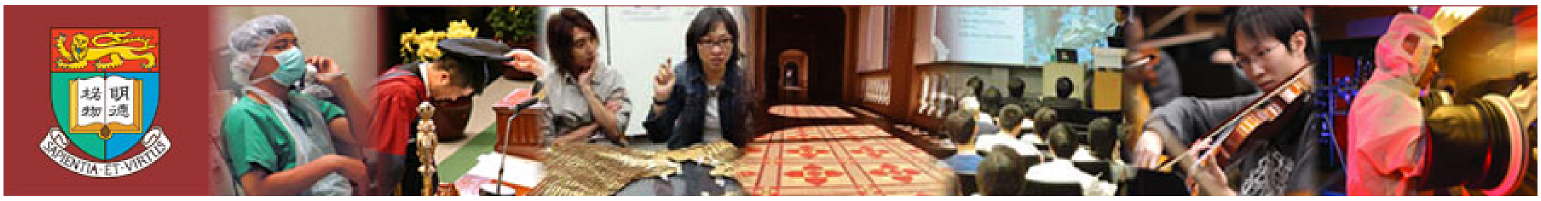

SYMMETRIC TENSOR DECOMPOSITION BY AN ITERATIVE EIGENDECOMPOSITION ALGORITHM

KIM BATSELIER AND NGAI WONG *

Abstract. We present an iterative algorithm, called the symmetric tensor eigen-rank-one iterative decomposition (STEROID), for decomposing a symmetric tensor into a real linear combination of symmetric rank-1 unit-norm outer factors using only eigendecompositions and least-squares fitting. Originally designed for a symmetric tensor with an order being a power of two, STEROID is shown to be applicable to any order through an innovative tensor embedding technique. Numerical examples demonstrate the high efficiency and accuracy of the proposed scheme even for large scale problems. Furthermore, we show how STEROID readily solves a problem in nonlinear block-structured system identification and nonlinear state-space identification.

Key words. Symmetric tensor, decomposition, rank-1, eigendecomposition, least-squares

AMS subject classifications. 15A69,15A18,15A23

1. Introduction.

Symmetric tensors arise naturally in various engineering problems. They are especially important in the problem of blind identification of underdetermined mixtures [6, 9, 7]. Applications of this problem are found in areas such as speech, mobile communications, biomedical engineering and chemometrics.

The main contribution of this paper is an algorithm, called the **S**ymmetric **T**ensor **E**igen-**R**ank-**O**ne **I**terative **D**ecomposition (STEROID), that decomposes a real symmetric tensor $\mathcal{A}$ into a linear combination of symmetric unit-norm rank-1 tensors

$$\begin{aligned} \mathcal{A} &= l_1\, x_1 \circ x_1 \circ \ldots \circ x_1 + \ldots + l_R\, x_R \circ x_R \circ \ldots \circ x_R, \\ &= l_1\, x_1^d + \ldots + l_R\, x_R^d, \end{aligned} \tag{1.1}$$

with $l_1, \ldots, l_R \in \mathbb{R}$ and $x_1, \ldots, x_R \in \mathbb{R}^n$. The reality of the scalar coefficients $l_1, \ldots, l_R$ is of particular importance in the nonlinear system identification algorithm presented in Section 5. The $\circ$ operation refers to the outer product, which we define in Section 1.1. The notation $x_i^d\,(i = 1, \ldots, R)$ denotes the d-times repeated outer product. In contrast to other iterative methods, STEROID does not require any initial guess and, as shown in Section 4, can handle large symmetric tensors. The minimal $R = R_{\min}$ that satisfies (1.1) is called the symmetric rank of $\mathcal{A}$. More information on the rank of tensors can be found in [14, 18] and specifically for symmetric tensors in [5]. The main idea of the algorithm is to first compute a set of vectors $x_1, \ldots, x_R\,(R \geq R_{\min})$ through repeated eigendecompositions of symmetric matrices. The coefficients $l_1, \ldots, l_R$ are then found from solving a least-squares problem. STEROID was originally developed for symmetric tensors with an order that is a power of 2. It is however perfectly possible to extend the applicability of the STEROID algorithm to symmetric tensors of arbitrary order by means of an embedding procedure, which we explain in Section 2.2.

In [2] an algorithm is described that decomposes a symmetric tensor over $\mathbb{C}$ using methods from algebraic geometry. This involves computing the eigenvalues of commuting matrices and as a consequence, the l coefficients obtained from this method are generally complex numbers. Most attention in the literature is spent in solving the low-rank (typically rank-1) approximation problem. This problem can be formulated as follows.

*Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong

Problem 1. *Given a dth-order symmetric tensor $\mathcal{A} \in \mathbb{R}^{n\times\cdots\times n}$ and a multilinear rank r, find an orthogonal $n \times r$ matrix U and a core tensor $\mathcal{S} \in \mathbb{R}^{r\times\cdots\times r}$ that minimizes the Frobenius norm*

$$||\mathcal{A} - \mathcal{S} \times_1 U \times_2 U \times_3 \cdots \times_d U||_F,$$

where $\times_i$ denotes the ith-mode product.

Note that the Tucker form $\mathcal{S} \times_1 U \times_2 U \times_3 \cdots \times_d U$ is intrinsically different from (1.1), since it will also contain rank-1 terms that are not symmetric. This implies that it is not very meaningful to compare the number of rank-1 terms from the Tucker form with the number of terms computed by STEROID. Algorithms designed specifically for finding rank-1 solutions to Problem 1 are the symmetric higher-order power method (S-HOPM) [13, 23] and the shifted version of S-HOPM (SS-HOPM) [16, 17]. General low-rank algorithms are the Quasi-Newton algorithm [25], the Jacobi algorithm [12] and the monotonically convergent algorithm described in [22].

Another common decomposition is the canonical tensor decomposition/parallel factors (CANDECOMP/PARAFAC) [3, 15]. This decomposition expresses a tensor as the sum of a finite number of rank-1 tensors. The tensor rank can then be defined as the minimum number of required rank-1 terms. Running a CANDECOMP algorithm such as Alternating Least Squares (ALS) on a symmetric tensor does not guarantee the symmetry of the rank-1 tensors. Other iterative methods [27], using nonlinear optimization methods, are able to guarantee the symmetry of the rank-1 terms. These methods however require the need for an initial guess and the number of computed terms also needs to be decided by the user beforehand. This is the main motivation for the development of the STEROID algorithm. STEROID is an adaptation for symmetric tensors of our earlier developed Tensor Train rank-1 SVD (TTr1SVD) algorithm [1], which in turn was inspired by Tensor Trains [19], and was an independent derivation of PARATREE [24]. In contrast to the iterative methods mentioned above, the STEROID algorithm does not require an initial guess and the total number of terms in the decomposition follows readily from the execution of the algorithm.

The outline of this paper is as follows. First, we define some basic notations in Section 1.1. In Section 2 we fully describe our algorithm by means of a running example, together with the required embedding procedure. Two methods for the reduction of the size of the least-squares problem in the STEROID algorithm are discussed in Section 3. One method exploits the symmetry of the tensor, while the other method exploits the structure of the matrix in the least-squares problem. The algorithm is applied to several examples in Section 4 and compared with the Jacobi algorithm [12], Regalia's iterative method described in [22] and the CANDECOMP-algorithm from the Tensorlab toolbox [27]. In Section 5 we show how STEROID readily solves a problem in nonlinear block-structured system identification [10] and nonlinear state-space identification [20]. In this setting, it is often desired to recover the internal structure of an identified static nonlinear mapping [26, 29, 31]. More specifically, it will be shown how STEROID can decouple a set of multivariate polynomials $f_1, \ldots, f_l$ into a collection of univariate polynomials $g_1, \ldots, g_n$, through both an affine and linear transformation.

