SYMMETRIC TENSOR DECOMPOSITION

Jerome Brachat*, Pierre Comon†, Bernard Mourrain*, Elias Tsigaridas†

*Laboratoire I3S, UMR6070 CNRS, UNS,
2000, route des Lucioles, BP 121
06903 Sophia Antipolis Cedex - France,
pcomon@unice.fr

†Laboratoire I3S, UMR6070 CNRS, UNS,
2000, route des Lucioles, BP 121
06903 Sophia Antipolis Cedex - France,
firstname.lastname@sophia.inria.fr

ABSTRACT

We present an algorithm for decomposing a symmetric tensor of dimension $n$ and order $d$ as a sum of rank-1 symmetric tensors, extending the algorithm of Sylvester devised in 1886 for symmetric tensors of dimension 2. We exploit the known fact that every symmetric tensor is equivalently represented by a homogeneous polynomial in $n$ variables of total degree $d$. Thus the decomposition corresponds to a sum of powers of linear forms.

The impact of this contribution is two-fold. First it permits an efficient computation of the decomposition of any tensor of sub-generic rank, as opposed to widely used iterative algorithms with unproved convergence (e.g. Alternate Least Squares or gradient descent). Second, it gives tools for understanding uniqueness conditions, and for detecting the tensor rank.

1. INTRODUCTION

Symmetric tensors show up in applications mainly as high-order derivatives of multivariate functions. For instance in Statistics, cumulant tensors are derivatives of the second characteristic function [1]. Tensors have been widely utilized in Electrical Engineering since the nineties, because of the use of High-Order Statistics [2] [3] [4] [5] [6] [7]. Even earlier in the seventies, tensors have been used in Chemometrics [8] or psychometrics [9]. Another important application field is Data Analysis. For instance, Independent Component Analysis was originally introduced for symmetric tensors whose rank did not exceed dimension [10] [11]. Now, it has become possible to estimate more factors than the dimension. Further references may be found in [12] [13], and numerous applications of tensor decompositions may be found in [14] [15].

The goal of this paper is to devise an algebraic technique able to decompose a symmetric tensor of arbitrary order and dimension in an essentially unique manner (i.e. up to scale and permutation) into a sum of rank-one terms. Of course, reaching such a goal requires some conditions, in particular related to its rank, which must be sub-generic. Our algorithm could be seen as an extension of the SVD algorithm from matrices to $n$-way arrays. We exploit the strong connection of symmetric tensors and homogeneous polynomials. This approach allows us to use effective algebraic geometry techniques, and to tackle the problem of decomposition using Veronese varieties, duality of vector spaces, and algorithms for polynomial system solving. To the best of our knowledge this is the first time that a decomposition algorithm for symmetric tensors is presented.

The rest of the paper is structured as follows: In the remaining of the section we present some historical remarks and we shed light to the connection of symmetric tensors and homogeneous polynomials. Sec. 2 presents Sylvester’s approach for the binary case. In Sec. 3 we exploit many different, albeit equivalent, algebraic formulations of the decomposition problem, as well as the necessary algebraic tools. In Sec. 4 we present the algorithm and illustrate it with an example.

1.1 Historical remarks

Despite their obvious practical interest as they can deal with inexact data, numerical algorithms presently used in most scientific communities are suboptimal, in the sense that they either do not fully exploit symmetries, minimize different successive criteria sequentially, or are iterative and lack a guarantee of global convergence. In addition, they often request the rank to be much smaller than generic.

On the other hand, the algorithm based on Sylvester’s theorem [22], recalled in section 2, provides a complete answer to the questions of uniqueness and computation, for any order [29]. However, the latter is devoted to 2-dimensional symmetric tensors, and techniques based on pairwise processing have a very limited range of use when the rank exceeds the dimension. For the decomposition of fourth-order tensors we refer the reader to [18].

The algorithm proposed in this paper is inspired from Sylvester’s theorem, and extends its principle to larger dimensions. In addition, it fully exploits symmetry, and when the solution is essentially unique, it provides the decomposition for any sub-generic rank.

