Enforcing symmetry in tensor network MIMO Volterra identification

Batselier, Kim

DOI
10.1016/j.ifacol.2021.08.404

Publication date
2021

Document Version
Final published version

Published in
IFAC-PapersOnline

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 54-7 (2021) 469–474

Enforcing symmetry in tensor network MIMO Volterra identification

Kim Batselier *

* *Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands.*

Abstract: The estimation of an exponential number of model parameters in a truncated Volterra model can be circumvented by using a low-rank tensor decomposition approach. This low-rank property of the tensor decomposition can be interpreted as the assumption that all Volterra parameters are structured. In this article, we investigate whether it is possible to explicitly enforce symmetry of the Volterra kernels to the low-rank tensor decomposition. We show that low-rank symmetric Volterra identification is an ill-conditioned problem as the low-rank property of the exact symmetric kernels cannot be upheld in the presence of measurement noise. Furthermore, an algorithm is derived to compute the symmetric Volterra kernels directly in tensor network form.

Keywords: Nonlinear system identification, truncated Volterra systems, tensors

1. INTRODUCTION

The identification of truncated Volterra series suffers from the curse of dimensionality. The total number of parameters to be identified grows exponentially with the order of the model, limiting the application of standard identification methods to weakly-nonlinear systems.

One way to lift this curse of dimensionality is by imposing additional structure onto the Volterra kernels. For example, Volterra kernels are generalizations of finite impulse responses to higher orders and are therefore expected to be smoothly decaying. Recent research enforces these constraints explicitly through regularization Birpoutsoukis et al. (2017, 2018), but these methods are unfortunately limited to third order kernels. Another way of adding structure to the kernels is by expanding them in terms of orthonormal basis functions Campello et al. (2004); Diouf et al. (2012). Tensor decompositions are also suitable candidates for adding structure through a low-rank constraint Shi and Townsend (2021). In Favier et al. (2012) both the canonical polyadic Harshman (1970); Carroll and Chang (1970) and Tucker tensor decompositions Tucker (1966) were used. The canonical polyadic decomposition, however, can suffer from numerical instability due to the ill-posedness of the problem of finding a best rank-r approximation de Silva and Lim (2008), and the determination of its rank is known to be an NP-hard problem Håstad (1990). The main disadvantage of the Tucker decomposition is that it still suffers from an exponential complexity. These issues were resolved in Batselier et al. (2017a,b) via low-rank tensor networks. Low-rank tensor networks also have the advantage that the complexity of estimating the model parameters scales linearly with the order of the truncated Volterra series. This linear scaling of the complexity opens up the possibility of identifying systems that are highly nonlinear and hence require high-order approximations. So far it remains an open question whether it is possible to enforce symmetry to the estimated Volterra kernel coefficients in tensor network form. The contribution of this article is to address this open problem by

- developing an algorithm that is guaranteed to result in the unique symmetric Volterra kernels of all orders,
- showing that the identification of low-rank symmetric Volterra kernels is an ill-conditioned problem as measurement noise completely destroys the low-rank property of the solution.

Numerical experiments demonstrate the effectiveness of the proposed algorithm and also illustrate the influence of measurement noise on the rank of the estimate. In the experiments it is also shown that the iterative identification algorithm described in Batselier et al. (2017a) requires a full-rank initialization in order to be able to find the low-rank symmetric solution. The reason for this behavior is not yet understood.

2. TENSOR BASICS

A tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$ is a D-dimensional array of numbers. Each entry of a tensor $\boldsymbol{\mathcal{A}}$ is determined by D indices $i_1, i_2, \ldots, i_D$. Commonly used tensors in control are scalars ($D = 0$), vectors ($D = 1$) and matrices ($D = 2$). A D-dimensional tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I \times I \times \cdots \times I}$ is per definition symmetric when the tensor entry $\boldsymbol{\mathcal{A}}(i_1, i_2, \ldots, i_D)$ remains invariant under any permutation of the indices $i_1, i_2, \ldots, i_D$. Following the convention used in Matlab, all indices are 1-based and hence start counting from 1 rather than from 0. Indices are always denoted by lowercase letters and their corresponding capital letters denote their respective upper bounds. For example, an entry of an $M \times N$ matrix $\boldsymbol{A}$ is denoted $\boldsymbol{A}(m, n)$. Indices can also be combined into a single multi-index. The conversion of D separate indices $i_1, i_2, \ldots, i_D$ into one single multi-index $[i_1 i_2 \cdots i_D]$ follows the definition

10.1016/j.ifacol.2021.08.404

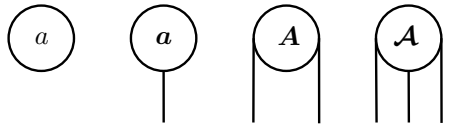

Fig. 1. Tensor diagrams of a scalar a, vector $\boldsymbol{a}$, matrix $\boldsymbol{A}$ and 3-way tensor $\boldsymbol{\mathcal{A}}$. Each edge represents a dimension of the tensor.

