes of length $I = 8$, a short frame of $J = 50$ QPSK symbols, $K = 4$ antennas, $L = 2$ interfering symbols, and $P = 2$ paths per user. The signal-to-noise ratio (SNR) at the input of the BCM receiver is defined by $\mathrm{SNR} = 10\log_{10}\left(\frac{\|\mathcal{Y}\|_F^2}{\|\mathcal{N}\|_F^2}\right)$, where $\mathcal{Y}$ is the complex-valued noise-free tensor of observations and $\mathcal{N}$ is the tensor of zero-mean white (in all dimensions) Gaussian noise. We run 1000 Monte Carlo experiments. For each one, the initialization of the iterate is performed by selecting the best one from ten different random starting points, which obey the standard Gaussian distribution. We stop the algorithm when the iteration number exceeds 500 or the stopping tolerance $\|\hat{\mathcal{Y}}^{(n)} - \hat{\mathcal{Y}}^{(n-1)}\|_F \le \epsilon$ is satisfied, where $\epsilon = 10^{-6}$.

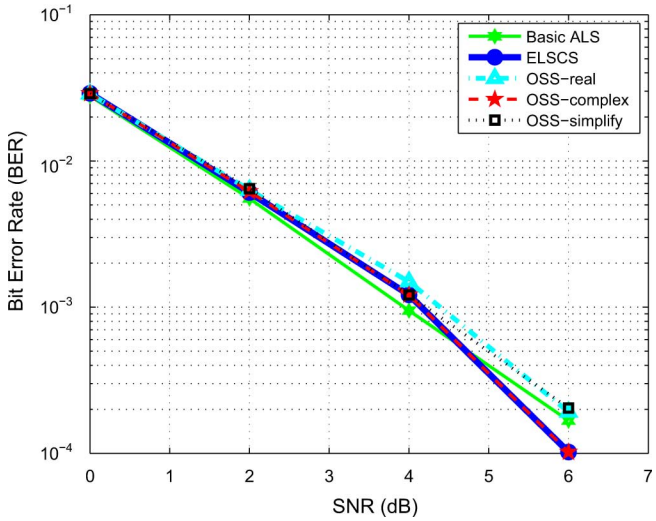



Fig. 1. BER versus SNR.

### A. The First Result: Step Size

In line search methods, the step size is important. Here, in ALS framework, we test five sorts of step size strategies with the same linear search directions.

- "Basic ALS", whose step size factor is always one.
- "ELSCS" that seeks the complex step size factor in an alternating manner [24].
- "OSS-real" that finds the optimal real step size factor.
- "OSS-complex" is the optimal complex step size.
- And "OSS-simplify" is the heuristic implementation of the optimal complex step size scheme described in the last paragraph of Section III.

Next, we give the results based on statistics.

*Which line search scheme gives the most accurate signal estimation?* We show the bit error rate (BER) over all users in Fig. 1. First, OSS-complex works as good as ELSCS. Second, compared with them, OSS-real and OSS-simplify work a little bit worse. We recall that the imaginary part of the step size factor of OSS-real and OSS-simplify is zero and a heuristic guesswork, respectively. Thus, iterates are easy to be trapped around a false local optimum, sometimes.

*Which line search scheme is the fastest one?* We illustrate in Table I the possibility of each step size strategy being the winner of these five strategies from a numerical viewpoint. The probability that OSS-simplify wins on a given problem in terms of iteration number is about 72.6%, and the corresponding probability in terms of CPU time is about 93.3%. We say two algorithms have the same CPU time if the difference of the corresponding CPU times is less than a tenth of a second. This rule is also valid in Table II.