1.2 Tensors and Polynomials

Any symmetric tensor of dimension $n$, i.e. the range of each index, and order $d$, i.e. the number of indices, can be associated with a homogeneous polynomial in $n$ variables of degree $d$. For instance, a third order tensor $T_{ijk}$ can be associated with the polynomial

$$
\sum_{i,j,k} c(i,j,k) x_{ij} x_{jk} x_{ki},
$$

where $c(\cdot)$ denotes some fixed symmetric function. See e.g. [23] for further details. We consider a homogeneous polynomial $F^h(x)$

$$
F^h(x) = \sum_{j_0+j_1+\ldots+j_n=d} a_{j_0,j_1,\ldots,j_n} x_0^{j_0} x_1^{j_1} \cdots x_n^{j_n}. \quad (1)
$$
Our goal is to compute a decomposition of \( F^h \) as a sum of \( d \)th powers of linear forms, 
\[
F^h(x) = \sum_{i=1}^{r} (\lambda_i (k_i, x) - x, x x_{n+1}^d)^d = \lambda_1 k_1(x)^d + \lambda_2 k_2(x)^d + \cdots + \lambda_r k_r(x)^d,
\]
where \( \lambda_i \in \mathbb{C} \), such that \( r \) is the smallest possible. This smallest \( r \) is often referred to as the tensor rank, or sometimes the polynomial width.

Let’s see how the decomposition is made possible in the case of homogeneous polynomials in two variables only.

2. THE BINARY CASE

Let \( p(x_1, x_2) = \sum_{i=0}^{d} c_i x_1^i x_2^{d-i} \) be a homogeneous polynomial of degree \( d \) in 2 variables. Denote \( H[r] \) the Hankel matrix of dimensions \( (d-r+1) \times (r+1) \) with entries \( H[r]_{ij} = c_{i+j-2} \):

\[
H[r] = \begin{bmatrix} c_0 & c_1 & \cdots & c_r \\ \vdots & \vdots & \ddots & \vdots \\ c_{d-r} & \cdots & c_{d-1} & c_d \end{bmatrix}.
\]

Then we have:

**Sylvester, 1886** \( p(x_1, x_2) \) can be written as a sum of \( d \)th powers of \( r \) distinct linear forms in \( \mathbb{C} \) as:

\[
p(x_1, x_2) = \sum_{j=1}^{r} (\alpha_j x_1 + \beta_j x_2)^d,
\]

and if only if (i) there exists a vector \( q \) of dimension \( r+1 \), with components \( q_{\ell} \), such that

\[
H[r] q = 0.
\]

and (ii) the polynomial \( q(x_1, x_2) = \sum_{\ell=0}^{r} q_{\ell} x_1^\ell x_2^{d-\ell} \) admits \( r \) distinct roots, i.e. it can be written as

\[
q(x_1, x_2) = \prod_{j=1}^{r} (\beta_j x_1 - \alpha_j x_2).
\]

It turns out that the proof of this theorem is constructive \([24],[23],[25]\) and yields the algorithm below.

1. Initialize \( r = 0 \)
2. Increment \( r \leftarrow r + 1 \)
3. If the row rank of \( H[r] \) is full, then go to step 2
4. Else compute a basis \( \{k_1, \ldots, k_r\} \) of the right kernel of \( H[r] \).
5. Specialization (pick a random vector in the kernel):
   - Take a generic vector \( q \) in the kernel, e.g. \( q = \sum_i \mu_i k_i \)
   - Compute the roots of the associated polynomial \( q(x_1, x_2) = \sum_{\ell=0}^{r} q_{\ell} x_1^\ell x_2^{d-\ell} \). Denote them \( (\beta_j, -\alpha_j) \), where \( |\alpha_j|^2 + |\beta_j|^2 = 1 \).
   - If the roots are not distinct in \( \mathbb{P}^2 \), try another specialization. If distinct roots cannot be obtained, go to step 2
   - Else if \( q(x_1, x_2) \) admits \( r \) distinct roots then compute the coefficients \( \lambda_j, 1 \leq j \leq r \), by solving the linear system below

\[
\begin{bmatrix} \alpha_1^d & \cdots & \alpha_r^d \\ \alpha_1^{d-1} \beta_1 & \cdots & \alpha_r^{d-1} \beta_r \\ \vdots & \ddots & \vdots \\ \beta_1^d & \cdots & \beta_r^d \end{bmatrix} \lambda = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_d \end{bmatrix}.
\]

6. The decomposition is \( p(x_1, x_2) = \sum_{j=1}^{r} \lambda_j k_j(x)^d \), where \( k_j(x) = (\alpha_j x_1 + \beta_j x_2) \).