1.1. Tensor Notations and Basics.

We will adopt the following notational conventions. A dth-order or d-way tensor, assumed real throughout this article, is a multi-dimensional array $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$ with elements $\mathcal{A}_{i_1 i_2 \ldots i_d}$ that can be seen as an extension of the matrix format to its general dth-order counterpart. Although the wordings 'order' and 'dimension' seem to be interchangeable in the tensor community, we prefer to call the number of indices i_k $(k = 1, \ldots, d)$ the order of the tensor, while the maximal value n_k $(k = 1, \ldots, d)$ associated with each index the dimension. A cubical tensor is a tensor for which $n_1 = n_2 = \ldots = n_d = n$. The inner product between two tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ is defined as

$$\langle \mathcal{A}, \mathcal{B} \rangle \;=\; \sum_{i_1, i_2, \ldots, i_d} \mathcal{A}_{i_1 i_2 \ldots i_d} \, \mathcal{B}_{i_1 i_2 \ldots i_d}.$$

The norm of a tensor is often taken to be the Frobenius norm $||\mathcal{A}||_F = \langle \mathcal{A}, \mathcal{A} \rangle^{1/2}$. A 3rd-order rank-1 tensor $\mathcal{A}$ can always be written as the outer product [15]

$$\mathcal{A} = \lambda \, a \circ b \circ c \quad \text{with components} \quad \mathcal{A}_{i_1 i_2 i_3} \;=\; \lambda \, a_{i_1} \, b_{i_2} \, c_{i_3}$$

with $\lambda \in \mathbb{R}$ whereas a, b and c are vectors of arbitrary lengths as demonstrated in Figure 1.1. Similarly, any d-way tensor of rank 1 can be written as an outer product of d vectors.

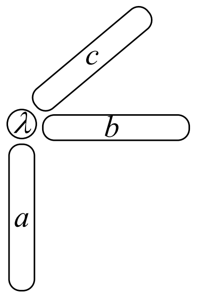

FIG. 1.1. *The outer product of 3 vectors a, b, c of arbitrary lengths forming a rank-1 tensor.*

We will only consider symmetric tensors in this article. A tensor $\mathcal{A}$ is symmetric if $\mathcal{A}_{i_1 \ldots i_d} = \mathcal{A}_{\pi(i_1 \ldots i_d)}$, where $\pi(i_1 \ldots i_d)$ is any permutation of the indices $i_1 \ldots i_d$. A rank-1 symmetric d-way tensor $\mathcal{A}$ is then given by the d-times repeated outer product $\mathcal{A} = \lambda \, a \circ a \circ \ldots \circ a \triangleq \lambda a^d$. The vectorization of a tensor $\mathcal{A}$, denoted $\text{vec}\,(\mathcal{A}) \in \mathbb{R}^{n_1 \cdots n_d}$, is the vector obtained from taking all indices together into one mode. This implies that for a symmetrical rank-1 tensor $\mathcal{A}$, its vectorization is

$$\text{vec}\,(\mathcal{A}) \;=\; \lambda \, a \otimes a \otimes \ldots \otimes a \;=\; \lambda \, a^{ⓓ},$$

where we have introduced the shorthand notation $a^{ⓓ}$ for the d-times repeated Kronecker product $\otimes$. Using the vectorization operation we can write (1.1) as

$$\text{vec}\,(\mathcal{A}) \;=\; \lambda_1 \, {x_1}^{ⓓ} + \ldots + \lambda_R \, {x_R}^{ⓓ}, \tag{1.2}$$

or equivalently as

$$\text{vec}\,(\mathcal{A}) \;=\; X \, l,$$

where X is the matrix that is formed by the concatenation of all the ${x_1}^{ⓓ}, \ldots, {x_R}^{ⓓ}$ vectors and $l \in \mathbb{R}^R$. In other words, the vectorization of a symmetric tensor $\mathcal{A}$ lives in the range of X, which is spanned by vectors ${x_i}^{ⓓ}$. This requirement puts the known constraint [5] on the rank of X

$$\text{rank}(X) \leq R_{\max} \triangleq \binom{d+n-1}{n-1},$$

for which we give a short proof in Lemma 2.1. The inverse vectorization operation unvec reshapes a vectorized tensor back into a tensor $\mathcal{A} = \text{unvec}\,(\text{vec}\,(\mathcal{A}))$.

2. Symmetric Tensor Eigen-Rank-One Iterative Decomposition.

2.1. Main Algorithm. We now demonstrate the STEROID algorithm that decomposes a symmetric tensor into a real finite sum of symmetric rank-one outer factors by means of a 4-way tensor. Later on, we then show that STEROID is applicable to any tensor order via an innovative tensor embedding technique. The first step in the STEROID algorithm is to reshape the (4-way) symmetric $\mathcal{A} \in \mathbb{R}^{n\times n\times n\times n}$ into a 2-way symmetric matrix $A^{(n^2\times n^2)}$, where the bracketed superscript indicates the dimensions. The symmetry of the reshaped A follows trivially from the symmetry of $\mathcal{A}$. Now the eigendecomposition of A can be computed, which allows us to write

$$A \;=\; \sum_{i=1}^{n^2} \lambda_i\, v_i \circ v_i \;=\; \sum_{i=1}^{n^2} \lambda_i\, v_i\, v_i^T. \tag{2.1}$$

The symmetry of A implies that the eigenvalues λ_i are real and the eigenvectors v_i will be orthonormal. Both eigenvalues and eigenvectors can be computed by for example the symmetric QR algorithm or the divide-and-conquer method [11]. Each of these eigenvectors v_i can now be reshaped into another 2-way symmetric matrix $\bar{v}_i^{(n\times n)}$. It is readily shown that the $\bar{v}_i$ vectors are also symmetric. Specifically, the symmetry of A implies that we can write

$$A\,P \;=\; A, \tag{2.2}$$

where P is any permutation matrix that permutes the indices $i_{\frac{d}{2}+1}\cdots i_d$. Using the eigendecomposition of A, we can rewrite (2.2) as

$$(\lambda_1\, v_1 v_1^T + \ldots + \lambda_{n^2}\, v_{n^2} v_{n^2}^T)\, P \;=\; (\lambda_1\, v_1 v_1^T + \ldots + \lambda_{n^2}\, v_{n^2} v_{n^2}^T). \tag{2.3}$$

Left-multiplying (2.3) with the eigenvector v_i^T $(i = 1, \ldots, n^2)$ of the ith term, assuming $\lambda_i \neq 0$, we obtain

$$\begin{aligned} v_i^T\,(\lambda_1\, v_1 v_1^T + \ldots + \lambda_{n^2}\, v_{n^2} v_{n^2}^T)\, P &= v_i^T\,(\lambda_1\, v_1 v_1^T + \ldots + \lambda_{n^2}\, v_{n^2} v_{n^2}^T),\\ \Leftrightarrow \lambda_i v_i^T P &= \lambda_i v_i^T,\\ \Leftrightarrow v_i^T P &= v_i^T, \end{aligned}$$

which implies that any of the eigenvectors $v_i\,(i = 1, \ldots, n^2)$ and consequently their reshaped $\bar{v}_i$ inhibit the same symmetry as A. The eigendecomposition of each of the symmetric $\bar{v}_i$'s can now also be computed. For example, $\bar{v}_1$ can then be written as

$$\bar{v}_1 \;=\; \sum_{i=1}^{n} \lambda_{1i}\, v_{1i} \circ v_{1i},$$

where the v_{1i}'s are again orthogonal due to the symmetry of $\bar{v}_1$. The whole procedure of repeated eigendecompositions of the reshaped eigenvectors for a $d = 4, n = 2$ example is depicted in Figure 2.1.

Referring to Figure 2.1, we now take the kth term of (2.1) and vectorize it to obtain