TABLE I
THE POSSIBILITY THAT EACH STEP SIZE STRATEGY WINS (%)

| | Basic | | OSS- | | |
|---|---|---|---|---|---|
| | ALS | ELSCS | real | complex | simplify |
| Iter. | 0.2 | 39.1 | 20.2 | 30.8 | 72.6 |
| Times | 0.6 | 58.8 | 51.6 | 29.0 | 93.3 |

TABLE II
THE POSSIBILITY THAT EACH SEARCH DIRECTION WINS (%)

| | none | linear | geometric | algebraic |
|---|---|---|---|---|
| Iter. | 0 | 2.8 | 46.8 | 58.0 |
| Times | 0 | 16.2 | 82.5 | 89.8 |

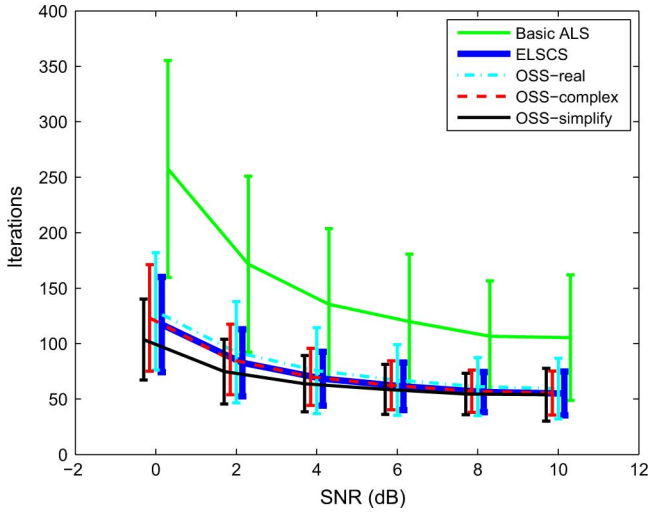



Fig. 2. Performance of mean number of iterations.

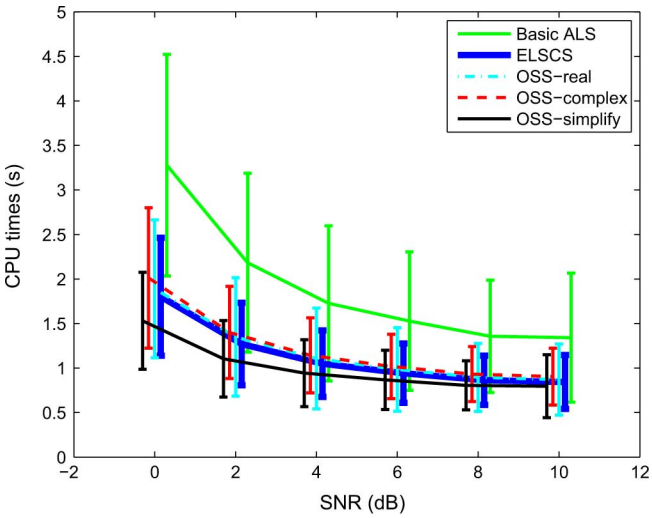



Fig. 3. Performance of mean CPU times.

From Table I, although the OSS-complex do not work well, the OSS-simplify that is a heuristic implementation of OSS-complex outperforms all the other line search strategies.

*How much does OSS-simplify faster than the others?* Figs. 2 and 3 illustrate the performance of mean value as well as standard deviation of iterations and CPU times, respectively. Form the two figures, we get the following conclusions.

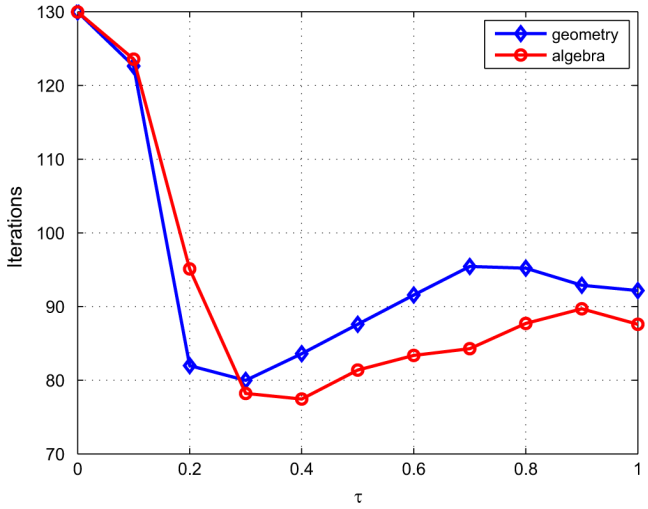



Fig. 4. Iterations versus values of $\tau_1$ and $\tau_2$.