Note that step 5 is a specialization only if the dimension of the right kernel is larger than one, which will not occur for ranks smaller than generic.

The goal is now to extend this kind of numerical algorithm to polynomials in more variables. This problem was open until now.

3. PROBLEM FORMULATIONS

**Notation.** If \( a = (a_1, \ldots, a_n) \) is a vector in \( \mathbb{N}^n \), then \( |a| \) is the sum of its elements, i.e. \( |a| = \sum_i a_i \). By \( x^a \) we denote the monomial \( x_1^{a_1} \cdots x_n^{a_n} \).

Let \( R \) be the ring of polynomials \( \mathbb{C}[x_1, \ldots, x_n] \), while \( R_d \) will denote the vector space of polynomials of (total) degree at most \( d \). The set \( \{x^a\}_{|a| \leq d} = \{x_1^{a_1} \cdots x_n^{a_n}\}_{a_1 + \cdots + a_n \leq d} \) represents the elements of the monomial basis of the vector space \( R_d \). It contains \( (n+d-1) \) elements. The corresponding basis of the dual space \( R_d^* \), that is the set of linears that compute the coefficients of a polynomial in the primal basis, is the set \( \{d^a\}_{|a| \leq d} \), where \( d^a : R_d \rightarrow \mathbb{C} \) and

\[
d^a(f) = \frac{1}{\prod_{i=1}^{n} a_i!} \left( \frac{d^{a_1}}{dx_1} \cdots \frac{d^{a_n}}{dx_n} f \right)(0).
\]

The superscript \( h \) denotes the homogeneous version of the polynomial. Let \( S \) be the set of homogeneous polynomials in \( n+1 \) variables, \( S_d \) represents the homogeneous polynomials of degree \( d \), and \( \mathbb{P}(S_d) \), the corresponding projective space. Similarly interpretations hold for the dual spaces \( \hat{S} \) and \( \hat{S}_d \). Analogous to the affine case, we can define primal and dual bases for the homogeneous case.

3.1 Direct approach by polynomial fitting

The first idea is merely to solve, in a given polynomial basis, the polynomial system, induced by the equation

\[
F^h(x) - \sum_i k_i(x)^d = 0,
\]

with respect to the coefficients of the linear forms \( k_i \). We call this the direct approach. In the tensor framework, even if the rank is supposed to be known, attempts to solve this problem have not entailed efficient algorithms (cf. section \( \|$). In the polynomial framework, it is easy to see that we end up with an over-determined polynomial system of \((n+d)\) equations in \(r(n+1)\) unknowns. This description of the problem is not optimal, since it introduces \( r! \) redundant solutions corresponding to permutations of the linear forms. Another drawback is that polynomials involved are of degree \( d \) in the coefficients \( k_{i,n} \), which are too high from the computational point of view. In fact, our approach does not involve the solution of polynomial systems of degree higher than 2.

3.2 Different views using duality

We consider the following Veronese map of degree \( d \)

\[
\nu : S_1 \rightarrow S_d \quad k(x) \mapsto k(x)^d.
\]
which sends a linear (homogeneous) polynomial to its $d$–th power. Recall that the (monomial) basis of $S_1$ is the set \[ \{ x_0, x_1, \ldots, x_n \} \], while the basis of $S_d$ is the set \[ \{ x_0^{a_0} x_1^{a_1} \cdots x_n^{a_n} \}_{a_0 + \cdots + a_n = d} \], viz. the set of all the monomials in $x_0, x_1, \ldots, x_n$ of total degree $d$. The cardinality of the basis is \( (n+d) - 1 \). Under the action of $\nu$, a linear polynomial $k(x) = k_0 x_0 + \cdots + k_n x_n$ corresponds to $k(x)^d = \sum_{i_0 + \cdots + i_n = d} k_0^{i_0} \cdots k_n^{i_n} x_0^{i_0} \cdots x_n^{i_n}$. In terms of vectors, $k = [k_0, \ldots, k_n]^\top$ corresponds to the vector $[\ldots, (i_0, \ldots, i_n) k_0^{i_0} \cdots k_n^{i_n}, \ldots]^\top$.