$$\text{vec}\,(\lambda_k v_k v_k^T) = \lambda_k v_k{}^{⓶}.$$

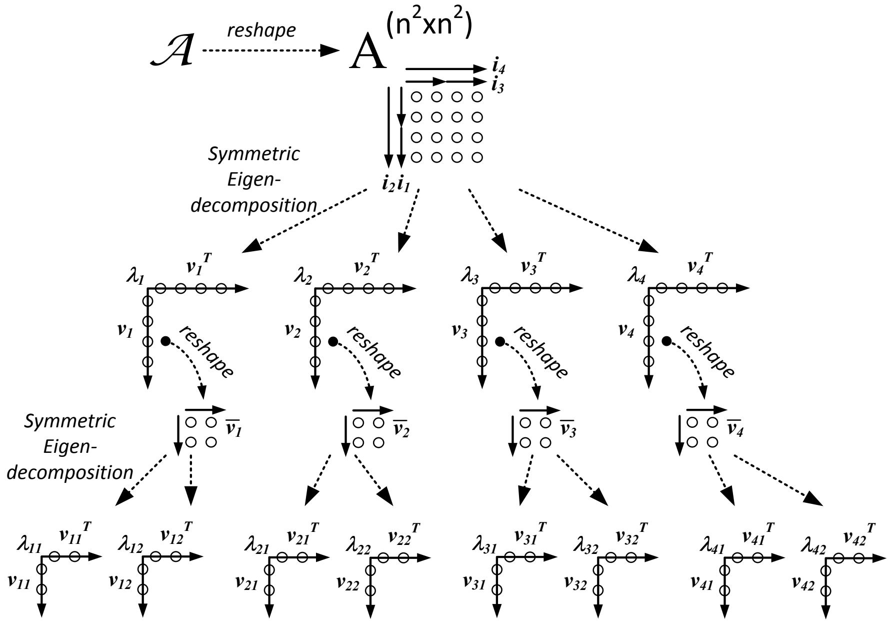
reshape
$\mathcal{A}$
$A^{(n^2xn^2)}$
i_4
i_3
$i_2 i_1$
Symmetric Eigen-decomposition
λ_1 v_1^T v_1
λ_2 v_2^T v_2
λ_3 v_3^T v_3
λ_4 v_4^T v_4
reshape
$\bar{v}_1$ $\bar{v}_2$ $\bar{v}_3$ $\bar{v}_4$
Symmetric Eigen-decomposition
λ_{11} v_{11}^T v_{11}
λ_{12} v_{12}^T v_{12}
λ_{21} v_{21}^T v_{21}
λ_{22} v_{22}^T v_{22}
λ_{31} v_{31}^T v_{31}
λ_{32} v_{32}^T v_{32}
λ_{41} v_{41}^T v_{41}
λ_{42} v_{42}^T v_{42}

FIG. 2.1. *Successive decompositions of the reshaped $A^{(n^2\times n^2)}$ for the specific case of $n = 2$. Note that $\bar{v}_i$'s are always symmetric due to the 4-way symmetry.*

Substitution of v_k by its eigendecomposition allows us to write

$$\begin{aligned} \lambda_k {v_k}^{②} =& \lambda_k \left(\lambda_{k1} {v_{k1}}^{②} + \lambda_{k2} {v_{k2}}^{②}\right) \otimes \left(\lambda_{k1} {v_{k1}}^{②} + \lambda_{k2} {v_{k2}}^{②}\right) \\ =& \underbrace{\lambda_k \lambda_{k1}^2 {v_{k1}}^{④} + \lambda_k \lambda_{k2}^2 {v_{k2}}^{④}}_{h_k} + \underbrace{\lambda_k \lambda_{k1} \lambda_{k2} \left({v_{k1}}^{②} \otimes {v_{k2}}^{②} + {v_{k2}}^{②} \otimes {v_{k1}}^{②}\right)}_{t_k} \end{aligned} \tag{2.4}$$

where h_k denotes the "head" part containing the *pure powers* ${v_{k1}}^{④}, {v_{k2}}^{④}$ of v_{k1} and v_{k2}, respectively, whereas t_k denotes the "tail" holding the sum of cross terms. Defining the head tensor $\mathcal{H} = \text{unvec}\left(\sum_k h_k\right)$ and the tail tensor $\mathcal{T} = \text{unvec}\left(\sum_k t_k\right)$, it can be deduced that $\mathcal{T} = \mathcal{A} - \mathcal{H}$ must also be 4-way symmetric since $\mathcal{A}$ and $\mathcal{H}$ are so. As we will explain further on, the symmetry of the tail tensor $\mathcal{T}$ is crucial, since it is possible that repeated eigendecompositions of the reshapings of $\mathcal{T}$ are also necessary in order to compute additional pure power vectors of the STEROID. The objective now is to write vec($\mathcal{A}$) as a linear combination

$$\text{vec}\,(\mathcal{A}) \;=\; l_1 {x_1}^{④} + \ldots + l_R {x_R}^{④}.$$

Good candidates for the ${x_i}^{④}$ vectors are the pure powers that span the head part in (2.4). No pure power vectors should be considered that come from an eigenvector with corresponding zero eigenvalue λ_k. Since each of the x_i's is an eigenvector of a symmetric matrix, it also follows that $||{x_i}^{④}||_F = 1$. Checking whether a decomposition as in (1.2) exists is done by computing the residual of the following least-squares problem

$$\hat{l} \;=\; \underset{l}{\text{argmin}}\; ||\, \text{vec}\,(\mathcal{A}) - X\, l \,||_F, \tag{2.5}$$

where X is the matrix obtained from the concatenation of all the obtained pure power vectors ${x_i}^{④}$. It is possible at this step to solve (2.5) with additional constraints. For example, if only positive l_i's are required then one could use a reflective Newton method as described in [4]. A sparse solution $\hat{l}$ with as few nonzero l_i's as possible can be computed using L1-regularization [28]. The particular repeated Kronecker product structure for each column of X results in the following upper bound on its rank.

LEMMA 2.1. *For the matrix X in the least-squares problem* (2.5) *we have that*

$$\textit{rank}\,(X) \leq R_{max} \triangleq \binom{d+n-1}{n-1}.$$

Proof. Each column of X corresponds with a vector ${x_i}^{ⓓ}$, with $x_i \in \mathbb{R}^n$. If we label the entries of x_i by $x_{i1}, \ldots, x_{in}$ then ${x_i}^{ⓓ}$ contains all monomials of degree d in n variables $x_{i1}, \ldots, x_{in}$. For example, if $d = 2$ and $n = 2$, then

$${x_i}^{②} = \begin{pmatrix} x_{i1} \\ x_{i2} \end{pmatrix} \otimes \begin{pmatrix} x_{i1} \\ x_{i2} \end{pmatrix} = \begin{pmatrix} x_{i1}^2 \\ x_{i1}x_{i2} \\ x_{i2}x_{i1} \\ x_{i2}^2 \end{pmatrix}$$

contains all homogeneous monomials in 2 variables of degree 2. Then for $i = 1, \ldots, 4$

$$X = \begin{pmatrix} x_{11}^2 & x_{21}^2 & x_{31}^2 & x_{41}^2 \\ x_{11}x_{12} & x_{21}x_{22} & x_{31}x_{32} & x_{41}x_{42} \\ x_{12}x_{11} & x_{22}x_{21} & x_{32}x_{31} & x_{42}x_{41} \\ x_{12}^2 & x_{22}^2 & x_{32}^2 & x_{42}^2 \end{pmatrix}$$

and has a rank of at most $\binom{2+2-1}{2-1} = 3$, since the second and the third row are identical. For the general case there are $\binom{d+n-1}{n-1}$ distinct homogeneous monomials of degree d in n variables and hence the rank is upper bounded by $R_{\max}$. ∎

Lemma 2.1 tells us that vec($\mathcal{A}$) of a symmetric tensor $\mathcal{A}$ lives in a $R_{\max}$-dimensional vector space. Therefore, instead of computing the STEROID, one could randomly generate $R_{\max}$ linearly independent vectors, construct their corresponding X matrix and decompose vec($\mathcal{A}$) along this basis. However, a random basis will most likely result in a decomposition with $R_{\max}$ nonzero terms, while the STEROID results in a more compact decomposition, as is illustrated in the following example.

EXAMPLE 2.1. *We construct the following symmetric tensor $\mathcal{A} = l_1\, a_1^4 + l_2\, a_2^4 + l_3\, a_3^4$ with l_1, l_2, l_3 random real numbers and a_1, a_2, a_3 real random 3×1 vectors. Since $d = 4, n = 3$, vec($\mathcal{A}$) lives in a $\binom{4+3-1}{3-1} = 15$-dimensional vector space $\mathcal{X}$. The STEROID of $\mathcal{A}$ consists of 9 nonzero terms, while the decomposition with respect to a random basis for $\mathcal{X}$ always generates 15 nonzero terms. Observe that the STEROID is not a canonical symmetric rank-1 decomposition, since the symmetric rank is by construction 3 while the STEROID consists of 9 terms.*

From Lemma 2.1 we learn two things. First, it is possible that not enough pure power vectors are computed to solve the least-squares problem (2.5). In this case the residual $||\,\text{vec}\,(\mathcal{A}) - X\,\hat{l}||_F$ will not be satisfactory and the same procedure of reshapings and eigendecompositions should be applied to the tail tensor $\mathcal{T}$. This will produce additional pure powers that can be used to extend X, upon which one can solve the least-squares problem (2.5) again. Further iterations on the resulting tail tensor can be applied until a satisfactory residual is obtained. The second thing we learn is that it is also possible that X becomes singular as soon as it has more than $\binom{d+n-1}{n-1}$

columns. In this case it is recommended to regularize the least-squares problem such that the obtained solution is not sensitive to perturbations of the tensor $\mathcal{A}$. This can be done by for example computing the minimum norm solution of (2.5). The whole STEROID algorithm for tensors with $d = 2^k$ ($k \in \mathbb{N}$) is summarized in pseudo-code in Algorithm 2.1. Matlab/Octave implementations can be freely downloaded from `https://github.com/kbatseli/STEROID`.