- Basic ALS that takes a fixed unit step size converges slowly. Hence an adaptive step size strategy is necessary.
- Compared with OSS-real, the complex step size strategies could improve ALS by saving about five iterations. Hence, the imaginary part of the step size factor is valuable. Consequently, a real step size is not recommended in algorithms for complex-valued tensor decompositions.
- When we look into the three complex step size strategies, OSS-complex takes exactly the optimal complex step size factor, ELSCS may take a local optimal one and OSS-simplify takes an approximate one. Compared with the OSS-complex, ELSCS saves about two ALS iterations, and OSS-simplify saves about seven ALS iterations. This phenomena confirms that in numerical applications, inexact line search methods perform better than the exact one. On the other hand, the comparative result of the CPU times for the three complex step size schemes are almost the same as that of iterations. From this observation we argue that the costs for computing a complex step size factor by these schemes are negligible when compared with the costs of ALS iterations (forming the search direction).
- ELSCS performs quite well. The main reason is that the local minimum of the objective function is usually unique, which means that ELSCS always obtains the global optimal complex step size factor.

### B. The Second Result: Directions

First, we assign two values to the parameters $\tau_1$ and $\tau_2$ involved in the geometric and algebraic search directions, respectively. Since no ideal of solving this problem from a theoretical point of view currently, we run 500 Monte Carlo tests with $\mathrm{SNR} = 0$ dB. Fig. 4 shows the mean iterations versus some values of $\tau_1$ and $\tau_2$. Obviously, $\tau_1 = 0.3$ and $\tau_2 = 0.4$ are the best choices for these two directions, respectively. Therefore, we keep these values for $\tau_1$ and $\tau_2$ in the following simulations.

Next, we test four kinds of search directions in an ALS framework to observe their performances, where all the step size scheme are the fastest OSS-simplify.

- "None": basic ALS, no line search.

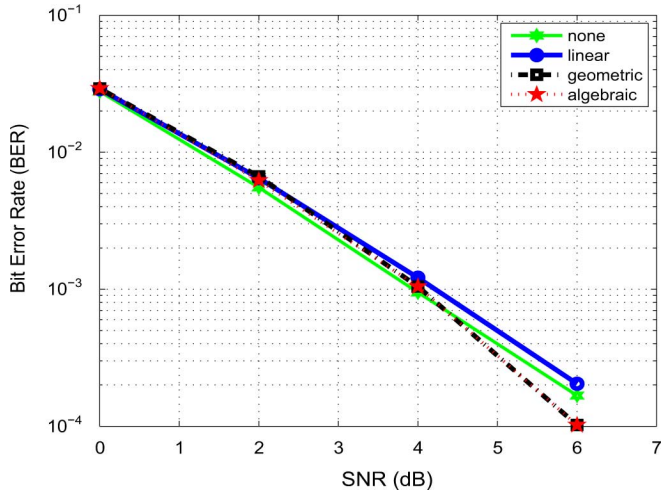



Fig. 5. BER versus SNR.

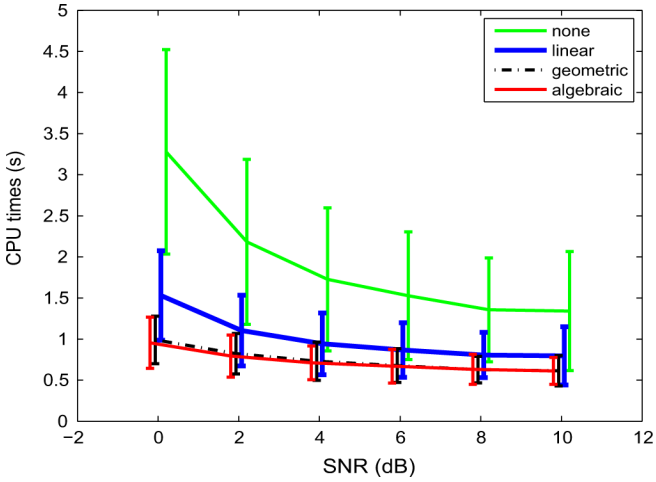



Fig. 7. Performance of mean CPU times.