Another Veronese map, also of degree $d$, is

$$
\delta : C^{n+1} \rightarrow \hat{S}_d
$$

which sends a point $z = (z_0, z_1, \ldots, z_n)$ to $[z_0, \ldots, z_0^{d} \cdots z_n^{d}]$. Recall that the linear functionals \( \{ \Phi |_{i=0} \} \) consist a basis of $\hat{S}_d$. It holds that $S_1 \cong C^{n+1}$. The map $\tau$

$$
\tau : \hat{S}_d \rightarrow S_d
$$

$$
[\ldots, z_0^{d} \cdots z_n^{d}]^\top \mapsto [\ldots, (i_0, \ldots, i_n) z_0^{i_0} \cdots z_n^{i_n}]^\top
$$

is an isomorphism. The map remains an isomorphism even if we restrict it to the images of the maps $\nu$ and $\delta$, that is $\nu(S_1)$ and $\delta(C^n)$, respectively. The inverse of $\tau$ is the map $\tau^{-1} : S_d \rightarrow \hat{S}_d$. Consider a polynomial in $S_d$, that is the $d$–th power of a linear form, say $k(x)^d$. If we apply the map $\tau^{-1}$ to this polynomial, then we have that $\tau^{-1}(k(x)^d) = 1_k$; that is the linear form that gives the evaluation of a polynomial (of degree $d$) over the point $k = [k_0, k_1, \ldots, k_n]^\top$.

Let us now revisit the problem of decomposition. Initially we are given a polynomial $F^h(x) \in S_d$. The decomposition $F^h(x) = \sum_{i=0}^{r} \lambda_i k_i(x)^d$ corresponds to a secant variety in $\nu(S_1) \subset \hat{S}_d$. Using the properties of $S_1$ and $\nu$, we can gain another view of the problem. If we apply $\tau^{-1}$ to $F^h$ we compute its dual, that is $\Phi = \tau^{-1}(F^h)$. The decomposition of the latter, i.e. $\Phi = \sum_{i=0}^{r} \lambda_i 1_{k_i}$, is a linear combination of elements in $\delta(C^n) \subset \hat{R}_d$.

Overall, it holds that

$$
\tau^{-1}(F^h(x)) = \tau^{-1} \left( \sum_{i=0}^{r} \lambda_i k_i(x)^d \right) = \sum_{i=0}^{r} \lambda_i \tau(1_{k_i}) = \sum_{i=0}^{r} \lambda_i \tau(1_{k_i}) = \sum_{i=0}^{r} \lambda_i 1_{k_i} = \Phi.
$$

Moreover, $\tau(\Phi) = \tau(\sum_{i=0}^{r} \lambda_i 1_{k_i}) = \sum_{i=0}^{r} \lambda_i \tau(1_{k_i}) = \sum_{i=0}^{r} \lambda_i k_i(x)^d = F^h(x)$.

All the previous views of the decomposition problem are equivalent. The results obtained for any of them could be translated for the other.

### 3.3 Quotient algebra and duality

The idea of the algorithm is to exploit the properties of $\Phi \in \hat{R}$, that we assume that is known up to degree $d$. More precisely, we consider the symmetric bilinear form $H_\Phi : (p, q) \mapsto \Phi(pq)$, the matrix of which in the monomial basis is $(\Phi(x^a y^b))_{a,b \in \mathbb{N}^n}$. Let $I_\Phi$ be the kernel of $H_\Phi$.

**Proposition 3.2** If $\Phi = \sum_{i=0}^{r} \lambda_i 1_{k_i}$ with $\lambda_i \neq 0$ and $k_i \in C^n$, then $p \in I_\Phi$ iff $p(k_i) = 0$ for $i = 1, \ldots, r$.

In other words, the common roots of all the polynomials in $I_\Phi$ define the linear terms in the tensor decomposition of $F$.