Algorithm 2.1. *STEROID algorithm*

```
Input: symmetric d-way tensor $\mathcal{A}$ with $d = 2^k, k \in \mathbb{N}$, tolerance $\tau$
Output: pure power vectors $x_i$, $\hat{l}$
  $\bar{A} \leftarrow$ reshape $\mathcal{A}$ into a $n^{d/2} \times n^{d/2}$ matrix
  $V_1, D_1 \leftarrow$ eig($\bar{A}$)
  for all eigenvectors $V_1(:,i)$ with $\lambda_i \neq 0$ do
    recursively reshape $V_1(:,i)$ and compute its eigendecomposition
  end for
  compute head tensor $\mathcal{H}$ from recursive eigendecompositions of $\mathcal{A}$
  $\mathcal{T} \leftarrow \mathcal{A}$
  collect all pure powers into $X$
  solve least-squares problem $\hat{l} = \underset{l}{\operatorname{argmin}} \, ||\operatorname{vec}(\mathcal{A}) - X\, l||_F$
  while $||\operatorname{vec}(\mathcal{A}) - X\, \hat{l}||_2 > \tau$ AND rank($X$) $< R_{max}$ do
    $\mathcal{T} \leftarrow \mathcal{T} - \mathcal{H}$
    add additional pure powers to $X$ by recursive eigendecompositions of $\mathcal{T}$
    compute new head tensor $\mathcal{H}$ from recursive eigendecompositions of $\mathcal{T}$
    extend $X$ with additional pure powers
    solve least-squares problem $\hat{l} = \underset{l}{\operatorname{argmin}} \, ||\operatorname{vec}(\mathcal{A}) - X\, l||_F$
  end while
```

Every matrix from which an eigendecomposition is computed in Algorithm 2.1 is symmetric. This implies that the computed eigenvalues are up to a sign equal to the singular values. Hence the same kind of tolerance as for the singular values can be used to determine whether any of the λ_i's are numerically zero [11]. Note that a user-defined tolerance τ is required to check whether additional iterations on the tail tensor $\mathcal{T}$ are required.

The computational complexity of the method is dominated by the very first eigendecomposition of the $n^{d/2} \times n^{d/2}$ matrix A and by solving the least-squares problem (2.5). The first eigendecomposition requires a tridiagonalization of A, which requires $8/3\, n^{3d/2}$ flops and dominates the cost. For the actual diagonalization, QR iterations or the divide-and-conquer method can be used. The matrix X in the least-squares problem also determines the computational cost. Its number of rows is n^d and we can assume its number of columns to be R_{max}. Solving the least-squares problem with the SVD of X then sets the maximal computational complexity to $2R_{\text{max}}^2(n^d - R_{\text{max}}/3)$. In Section 3 we discuss two ways in which the size of the least-squares problem can be significantly reduced by exploiting the symmetry of $\mathcal{A}$ and the structure of X.

It is clear that the STEROID algorithm presented in Algorithm 2.1 only works for symmetric tensors for which the order is a power of 2. Indeed, this is a necessary requirement such that the recursive reshapings in the algorithm always lead to a square symmetric matrix. Fortunately, by employing an embedding procedure one can compute the STEROID of a symmetric tensor $\mathcal{A}$ of any order. We now discuss this embedding procedure in the next section.

2.2. Tensor Embedding. The embedding procedure presented in this section allows us to compute the STEROID for a symmetric tensor of an arbitrary order. Algorithm 2.1 relies on the reshaping of the tensor into a square matrix and therefore the order of the tensor should be divisible by two. If the order d is odd, then one can embed the tensor into a symmetric tensor of order $d+1$, reshape it again into a square matrix and continue Algorithm 2.1. We now illustrate the embedding procedure with a 3-way symmetric tensor $\mathcal{A}$ of dimension 2 into a symmetric tensor $\mathcal{B}$ of order 4. Since $\mathcal{A}$ is symmetric, it only has 4 distinct entries, viz. $\mathcal{A}_{111}, \mathcal{A}_{211}(=\mathcal{A}_{121}=\mathcal{A}_{112}), \mathcal{A}_{221}(=\mathcal{A}_{212}=\mathcal{A}_{122}), \mathcal{A}_{222}$. The idea now is to consider $\mathcal{A}$ as the frontal "slice" of $\mathcal{B}$ in the following straightforward manner

$$\begin{aligned}
\mathcal{A}_{111} &\Rightarrow \mathcal{B}_{1111},\\
\mathcal{A}_{211} &\Rightarrow \mathcal{B}_{2111},\\
\mathcal{A}_{221} &\Rightarrow \mathcal{B}_{2211},\\
\mathcal{A}_{222} &\Rightarrow \mathcal{B}_{2221}.
\end{aligned}$$

In order to make sure that $\mathcal{B}$ is symmetric, one needs to enforce the following equalities

$$\begin{aligned}
&\mathcal{B}_{2111}=\mathcal{B}_{1211}=\mathcal{B}_{1121}=\mathcal{B}_{1112},\\
&\mathcal{B}_{2211}=\mathcal{B}_{2121}=\mathcal{B}_{2112}=\mathcal{B}_{1221}=\mathcal{B}_{1212}=\mathcal{B}_{1122},\\
&\mathcal{B}_{2221}=\mathcal{B}_{2212}=\mathcal{B}_{2122}=\mathcal{B}_{1222}.
\end{aligned}$$

All other entries of $\mathcal{B}$, in this example $\mathcal{B}_{2222}$, can be set to zero. We now have a symmetric $\mathcal{B}$ with $\mathcal{B}_{i_1 i_2 i_3 1}=\mathcal{A}_{i_1 i_2 i_3}$. The general embedding algorithm is presented in pseudo-code in Algorithm 2.2.

```
ALGORITHM 2.2. symmetric tensor embedding algorithm
Input: symmetric d-way cubical tensor $\mathcal{A}$ with d an odd number
Output: symmetric $d+1$-way cubical tensor $\mathcal{B}$ with $\mathcal{B}_{i_1\dots i_d 1}=\mathcal{A}_{i_1\dots i_d}$.
  initialize $\mathcal{B}$ with zeros
  for all nonzero $\mathcal{A}_{i_1\dots i_d}$ do
    for all permutations $\pi(i_1\dots i_d 1)$ do
      $\mathcal{B}_{\pi(i_1\dots i_d 1)} \leftarrow \mathcal{A}_{i_1\dots i_d}$
    end for
  end for
```

Using Algorithm 2.2, it now becomes possible to adjust the STEROID algorithm such that it works for a symmetric tensor of any order. Indeed, if the order d is odd, then application of Algorithm 2.2 guarantees that the new symmetric tensor can be reshaped into a square matrix. Similarly, the obtained eigenvectors can be embedded if necessary. The following example illustrates the STEROID algorithm with embedding.

EXAMPLE 2.2. *Suppose we have a symmetric tensor $\mathcal{A} \in \mathbb{R}^{n\times\cdots\times n}$ of order $d=5$. Since d is odd, it is not possible to reshape $\mathcal{A}$ into a square matrix. Application of Algorithm 2.2 returns a symmetric tensor $\mathcal{B}$ of order 6 such that $\mathcal{B}_{i_1\dots i_d 1}=\mathcal{A}_{i_1\dots i_d}$. The tensor $\mathcal{B}$ is then reshaped into a symmetric $n^3\times n^3$ matrix and its eigendecomposition is computed. Each obtained eigenvector corresponding with a nonzero eigenvalue can only be reshaped into a tensor of order 3, hence the embedding has to be applied again. One can then reshape each eigenvector into a symmetric $n^2\times n^2$ matrix* *and compute its eigendecomposition. Finally, the obtained eigenvectors corresponding with a nonzero eigenvalue are reshaped into a symmetric $n \times n$ matrix and the final eigendecomposition is computed, from which we obtain the pure power vectors.*

3. Reducing the size of the least-squares problem. The scalar factors l in the linear combination (1.1) are found as the solution of the least-squares problem

$$\hat{l} \;=\; \underset{l}{\operatorname{argmin}}\; ||\operatorname{vec}(\mathcal{A}) - X\, l||_F.$$

Since each column of X is a $x_i{}^{\textcircled{d}}$ vector, the total number of rows is n^d. The number of columns of X is determined by the total number of nonzero eigenvalues in the STEROID but is in practice much less than the number of rows. The feasibility of solving the least-squares problem will therefore be largely determined by the n^d number of rows of the X matrix, requiring large amounts of memory. In this section we discuss two effective methods to reduce the size of X and thus alleviate the memory requirement. The first method exploits the symmetry of $\mathcal{A}$ directly, the second method exploits the structure of the X matrix to efficiently compute $X^T X$.