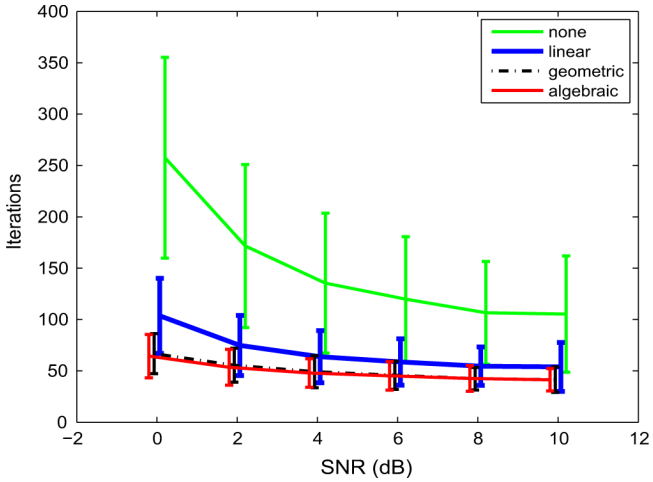



Fig. 6. Performance of mean number of iterations.

- "Linear": linear directions (3)–(5) that is often used.
- "Geometric": geometric directions (6)–(8).
- "Algebraic": algebraic directions (10)–(12).

We now give some results based on statistics.

*Which search direction gives the most accurate signal estimation?* The average bit error rate (BER) over all users is illustrated by Fig. 5. Compared to the often used linear directions, the new geometric and algebraic directions have more chance to point to the accurate decomposition of tensors.

*Which search direction is the fastest one?* See Table II. Compared with the low percentage that the linear direction wins, the geometric and algebraic directions work much better. This result indicates the importance of search directions. Additionally, the algebraic directions works slightly better than the geometric ones.

*How much does the new directions faster than the others?* Figs. 6 and 7 show the performance of mean value as well as standard deviation of iterations and CPU times, respectively. From them, we get some results as follows.

- Compared with basic ALS, the algorithm using linear directions saves about 54% ALS iterations and 47% CPU times averagely. The result of this experiment is consistent with the fact that a line search method can accelerate ALS.
- The algorithms using the geometric and algebraic directions save about 67% ALS iterations and 62% CPU times. This is better than the results of linear directions. When higher-order information is exploited in search directions, the iterations and CPU times of ALS are reduced significantly, although the memory is slightly enlarged.
  Since the vertical segments denoting standard deviation of the new geometric and algebraic directions are all lower than the average line of the often used linear directions, we say that the new directions are superior to the linear direction in a large possibility. This is concordant with the result in Table II.
- The algebraic directions that formed by the convex combination of the linear and quadratic extrapolating directions are more suitable for noise corrupted data than the geometric ones. We believe that the Lagrangian extrapolating technique has a large potential for extracting higher-order information and forming better search directions.

### C. The Third Result: Nearly Collinear Case

When the signals of the users are nearly collinear, which may happen in practice, the convergent rate of ALS is very slow. In the experiment of this subsection, the first user's signal is random QPSK symbols, and the other users' signals differ from it in just one bit. The condition number of factor $\mathbf{A}$ is 10, and tensor $\mathcal{Y}$ is noise free.

The typical curve of the cost function values versus iterations is shown by Fig. 8. Obviously, all the algorithms converge to the right decomposition. The new ALS algorithms with extrapolating search directions and optimal step size strategies outperform the existing algorithms. Since some second-order information is used, we observe a faster local convergence rate when iterations approach the solution.