In order to compute the zeros of $I_\Phi$, we may use a well-known theorem (see e.g. [26], [27], [28]), which we apply to the zero-dimensional ideal $I_\Phi$:

**Theorem 3.3** The eigenvalues of the matrices $M_i$ and $R^i_\Phi$, of the linear operators that correspond to the multiplication by $a$ in $\hat{R}$ modulo $I_\Phi$, and its transposed, are $\{a(k_1), \ldots, a(k_r)\}$. The common eigenvectors of the matrices (the $M_i$) are (up to a scalar) $1_{k_i}$, $i = 1, \ldots, r$.

If we denote by $H^E_\Phi$ the restriction of $H_\Phi$ to a vector space $E$ of dimension $r$ on which $H_\Phi$ is invertible, we have the relation $H^E_\Phi \cdot M_i = M_i^E H^E_\Phi$, where $H_\Phi : (p, q) \mapsto \Phi(p \cdot q)$. The solution of the generalized eigenvalue problem $(H^E_\Phi - \lambda \cdot I) \Phi = 0$ yields the eigenvector $H^E_\Phi^E \Phi$ of $M_1^E$, which is by Th. 3.3 the evaluations $1_{k_i}$. From these eigenvectors, we deduce the linear factors in the tensor decomposition. The coefficients $\lambda_i$ $(i = 1, \ldots, r)$ can then be computed by solving a linear system of size $r$.

### 4. ALGORITHM

The algorithm that we will present for decomposing a symmetric tensor as sum of rank 1 symmetric tensors generalizes the algorithm of Sylvester [24], devised for dimension 2 tensors, see also [29].

#### 4.1 Overview

**Algorithm 1:** Symmetric Tensor Decomposition

**Input:** A homogeneous polynomial $f(x_0, x_1, \ldots, x_n)$ of degree $d$.

**Output:** A decomposition of $f$ as $f = \sum_{r=1}^{r} \lambda_i k_i(x)^d$ with $r$ minimal.

- Compute the coefficients of $f'$: $c_\alpha = a_\alpha (d^{-1})$, for $|\alpha| \leq d$, $\alpha = (\alpha_1, \ldots, \alpha_n)$;
- $r := 1$;
- Repeat
  1. Compute a set $B$ of monomials of degree $\leq d$ connected to 1 with $\{B\} = r$;
  2. Find parameters $h$ s.t. $\det(H^E_\Phi) \neq 0$ and the operators $M_i = H^E_\Phi^{-1} H^E_{\Phi(h)}$ commute.
  3. If there is no solution, restart the loop with $r := r + 1$.
- Else compute the $n \times r$ eigenvalues $\zeta_{ij}$ and the eigenvectors $v_j$, s.t. $M_i v_j = \zeta_{ij} v_j$, $i = 1, \ldots, n$, $j = 1, \ldots, r$.

until the eigenvalues are simple.

Let us briefly comment on the computation process. A basis connected to 1, is a basis containing 1 where each
element different from 1, is the product of a variable by another element of the basis. Consider the homogeneous polynomial $f(x)$ in $\mathbb{L}$ that we want to decompose. We may assume without loss of generality, that for at least one variable, say $x_0$, all its coefficients in the decomposition are non zero, i.e. $k_{i,0} \neq 0$, for $1 \leq i \leq r$. We dehomogenize $f$ with respect to this variable and we denote this polynomial by $f^* := f(1,x_1,\ldots,x_n)$. We want to decompose the polynomial $f^*(x) \in R_d$ as a sum of powers of linear forms, i.e. $f(x) = \sum_{i=1}^{r} \lambda_i (1+k_{i,1} x_1 + \cdots + k_{i,n} x_n)^d$. Equivalently, we want to decompose its corresponding dual element $f^* \in \hat{R}_d$ as a linear combination of evaluations over the distinct points $k_i := (k_{i,1},\ldots,k_{i,n})$: $f^* = \sum_{i=1}^{r} \lambda_i \mathbb{1}_{k_i}$ (we refer the reader to Section 3).

Assume that we know the value of $r$. If we know the value of $\Phi$ on polynomials of degree high enough, it allows us to compute the tables of multiplication modulo the kernel of $H_\Phi$. By Theorem 3.3 if we solve the generalized eigenvector problem $(H_\Phi, \Phi - \zeta H_\Phi)\nu = \nu$, then we can recover the points of evaluation $k_i$. By solving a linear system, we will then deduce the value of $\lambda_1,\ldots,\lambda_r$. For certain (big) values of $r$ it can happen that not all the elements of the corresponding matrices are known. In this case, we use the property that the matrices of multiplication commute, and we form a system, the solutions of which are these unknown elements. We refer the reader to [12] [13] for details, and we present an example to illustrate the algorithm.