3.1. Exploiting the symmetry of $\mathcal{A}$. The symmetry of $\mathcal{A}$ implies that many rows of X will be identical. In fact, only $\binom{d+n-1}{n-1}$ rows are unique due to Lemma 2.1. If S is the row selection matrix that selects the $\binom{d+n-1}{n-1}$ unique rows from X then (2.5) can be rewritten as the mathematically equivalent problem

$$\hat{l} \;=\; \underset{l}{\operatorname{argmin}}\; ||S\, \operatorname{vec}(\mathcal{A}) - S\, X\, l||_F. \tag{3.1}$$

The number of rows of X are then reduced with a factor of

$$\gamma \triangleq \frac{n^d}{\binom{d+n-1}{n-1}} \;=\; \frac{n^d\,(n-1)!}{(d+n-1)\cdots(d+1)},$$

which grows exponential in d. Figure 3.1 demonstrates the reduction factor γ as a function of n for different orders d. It can be seen that the reduction in number of rows first increases exponentially for small n and then quickly 'saturates' to an almost constant factor.

Although massive savings can be achieved by exploiting the symmetry of $\mathcal{A}$, $S\,X$ still has $\binom{d+n-1}{n-1} = d^{n-1}/(n-1)! + O(d^{n-2})$ rows, which grows exponential in n. This implies that even for moderate d, (2.5) will quickly become infeasible for increasing n. In the next section we discuss how the size of the least-squares problem can be further reduced by exploiting the particular structure of X. This will come, however, at the cost of a squared condition number when solving (2.5).

3.2. Exploiting the structure of X. The matrix X is typically very thin, with much more rows than columns. We assume in this section that X is of full column rank. One straightforward way then to reduce the size of the matrix is to left-multiply with X^T to obtain

$$X^T\, X\, l \;=\; X^T \operatorname{vec}(\mathcal{A}).$$

Since X is of full column rank, l will be unique and $X^T X$ will be symmetric and positive definite. This means that in addition to getting rid of the n^d rows, only half of the entries of $X^T X$ need to be stored. This comes at the cost of a squared

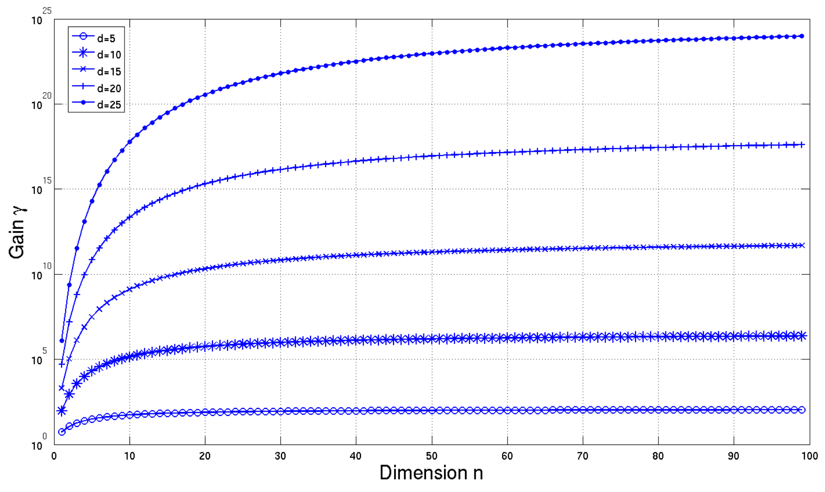

Fig. 3.1. *The reduction in number of rows of X as a function of n for different values of d.*

condition number $\kappa(X^T X) = \kappa(X)^2$, where $\kappa(X)$ denotes the condition number of the matrix X.

The matrix $X^T X$ can also be constructed without explicitly constructing X, what is to be avoided in the first place. Indeed, each column of X is a repeated Kronecker product of a pure power vector $x_i{}^{ⓓ}$. Each element of $X^T X$ is therefore an inner product $x_i{}^{ⓓ}\, x_j{}^{ⓓ}$, which can be rewritten as

$$x_i^T x_j \otimes x_i^T x_j \otimes \ldots \otimes x_i^T x_j \;=\; (x_i^T x_j)^d.$$

This allows us to construct $X^T X$ without the explicit construction of X as described in the following lemma.

Lemma 3.1. *If V is the matrix that consists of the pure power vectors obtained from the STEROID, then $X^T X$ is constructed from $(V^T V).^d$, where $.^d$ denotes the entrywise operation of raising to the power d.*

4. Numerical examples.

In this section we demonstrate the STEROID algorithm on different examples. All examples were run in Matlab [21] on a 64-bit desktop computer with 4 cores @ 3.30 GHZ and 16 GB of memory. The first example illustrates the different steps of the STEROID algorithm on a small symmetric tensor. The second example illustrates the case where more than one STEROID iteration is required to obtain the full decomposition. In Example 3, we demonstrate the impact of the two methods to reduce the size of the least-squares problem on the residuals and run times of the STEROID algorithm. Finally, we compare the STEROID algorithm with the output of the Jacobi algorithm [12], Regalia's iterative symmetric tensor approximation algorithm [22] and the CANDECOMP algorithm from the Tensorlab toolbox [27]. It is important to realize that in contrast to other methods, the STEROID algorithm does not require any initial guess and, as illustrated in Example 3, can handle large problems.

4.1. Example 1: STEROID algorithm illustration on a $2 \times 2 \times 2$ symmetric tensor.

We first demonstrate the STEROID on a simple symmetric 3rd-order tensor

$$\mathcal{A}_{111} = 24, \mathcal{A}_{211} = 18, \mathcal{A}_{221} = 12, \mathcal{A}_{222} = 6,$$

which we first need to extend to a 4th-order symmetrical cubical tensor $\mathcal{B}$ using Algorithm 2.2. The next step of the STEROID algorithm is to reshape $\mathcal{B}$ into the following 4×4 symmetric matrix

$$B = \begin{pmatrix} 24 & 18 & 18 & 12 \\ 18 & 12 & 12 & 6 \\ 18 & 12 & 12 & 6 \\ 12 & 6 & 6 & 0 \end{pmatrix}$$

and compute its eigendecomposition

$$B = V \begin{pmatrix} -5.3939 & 0 & 0 & 0 \\ 0 & -6.29 \times 10^{-15} & 0 & 0 \\ 0 & 0 & 2.64 \times 10^{-15} & 0 \\ 0 & 0 & 0 & 53.3939 \end{pmatrix} V^T.$$

Since B has 2 eigenvalues that are numerically zero, we only need to proceed with the eigenvectors $V(:,1)$ and $V(:,4)$, where we used MATLAB notation to denote the first and fourth columns of V. Reshaping both $V(:,1)$ and $V(:,4)$ into a symmetric 2×2 matrix and computing their eigendecomposition results in the following 4 pure power vectors

$$x_1 = \begin{pmatrix} -0.9939 \\ 0.1103 \end{pmatrix}, x_2 = \begin{pmatrix} -0.1103 \\ -0.9939 \end{pmatrix}, x_3 = \begin{pmatrix} -0.8396 \\ -0.5431 \end{pmatrix}, x_4 = \begin{pmatrix} 0.5431 \\ -0.8396 \end{pmatrix}.$$

Solving the least-squares problem

$$\hat{l} = \underset{l}{\operatorname{argmin}} \; || \operatorname{vec}(\mathcal{A}) - \begin{pmatrix} x_1{}^{③} & x_2{}^{③} & x_3{}^{③} & x_4{}^{③} \end{pmatrix} l ||_F,$$

results in

$$\hat{l} = \begin{pmatrix} 3.9934 \\ 0.6922 \\ -46.79 \\ 1.3916 \end{pmatrix},$$

with a residual of 1.8546×10^{-14}. Observe that the total number of terms in the computed decomposition equals the upper bound $\binom{3+2-1}{2-1} = 4$. The total run time to compute the decomposition was 7.5×10^{-4} seconds.