## VI. Conclusion

In this paper, we applied the extrapolating technique to construct two new search directions and proposed a novel tech-

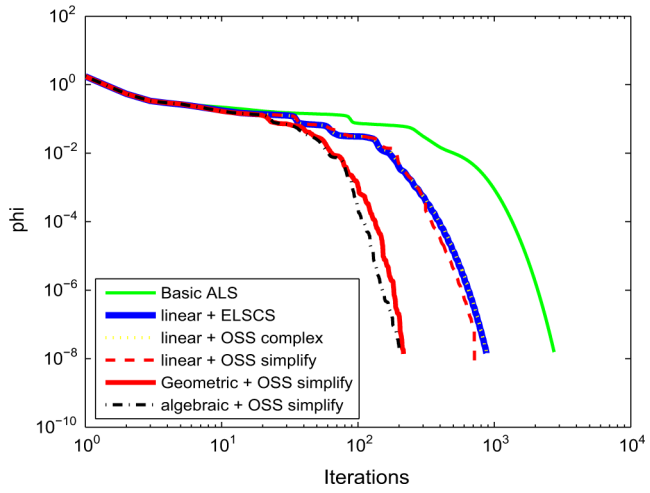



Fig. 8. Cost function values versus iterations.

nique to compute the optimal complex step size factor. Simulation results showed that these techniques have successfully and remarkably improved the rate of convergence of ALS. All the techniques could be used to compute the decomposition of complex-valued tensors that follow CP and BCM.

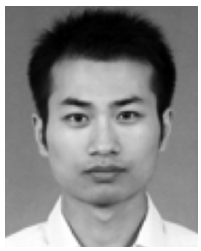
For the extraction of the second-order information, three iterates were used to form the search directions, so the new method can be viewed as a multistep method. How to explore the potentiality of multistep method and form effective search directions? This is one of our further works.

## ACKNOWLEDGMENT

The authors thank the reviewers for their comments and suggestions, which help us improve the presentation of the paper essentially.