4.2 Example

(1) Convert the symmetric tensor to the corresponding homogeneous polynomial.

Assume that we have given a tensor of dimension 3 and order 5, and that the corresponding homogeneous polynomial is $f = -1549440 x_0 x_1 x_2^2 + 2417040 x_0 x_1^2 x_2^3 + 166320 x_0^2 x_1 x_2^2 - 829440 x_0 x_1 x_2 - 5760 x_0 x_1 x_2^2 - 222480 x_0 x_1^2 x_2 + 38 x_0^2 - 497664 x_1^5 - 1107804 x_2^5 - 120 x_0 x_1 + 180 x_0 x_2 + 12720 x_1^2 x_2 + 8220 x_0 x_2^2 - 34560 x_0 x_1^2 - 59160 x_0 x_2^3 + 831840 x_1 x_2^4 + 442950 x_0 x_2 - 5591502 x_1 x_2 + 7983360 x_1^3 x_2^2 - 9653040 x_1^2 x_2^2 + 5116680 x_1 x_2^3.

(2) Compute the matrix of the linear form.

We form a $(n+d-1) \times (n+d-1)$ matrix, the rows and the columns of which correspond to the evaluation of the dual of the polynomial over all the monomial $\{x^a\}_{|a| \leq d}$, using the map $a_0 j_0 \ldots j_n \mapsto a_0 j_0 \ldots j_n \frac{\partial^d}{\partial x_0^d} |_{x_0 = 1} j_1 \ldots j_n$, where $a_0 j_0 \ldots j_n$ is the coefficient of the monomial $x_0^{a_0} \cdots x_n^{a_n}$.

Part of the corresponding matrix follows:

<table>
<thead>
<tr>
<th></th>
<th>$x_0^2$</th>
<th>$x_0 x_1$</th>
<th>$x_0 x_2$</th>
<th>$x_1^2$</th>
<th>$x_1 x_2$</th>
<th>$x_2^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>-24</td>
<td>-24</td>
<td>1272</td>
<td>-298</td>
<td>872</td>
<td>-3456</td>
</tr>
<tr>
<td>-24</td>
<td>1272</td>
<td>-288</td>
<td>-3456</td>
<td>-7416</td>
<td>5544</td>
<td>166368</td>
</tr>
<tr>
<td>36</td>
<td>-24</td>
<td>7416</td>
<td>5544</td>
<td>-5916</td>
<td>-41472</td>
<td>-80568</td>
</tr>
<tr>
<td>1272</td>
<td>-3456</td>
<td>166368</td>
<td>41472</td>
<td>80568</td>
<td>-497664</td>
<td></td>
</tr>
<tr>
<td>-288</td>
<td>41472</td>
<td>80568</td>
<td>-74772</td>
<td>-74772</td>
<td>-118304</td>
<td></td>
</tr>
<tr>
<td>5544</td>
<td>-5916</td>
<td>80568</td>
<td>-77472</td>
<td>85818</td>
<td>793836</td>
<td></td>
</tr>
<tr>
<td>-3456</td>
<td>-118304</td>
<td>965304</td>
<td>965304</td>
<td>-965304</td>
<td>-965304</td>
<td></td>
</tr>
<tr>
<td>144</td>
<td>-118304</td>
<td>965304</td>
<td>965304</td>
<td>-965304</td>
<td>-965304</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-118304</td>
<td>965304</td>
<td>965304</td>
<td>-965304</td>
<td>-965304</td>
<td></td>
</tr>
</tbody>
</table>

The whole matrix is $21 \times 21$. For reasons of space we present only the first 7 columns. Notice that we do not know the elements in some positions of the matrix (in the 7th column). In general we do not know the elements that correspond to monomials with (total) degree higher than 5.

(3) Extract a principal minor of full rank.