Another interesting example, that can be found in [5], is the symmetric tensor defined by

$$\mathcal{A}_{111} = -1, \mathcal{A}_{221} = 1.$$

The STEROID algorithm computes the following decomposition

$$\mathcal{A} = -2 \begin{pmatrix} 1 \\ 0 \end{pmatrix}^{③} - 1.4142 \begin{pmatrix} -0.7071 \\ 0.7071 \end{pmatrix}^{③} + 1.4142 \begin{pmatrix} 0.7071 \\ 0.7071 \end{pmatrix}^{③}$$

in 0.0012 seconds. This is the same decomposition as given in [5].

4.2. Example 2: Second STEROID iteration on tail tensor $\mathcal{T}$. Consider a random symmetric tensor $\mathcal{A} \in \mathbb{R}^{7\times 7\times 7\times 7}$ with integer entries between 24 and 100. The STEROID algorithm returns 196 pure power vectors. The rank of X is upper bounded by $\binom{4+7-1}{7-1} = 210$ and not surprisingly we have a residual of 70.2320, which indicates that additional pure power vectors are required. Running Algorithm 2.1 on the tail tensor $\mathcal{T}$ returns an additional 189 pure power vectors. Solving the least-squares problem (2.5) with all 385 pure power vectors results in a residual of 1.49×10^{-11} and 167 nonzero entries in l. The upper bound of 210 on the rank of X, together with the set of 385 pure power vectors implies that there is distinct non-uniqueness in the decomposition.

4.3. Example 3: Comparison between original STEROID algorithm, exploitation of symmetry and using X^TX. For this numerical experiment six random symmetric tensors with dimension $n = 5$ and orders $d = 3$ up to $d = 8$ were generated. Columns two and three of Table 4.1 list the total number of computed eigendecompositions and the total number of required embeddings for each value of d in the STEROID computation. Columns four and five list the total run times in seconds for computing all eigendecompositions and doing the tensor embeddings. The rightmost column lists the total time required to compute the pure power x_i vectors of the STEROID and is the sum of the entries in the fourth and fifth column. From Table 4.1 it can be seen that, unless the order is a power of two, the main contribution in the total time required to compute the pure power vectors comes from the embedding procedure. The 35 embeddings for $d = 6$ take about half the amount of time as the 31 embedding for $d = 5$. This is explained by the fact that $d = 5$ requires 1 embedding from order 5 to 6 and 30 embeddings from order 3 to 4, while for $d = 6$ there are 35 embeddings from order 3 to 4. Embedding a symmetric tensor from order 5 to order 6 is a much more time-consuming process than from order 3 to 4. The largest run time is observed for $d = 7$, which requires only 1 embedding from order 7 to order 8.

TABLE 4.1
Number of eigendecompositions and embeddings and their respective total run times.

d	total number of eigs	total number of embeddings	total time eigs [seconds]	total time embedding [seconds]	total time x_i vectors [seconds]
3	11	1	0.0008	0.4091	0.4099
4	16	0	0.0012	0	0.0012
5	181	31	0.0392	19.5757	19.6149
6	386	35	0.0449	9.5274	9.5714
7	606	1	1.0092	458.3181	459.3273
8	1184	0	0.9643	0	0.9643

Once all x_i vectors are computed, we solve the least-squares problem (2.5) in three different ways: original (no reduction of the X matrix), symmetry (exploiting the symmetry of $\mathcal{A}$), X^TX (computes X^TX using Lemma 3.1). Table 4.2 lists the residuals $||\mathcal{A} - X\hat{l}||_F$ and total run times in seconds for each of the three methods. The residuals for the X^TX method are only slightly worse than the other two methods due to the squared condition number. This implies that the condition numbers of

X were relatively small. The difference in run time is more pronounced for higher orders. Most apparent is the saving with a factor of 6942 in run time between the original and symmetry exploiting methods when $d = 8$. Exploiting the symmetry reduces the total number of rows of X from $5^8 = 390625$ down to $\binom{8+5-1}{5-1} = 495$. This reduces X to a 495×5550 matrix of rank 495. Using the $X^T X$ method when $d = 8$ reduces the run time with a factor 1174 but is clearly not as good as exploiting the symmetry. The reason for this difference lies in the fact that 5550 pure power vectors are computed and therefore $X^T X$ is a 5550×5550 matrix, compared to the 495×5550 matrix when symmetry is exploited.

TABLE 4.2
Residuals and run times for solving the least-squares problem (2.5).

d	original		symmetry		$X^T X$	
	$\|\|\mathcal{A} - X\hat{l}\|\|_F$	run time [seconds]	$\|\|\mathcal{A} - X\hat{l}\|\|_F$	run time [seconds]	$\|\|\mathcal{A} - X\hat{l}\|\|_F$	run time [seconds]
3	6.38e−15	0.0003	6.45e−15	0.0002	3.71e−14	0.0337
4	7.25e−14	0.0024	7.82e−15	0.0003	5.65e−13	0.0256
5	5.20e−13	0.3866	1.06e−14	0.0189	1.91e−12	0.0555
6	1.86e−12	8.5901	1.81e−14	0.0685	2.17e−11	0.2721
7	2.46e−11	186.5544	3.30e−14	0.6224	8.36e−12	1.9014
8	1.42e−10	13312.67	2.97e−14	1.9176	2.16e−10	11.338

4.4. Experiment 4: Comparison with other iterative methods. In this numerical experiment, we apply the Jacobi [12], Regalia's iterative symmetric tensor approximation algorithm [22] and the CANDECOMP-algorithm of the Tensorlab toolbox [27] to the symmetric tensors of Experiment 3. The Matlab implementation of the Jacobi algorithm was provided by Dr. Mariya Ishteva. This Jacobi method implementation only works for 3rd-order tensors. We implemented Regalia's iterative algorithm and confirmed its results with the numerical experiments described in [22]. The Tensorlab toolbox is freely available. For a given multilinear rank r, the Jacobi and Regalia's method return an orthogonal $n \times r$ matrix U and symmetric tensor core $\mathcal{S} \in \mathbb{R}^{n\times\cdots\times n}$ that minimize $||\mathcal{A} - \mathcal{S} \times_1 U \times_2 \cdots \times_d U||_F$. The maximal number of columns of U is therefore limited to n. This implies that for a fully dense 3rd-order core tensor $\mathcal{S}$ one will have $1, 4, 10, 20, 35$ respective number of terms in the decomposition for $r = 1, 2, 3, 4, 5$. Table 4.3 lists the total number of required iterations, the residual and total run time for the Jacobi method applied for all possible values of r. The initial orthogonal matrices to start the iterations were obtained from applying a QR orthogonalization on a random $n \times r$ matrix. The full Tucker decomposition is obtained for $r = 5$ and consists of 32 nonsymmetric and 3 symmetric terms. In contrast, the STEROID consists of 50 symmetric terms and is obtained more than 3 times faster when symmetry is exploited.

Regalia's iterative method is not limited to 3rd-order tensors and is therefore applied to all symmetric tensors of Experiment 3. Since we are interested in a full decomposition we set $r = 5$ and therefore obtain $\binom{3+5-1}{5-1}, \ldots, \binom{8+5-1}{5-1}$ terms for every respective decomposition. The initial orthogonal matrices to start the iterations were obtained from applying a QR orthogonalization on a random $n \times 5$ matrix. Table 4.4 lists the total number of required iterations, the residual and total run time for each symmetric tensor from Experiment 3. Iterations were stopped when the differ-

Table 4.3
Number of iterations, residuals and run times for the Jacobi method.

r	total number of iterations	$\lvert\lvert\mathcal{A} - \mathcal{S} \times_1 U \times_2 \cdots \times_d U\rvert\rvert_F$	total runtime [seconds]
1	38	5.28	1.38
2	118	4.63	2.02
3	111	3.47	2.46
4	55	1.83	1.16
5	1	7.22e−15	0.01

ence in consecutive orthogonal vectors over 2 iterations was smaller than 1e−10. An additional parameter γ to ensure monotonic convergence was set to 20. For $d = 8$, the algorithm failed to finish due to a lack of sufficient memory. Compared to the symmetry exploiting STEROID algorithm, the total run time of Regalia's iterative method is up to 35 times slower.