## REFERENCES

[1] E. Acar, D. M. Dunlavy, and T. G. Kolda, "A scalable optimization approach for fitting canonical tensor decompositions," *J. Chemom.*, vol. 25, no. 2, pp. 67–86, 2011.
[2] E. Acar, T. G. Kolda, and D. M. Dunlavy, "An Optimization Approach for Filtting Canonical Tensor Decompositions," Sandia Report, 2009.
[3] E. Acar and B. Yener, "Unsupervised multiway data analysis: A literature survey," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 1, pp. 6–20, 2009.
[4] J. D. Carroll and J. Chang, "Analysis of individual differences in multidimensional scaling via an N-way generalization of 'Eckart-Young' decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–391, 1970.
[5] P. Comon, J. McWhirter and I. Proudler, Eds., "Tensor decompositions," in *Proc. Math. Signal Process. V*, Oxford, U.K., 2002, pp. 1–24.
[6] P. Comon, X. Luciani, and A. L. F. de Almeida, "Tensor decompositions, alternating least squares and other tales," *J. Chemom.*, vol. 23, pp. 393–405, 2009.
[7] D. Cox, J. Little, and D. O'Shea, *Using Algebraic Geometry*. New York: Springer-Verlag, 1998.
[8] A. L. F. de Almeida, G. Favier, and J. C. M. Mota, "Generalized PARAFAC model for multidimensional wireless communications with application to blind multiuser equalization," in *Proc. 39th ASILOMAR Conf. Signal Syst. Comput.*, Pacific Grove, CA, 2005.
[9] A. L. F. de Almeida, G. Favier, and J. C. M. Mota, "PARAFAC-based unified tensor modeling for wireless communication systems with application to blind multiuser equalization," *Signal Process.*, vol. 87, pp. 337–351, 2007.
[10] A. L. F. de Almeida, G. Favier, and J. C. M. Mota, "Constrained tensor modeling approach to blind multiple-antenna CDMA schemes," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2417–2428, 2008.
[11] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, "An introduction to independent component analysis," *J. Chemom.*, vol. 14, pp. 123–149, 2000.
[12] L. D. Lathauwer, "Decompositions of a higher-order tensor in block terms. Part I: Lemmas for partitioned matrices," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1022–1032, 2008.
[13] L. D. Lathauwer, "Decompositions of a higher-order tensor in block terms. Part II: Definitions and uniqueness," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1033–1066, 2008.
[14] L. D. Lathauwer and D. Nion, "Decompositions of a higher-order tensor in block terms. Part III: Alternating least squares algorithms," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1067–1083, 2008.
[15] S. Goedecker, "Remark on algorithms to find roots of polynomials," *SIAM J. Sci. Comput.*, vol. 15, no. 5, pp. 1059–1063, 1994.
[16] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The Johns Hopkins Univ. Press, 1996.
[17] R. A. Harshman, "Foundations of the PARAFAC procedure: Model and conditions for an 'explanatory' multi-mode factor analysis," in *Proc. UCLA Working Papers in Phonet.*, 1970, vol. 16, pp. 1–84.
[18] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *J. Math. Phys.*, vol. 6, pp. 164–189, 1927.
[19] F. L. Hitchcock, "Multiple invariants and generalized rank of a p-way matrix or tensor," *J. Math. Phys.*, vol. 7, pp. 39–79, 1927.
[20] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.
[21] MathWorks [Online]. Available: http://www.mathworks.com/help/techdoc/ref/roots.html
[22] D. Nion and L. D. Lathauwer, "A block factor analysis based receiver for blind multi-user access in wireless communications," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Toulouse, France, May 2006, pp. 825–828.
[23] D. Nion and L. D. Lathauwer, "Line search computation of the block factor model for blind multi-user access in wireless communications," in *Proc. IEEE Workshop on Signal Process. Adv. Wireless Commun. (SPAWC)*, Cannes, France, Jul. 2–5, 2006.
[24] D. Nion and L. D. Lathauwer, "An enhanced line search scheme for complex-valued tensor decompositions. Application in DS-CDMA," *Signal Process.*, vol. 88, no. 3, pp. 749–755, 2008.
[25] D. Nion and L. D. Lathauwer, "A block component model-based blind DS-CDMA receiver," *IEEE Trans. Signal Process.*, vol. 56, no. 11, pp. 5567–5579, 2008.
[26] L. Qi, W. Sun, and Y. Wang, "Numerical multilinear algebra and its applications," *Front. Math. China*, vol. 2, no. 4, pp. 501–526, 2007.
[27] N. D. Sidiropoulos, G. B. Giannakis, and R. Bro, "Blind PARAFAC receivers for DS-CDMA systems," *IEEE Trans. Signal Process.*, vol. 48, pp. 810–823, 2000.
[28] A. Smilde, R. Bro, and P. Geladi, *Multi-Way Analysis, Applications in the Chemical Sciences*. Chichester, U.K.: Wiley, 2004.
[29] M. Rajih and P. Comon, "Enhanced line search a novel method to accelerate PARAFAC," in *Proc. Eusipco'05*, Antalya, Turkey, 2005.
[30] M. Rajih, P. Comon, and R. A. Harshman, "Enhanced line search: A novel method to accelerate PARAFAC," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1148–1171, 2008.



**Yannan Chen** was born in China in 1981. He received the master degree from Nanjing Normal University, China, in 2007.

He is currently pursuing the Ph.D. degree at the School of Mathematical Sciences, Nanjing Normal University. He is a Lecturer in Nanjing Forestry University. His research interests include multilinear algebra, tensor computation, nonlinear numerical optimization, and applications.

**Deren Han** was born in China in 1974. He received the B.S. and Ph.D. degrees at Nanjing University, China, in 1997 and 2002, respectively.

He is now an Associate Professor at School of Mathematical Sciences, Nanjing Normal University. His research interests include numerical optimization, variational inequalities, tensor decomposition, transportation research, and signal and image processing.

**Liqun Qi** received the B.S. degree at Tsinghua University, Beijing, China, in 1968 and the M.S. and Ph.D. degrees from the University of Wisconsin-Madison in 1981 and 1984, respectively.

He has taught at Tsinghua University, the University of Wisconsin-Madison, the University of New South Wales, and the City University of Hong Kong. He is now the Chair Professor of applied mathematics and Head of the Department of Applied Mathematics, The Hong Kong Polytechnic University. He has published more than 150 research papers in international journals. His research interests include multilinear algebra, tensor computation, nonlinear numerical optimization and applications.