We should re-arrange the rows and the columns of the matrix so that the first principal minor is of full rank, $r$. We call this minor $H_\Phi$. In order to do that we try to put the matrix in row echelon form, using elementary row and column operations. In our example the $4 \times 4$ principal minor is of full rank, so there is no need for re-arranging the matrix. The matrix $H_\Phi$ is

$$H_\Phi = \begin{bmatrix} 38 & -24 & 36 & 1272 \\ -24 & 1272 & -288 & -3456 \\ 36 & -288 & 822 & -7416 \\ 1272 & -3456 & -7416 & 166368 \end{bmatrix}$$

Notice that the columns of the matrix correspond to the monomials $\{1, x_1, x_2, x_1^2\}$.

(4) We compute the ‘shifted’ matrix $H_{x_1,\Phi}$.

If the columns of $H_\Phi$ correspond to set of some monomials, say $\{x^a\}$, then the columns of $H_{x_1,\Phi}$ correspond to the set of monomials $\{x_1 x^a\}$. In our example

$$H_{x_1,\Phi} = \begin{bmatrix} -24 & 1272 & -288 & -3456 \\ -288 & -7416 & 5544 & -41472 \\ -3456 & 166368 & -41472 & 97664 \\ -118304 & 965304 & -965304 & -965304 \end{bmatrix}$$

the columns of which correspond to the monomials $\{x_1 x_1^2, x_1 x_2, x_1^2 x_2\}$, i.e. the monomials of $H_\Phi$, $\{1, x_1, x_2, x_1^2\}$, multiplied by $x_1$.

We assume for the moment that all the elements of the matrices $H_{x_1,\Phi}$ and $H_\Phi$ are known. If this is not the case, then we can compute the unknown entries of the matrix, using necessary and sufficient conditions of the quotient algebra; it holds that $M_x, M_{x_1} - M_x M_{x_1} = \mathbb{O}$. We refer the reader to [12] for details.

(5) We solve the equation $(H_{x_1,\Phi} - \zeta H_\Phi)\nu = 0$.

We solve the generalized eigenvalue/eigenvector problem using one of the well-known techniques [20]. We multiply the (generalized) eigenvectors by $H_\Phi$ and we normalize the resulting vectors so that the first element is 1, and we read the solutions from the coordinates of the (normalized) eigenvectors, according to Th. 4.3

In our example the normalized eigenvectors are

$$\begin{bmatrix} 1 \\ -12 \\ -3 \\ 144 \\ 144 \\ 144 \\ 144 \end{bmatrix}$$

The coordinates of the eigenvectors correspond to the elements $\{1, x_1, x_2, x_1^2\}$. Thus, we can recover the coefficients of $x_1$ and $x_2$ in the decomposition from coordinates of the eigenvectors. Thus, The polynomial admits a decomposition $f = \lambda_1 (x_0 - 12 x_1 - 3 x_2)^5 + \lambda_2 (x_0 - 2 x_1 + 3 x_2)^5 + \lambda_3 (x_0 + 2 x_1 + 3 x_2)^5 + \lambda_4 (x_0 + 12 x_1 + 13 x_2)^5$.

It remains to compute $\lambda_i$'s. We can do this easily by solving an over-constrained linear system, which we know that always has a solution, since the decomposition exists. Doing that, we deduce that $\lambda_1 = 5$, $\lambda_2 = 15$, $\lambda_3 = 15$ and $\lambda_4 = 3$. We obtain the following minimum decomposition of the polynomial as a sum of powers of linear forms: $f = 5(x_0 - 12 x_1 - 3 x_2)^5 + 15(x_0 - 2 x_1 + 3 x_2)^5 + 15(x_0 + 2 x_1 + 3 x_2)^5 + 3(x_0 + 12 x_1 + 13 x_2)^5$ that is the corresponding tensor is of rank 4.
5. CONCLUSIONS AND FUTURE WORK

We proposed an algorithm for symmetric tensor decomposition, extending the algorithm of Sylvester to dimensions higher than 2. The algorithm decomposes symmetric tensors when the rank is sub-generic and when the decomposition is unique. In order for the algorithm to work for any rank, we should be able to extend the quotient matrix defined in Sec. 3.3. We will report on this in the near future. We are currently working on an efficient C++ implementation of the algorithm.

We thank the anonymous referees for their comments that helped us improve the paper.

REFERENCES