Table 4.4
Number of iterations, residuals and run times for Regalia's iterative method, $r = 5$.

d	total number of iterations	$\lvert\lvert\mathcal{A} - \mathcal{S} \times_1 U \times_2 \cdots \times_d U\rvert\rvert_F$	total runtime [seconds]
3	72	1.78e−15	0.2034
4	69	1.24e−14	1.2668
5	58	4.08e−14	8.4662
6	98	3.27e−13	314.2224
7	251	1.7373e−12	16301.7894
8	NA	NA	NA

Finally, the CANDECOMP-algorithm from Tensorlab is applied to all symmetric tensors of Experiment 3. This algorithm allows each rank-1 term to be symmetric as well. In Table 4.5 the total number of computed terms, the residual and total run time in seconds are listed. As with Regalia's iterative method, for $d = 8$ the algorithm also fails due to the intermediate result being too large. Since we are interested in a full decomposition we set the total number of desired rank-1 terms equal to the number of terms obtained from the STEROID. This does surprisingly not result in small residuals for the CANDECOMP-algorithm. The total run time is highly variable over the different orders and for the $d = 6$ case 4 times slower compared to the symmetry exploiting STEROID algorithm.

5. Application.

In this section we show how STEROID readily solves a problem in nonlinear block-structured system identification [10] and nonlinear state-space identification [20]. As illustrated in Figure 5.1, the goal is to recover the internal structure of an identified static polynomial mapping

$$\left\{\begin{array}{lcl} y_1(t) & = & f_1(u_1(t), \ldots, u_p(t)), \\ \vdots & & \\ y_l(t) & = & f_l(u_1(t), \ldots, u_p(t)), \end{array}\right. \tag{5.1}$$

which relates the p inputs $u_1(t), \ldots, u_p(t)$ to l outputs $y_1(t), \ldots, y_l(t)$. This internal structure is determined by writing each of these multivariate polynomials $f_1, \ldots, f_l$

TABLE 4.5
Number of terms, residuals and run times for Tensorlab's iterative method.

d	total number of terms R	$\|\|\mathcal{A} - \sum_{i=1}^{R} a_i^d\|\|_F$	total runtime [seconds]
3	50	2.7187e−9	0.1480
4	75	5.8317	9.1233
5	750	1.1831e−5	21.4996
6	1750	12.7366	1793.0454
7	2675	0.0909	802.7768
8	5550	NA	NA

as a linear combination of univariate polynomials $g_j(x_j)$

$$f_i \;=\; \sum_{j=1}^{n} l_{ij}\, g_j(x_j) \quad (l_{ij} \in \mathbb{R}) \tag{5.2}$$

where each x_j is an affine transformation of the inputs

$$x_j = b_j + \sum_{k=1}^{p} t_{jk} u_k. \quad (b_j, t_{jk} \in \mathbb{R}). \tag{5.3}$$

The scalars b_j are typically called the bias or threshold. A slightly different version of this problem has been solved in [8, 30], where the conversion of the inputs $u_1, \ldots, u_p$ to the states $x_1, \ldots, x_n$ happens by means of a linear transformation.

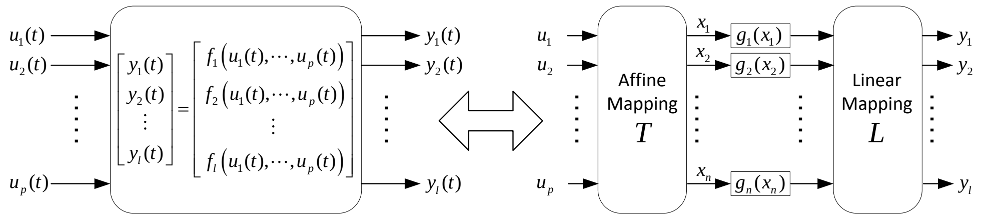
$u_1(t)$
$u_2(t)$
$u_p(t)$
$\begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_l(t) \end{bmatrix} = \begin{bmatrix} f_1(u_1(t),\cdots,u_p(t)) \\ f_2(u_1(t),\cdots,u_p(t)) \\ \vdots \\ f_l(u_1(t),\cdots,u_p(t)) \end{bmatrix}$
$y_1(t)$
$y_2(t)$
$y_l(t)$
u_1
u_2
u_p
Affine Mapping T
x_1
x_2
x_n
$g_1(x_1)$
$g_2(x_2)$
$g_n(x_n)$
Linear Mapping L
y_1
y_2
y_l

FIG. 5.1. *The polynomial mapping $f_1, \ldots, f_l$ is decoupled into a set of parallel univariate polynomials $g_1, \ldots, g_n$ by means of an affine transformation T and linear transformation L.*

The coefficients b_j, t_{jk} will turn out to be the entries of the eigenvectors v_j of a STEROID and similarly the l_{ij} coefficients are the real l coefficients of a STEROID. We now show how this comes about. It is important to observe here that we replace our earlier notation of x_j as the eigenvectors computed by the STEROID algorithm with v_j in order to avoid confusion with the internal state variables x_j of the nonlinear system. Let d be the maximal total degree of the polynomial system (5.1). In order to estimate all coefficients b_j, t_{jk}, we first need to make sure that each of the polynomials (5.1) is homogeneous of degree d. This is achieved by introducing the homogenization variable $u_0(t)$, which satisfies $u_0(t) = 1\, \forall\, t \in \mathbb{R}$. For example, if we have $d = 4$ and the polynomial $f_1 = u_1^2 + 5u_1u_2 - 9$, then its homogenization is $f_1^h = u_0^2u_1^2 + 5u_0^2u_1u_2 - 9u_0^4$. The natural isomorphism between homogeneous polynomials and symmetric tensors allows us then to write the homogeneous polynomials $f_1^h, \ldots, f_l^h$ as symmetric tensors

$\mathcal{F}_1, \ldots, \mathcal{F}_l$. Each of these tensors is of order d and has dimension $p+1$, due to the extra homogenization variable. From the application of the STEROID Algorithm onto each symmetric tensor $\mathcal{F}_1, \ldots, \mathcal{F}_l$, a set of pure power vectors v_j is obtained. From these vectors v_j a basis X can be constructed such that each symmetric tensor $\mathcal{F}_1, \ldots, \mathcal{F}_l$ can be decomposed in terms of this basis. The l_{ij} coefficients are then found from solving the least-squares problems

$$l_i \;=\; \underset{l}{\operatorname{argmin}} \; ||\operatorname{vec}(\mathcal{F}_i) - X\, l||_F \quad (i = 1, \ldots, l).$$

The STEROID decomposition

$$\mathcal{F}_i \;=\; \sum_{j=1}^{N} l_{ij}\, v_j^d,$$

can then be written in terms of homogeneous polynomials as

$$f_i^h \;=\; \sum_{j=1}^{N} l_{ij}\, (\sum_{k=0}^{p} t_{jk} u_k)^d. \tag{5.4}$$

Setting the homogenization variable $u_0(t) \triangleq 1$ effectively de-homogenizes all homogeneous polynomials f_i^h into

$$f_i \;=\; \sum_{j=1}^{N} l_{ij}\, (t_{i0} + \sum_{k=1}^{p} t_{jk} u_k)^d. \tag{5.5}$$

By retaining the n vectors v_j corresponding with nonzero l_{ij}'s over all i's and introducing the definitions $g_j(x_j) \triangleq x_j^d$ and $b_j \triangleq t_{i0}$ into (5.5), the problem of reconstructing the internal structure of the nonlinear system as given by (5.2) and (5.3) is completely solved. The whole algorithm is summarized in pseudo-code in Algorithm 5.1.

```
ALGORITHM 5.1. nonlinear block-structured system identification
Input: multivariate polynomials $f_1, \ldots, f_l$
Output: affine transformation $T$, linear transformation $L$
  $d \leftarrow$ maximal total degree of $f_1^h, \ldots, f_l^h$
  homogenize all $f_1, \ldots, f_l$ into $f_1^h, \ldots, f_l^h$ of degree $d$
  for $i = 1, \ldots, l$ do
    $\mathcal{F}_i \leftarrow$ symmetric tensor corresponding with $f_i^h$
    $V_i \leftarrow$ STEROID($\mathcal{F}_i$)
  end for
  $X \leftarrow$ construct basis from all $V_i$ vectors
  for $i = 1, \ldots, l$ do
    $l_i = \underset{l}{\operatorname{argmin}} \; ||\operatorname{vec}(\mathcal{F}_i) - X\, l||_F$
  end for
  $L \leftarrow$ collect all nonzero $l_{ij}$ coefficients
  $T \leftarrow$ retain only $n$ vectors from $V$ corresponding with $L$
```

The following example illustrates the whole identification algorithm in detail.

EXAMPLE 5.1. *Consider the nonlinear 2-input-2-output system described by the polynomials f_1, f_2 of total degree $d = 3$*

$$\begin{aligned} f_1 &= 54u_1^3 - 54u_1^2u_2 + 8u_1^2 + 18u_1u_2^2 + 16u_1u_2 - 2u_2^3 + 8u_2^2 + 8u_2 + 1, \\ f_2 &= -27u_1^3 + 27u_1^2u_2 - 24u_1^2 - 9u_1u_2^2 - 48u_1u_2 - 15u_1 + u_2^3 - 24u_2^2 - 19u_2 - 3. \end{aligned}$$

After homogenization we obtain

$$\begin{aligned} f_1^h &= 54u_1^3 - 54u_1^2u_2 + 8u_0u_1^2 + 18u_1u_2^2 + 16u_0u_1u_2 - 2u_2^3 + 8u_0u_2^2 \\ &\quad +8u_0^2u_2 + u_0^3, \\ f_2^h &= -27u_1^3 + 27u_1^2u_2 - 24u_0u_1^2 - 9u_1u_2^2 - 48u_0u_1u_2 - 15u_0^2u_1 + u_2^3 - 24u_0u_2^2 \\ &\quad -19u_0^2u_2 - 3u_0^3. \end{aligned}$$

The homogeneous polynomials f_1^h and f_2^h are converted into the symmetric third order tensors $\mathcal{F}_1, \mathcal{F}_2 \in \mathbb{R}^{3\times3\times3}$. Each application of the STEROID algorithm results in 15 v_j vectors, from which a basis X of 30 vectors is constructed. The least-squares problem (2.5) *is then solved for its minimum norm solution l_1, l_2, with residuals* 2.30×10^{-14} *and* 2.36×10^{-14} *respectively. Both l_1 and l_2 contains 10 nonzero entries. Setting $u_0 = 1$ we obtain*

$$\begin{aligned} f_1 &= \textstyle\sum_{j=1}^{10} l_{1j}\,(b_j + \sum_{k=1}^{2} t_{jk}u_k)^3, \\ f_2 &= \textstyle\sum_{j=1}^{10} l_{2j}\,(b_j + \sum_{k=1}^{2} t_{jk}u_k)^3. \end{aligned}$$

6. Conclusions and Remarks.

A constructive decomposition algorithm, named STEROID, has been proposed to decompose a symmetric tensor into a real linear combination of symmetric unit-norm rank-1 tensors. The method exploits symmetry and permits an efficient computation, e.g. via the symmetric QR algorithm or divide-and-conquer method, in subsequent reshapings and foldings of intermediate symmetric matrices. In contrast to other iterative methods, STEROID does not require any initial guess and and can handle large symmetric tensors. The original STEROID algorithm works with symmetric tensors whose order is a power of two, whereas an innovative tensor embedding technique is developed to remove this constraint and allows the computation of a STEROID for arbitrary orders. In addition, two methods are discussed that reduce the size of the least-squares problem, thereby increasing the feasibility to tackle large-size problems. Numerical examples have verified the high efficiency and scalability of STEROID and have demonstrated its superior performance in comparison to existing iterative methods. Finally, it was shown how STEROID can be used to decouple a set of multivariate polynomials into a collection of univariate polynomials in the setting of block-structured nonlinear system identification.

REFERENCES

[1] K. Batselier, H. Liu, and N. Wong, *A constructive algorithm for decomposing a tensor into a finite sum of orthonormal rank-1 terms*, ArXiv e-prints, (2014).

[2] J. Brachat, P. Comon, B. Mourrain, and E. P. Tsigaridas, *Symmetric tensor decomposition*, Linear Algebra Appl., 433 (2010), pp. 1851–1872.

[3] J.D. Carroll and J.-J. Chang, *Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition*, Psychometrika, 35 (1970), pp. 283–319.

[4] T. Coleman and Y. Li, *A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables*, SIAM J. Optimiz., 6 (1996), pp. 1040–1058.

[5] P. Comon, G. Golub, L.-H. Lim, and B. Mourrain, *Symmetric tensors and symmetric tensor rank.*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1254 – 1279.

[6] P. Comon and M. Rajih, *Blind identification of under-determined mixtures based on the characteristic function*, Signal Processing, 86 (2006), pp. 2271 – 2281. Special Section: Signal Processing in {UWB} Communications.

[7] L. De Lathauwer, J. Castaing, and J. Cardoso, *Fourth-order cumulant-based blind identification of underdetermined mixtures*, IEEE T. Signal Proces., 55 (2007), pp. 2965–2973.

[8] P. Dreesen, M. Ishteva, and J. Schoukens, *Decoupling multivariate polynomials using first-order information and tensor decompositions*, SIAM J. Matrix Anal. Appl., (2015). Accepted for publication (preprint arXiv:1410.4060 [math.NA]).

[9] A. Ferreol, L. Albera, and P. Chevalier, *Fourth-order blind identification of underdetermined mixtures of sources (FOBIUM)*, IEEE T. Signal Proces., 53 (2005), pp. 1640–1653.

[10] F. Giri and E.-W. Bai, *Block-oriented nonlinear system identification*, vol. 1, Springer, 2010.

[11] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, 3rd ed., Oct. 1996.

[12] M. Ishteva, P.-A. Absil, and P. Van Dooren, *Jacobi algorithm for the best low multilinear rank approximation of symmetric tensors*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 651–672.

[13] E. Kofidis and P. A. Regalia, *On the best rank-1 approximation of higher-order supersymmetric tensors*, SIAM J. Matrix Anal. Appl., 23 (2002), pp. 863–884.

[14] T.G. Kolda, *Orthogonal tensor decompositions*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 243–255.

[15] T.G. Kolda and B.W. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.

[16] T. G. Kolda and J. R. Mayo, *Shifted power method for computing tensor eigenpairs*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 1095–1124.

[17] ———, *An adaptive shifted power method for computing generalized tensor eigenpairs*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 1563–1581.

[18] JM Landsberg, *Tensors: geometry and applications*, vol. 128, American Mathematical Society, 2012.

[19] I. Oseledets, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317.

[20] Johan Paduart, Lieve Lauwers, Jan Swevers, Kris Smolders, Johan Schoukens, and Rik Pintelon, *Identification of nonlinear systems using polynomial nonlinear state space models*, Automatica, 46 (2010), pp. 647–656.

[21] MATLAB R2012a, *The Mathworks Inc.*, 2012. Natick, Massachusetts.

[22] P.A Regalia, *Monotonically convergent algorithms for symmetric tensor approximation*, Linear Algebra Appl., 438 (2013), pp. 875 – 890.

[23] P.A. Regalia and E. Kofidis, *The higher-order power method revisited: convergence proofs and effective initialization*, in Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on, vol. 5, 2000, pp. 2709–2712 vol.5.

[24] J. Salmi, A. Richter, and V. Koivunen, *Sequential unfolding svd for low rank orthogonal tensor approximation*, in Signals, Systems and Computers, 2008 42nd Asilomar Conference on, Oct 2008, pp. 1713–1717.

[25] B. Savas and L.-H. Lim, *Quasi-newton methods on grassmannians and multilinear approximations of tensors*, SIAM J. Sci. Comput., 32 (2010), pp. 3352–3393.

[26] M. Schoukens and Y. Rolain, *Cross-term elimination in parallel wiener systems using a linear input transformation*, IEEE T. Instrum. Meas., 61 (2012), pp. 845–847.

[27] L. Sorber, M. Van Barel, and L. de Lathauwer, *Tensorlab Version 2.0.* Available online, January 2014.

[28] R. Tibshirani, *Regression shrinkage and selection via the lasso*, J. Roy. Stat. Soc. B, 58 (1994), pp. 267–288.

[29] K. Tiels and J. Schoukens, *From coupled to decoupled polynomial representations in parallel wiener-hammerstein models*, in Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on, IEEE, 2013, pp. 4937–4942.

[30] K. Usevich, *Decomposing multivariate polynomials with structured low-rank matrix completion*, in Proc. 21st MTNS, July 2014, pp. 1826–1833.

[31] A. Van Mulders, J. Schoukens, and L. Vanbeylen, *Identification of systems with localised nonlinearity: from state-space to block-structured models*, Automatica, 49 (2013), pp. 1392–1396.