$$i := [i_1 i_2 \cdots i_D] = i_1 + \sum_{d=2}^{D} (i_d - 1) \prod_{l=1}^{d-1} I_l. \tag{1}$$

From (1) it follows that $I = I_1 I_2 \cdots I_D$. The vectorization $\text{vec}(\boldsymbol{\mathcal{A}})$ of a tensor $\boldsymbol{\mathcal{A}}$ is obtained by combining all indices of $\boldsymbol{\mathcal{A}}$ into one single index. Equation (1) also applies when a single index $i = [i_1 i_2 \cdots i_D]$ is split into D separate indices.

In this article, matrices with exponentially large dimensions are used. It is possible to lower the storage complexity of such matrices by representing them in a tensor train matrix form (Oseledets, 2010).

Definition 1. Let $\boldsymbol{A} \in \mathbb{R}^{I^D \times J^D}$ with entries

$$\boldsymbol{A}([i_1 i_2 \cdots i_D], [j_1 j_2 \cdots j_D]). \tag{2}$$

This implies that

$$I^D = I_1\, I_2 \cdots I_D,$$
$$J^D = J_1\, J_2 \cdots J_D.$$

The tensor train matrix of $\boldsymbol{A}$ consists of D 4-dimensional tensors $\boldsymbol{\mathcal{A}}^{(d)} \in \mathbb{R}^{R_d \times I_d \times J_d \times R_{d+1}}$ such that each matrix entry (2) is equal to

$$\sum_{r_2=1}^{R_2} \cdots \sum_{r_D=1}^{R_D} \boldsymbol{\mathcal{A}}^{(1)}(1, i_1, j_1, r_2) \boldsymbol{\mathcal{A}}^{(2)}(r_2, i_2, j_2, r_3) \cdots$$
$$\boldsymbol{\mathcal{A}}^{(D)}(r_D, i_D, j_D, 1). \tag{3}$$

Each tensor $\boldsymbol{\mathcal{A}}^{(d)}$ $(d = 1, \ldots, D)$ is called a core tensor. The dimensions $R_1, R_2, \ldots, R_{D+1}$ are called the tensor train matrix-ranks and per definition $R_1 = R_{D+1} = 1$. A more convenient representation of a tensor train matrix is via a tensor diagram. Each node in a tensor diagram represents a tensor and each each edge represents a particular index or dimension. Figure 1 shows the diagram of a scalar, vector, matrix and 3-dimensional tensor. Index summations as in (3) are represented in a tensor diagram as an edge that connects two nodes. The diagram representation of a tensor train matrix is shown in Figure 2. Note that the one-dimensional edges for $R_1 = R_{D+1}$ are not drawn in the diagram. Assuming uniform ranks $R_2 = \cdots = R_D = R$ and likewise for the dimensions I, J, then the storage cost of a tensor train matrix is $O(DIJR^2)$, compared to $O(I^D J^D)$ for the original matrix. The exponential dependency on the order D of the storage cost for the matrix is hence reduced to a linear dependency. The smaller the ranks R, the larger the gain in storage efficiency.

3. MIMO VOLTERRA SYSTEMS

In this section a brief overview of the MIMO Volterra identification problem as described in Batselier et al. (2017a) is given. A truncated Volterra model extends the linear finite impulse response (FIR) model

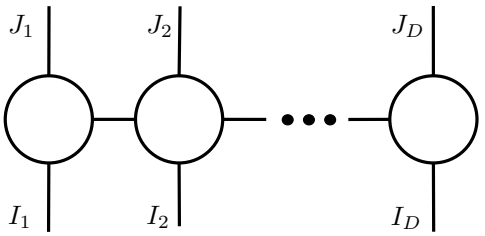

Fig. 2. Tensor diagram of a tensor train matrix that represents a matrix $\boldsymbol{A} \in \mathbb{R}^{I_1 I_2 \cdots I_D \times J_1 J_2 \cdots J_D}$. All row indices point downward and all column indices point upward.

$$y(n) = h_0 + \sum_{m_1=0}^{M} h_1(m_1)\, u(n - m_j) \tag{4}$$

to higher orders of nonlinearity by adding homogeneous polynomials in the lagged input $u(n)$ to (4). Suppose there are L outputs and P inputs, which implies that

$$\boldsymbol{y}(n) = (y_1(n)\ y_2(n)\ \cdots\ y_L(n))^T \in \mathbb{R}^L,$$
$$\boldsymbol{u}(n) = (u_1(n)\ u_2(n)\ \cdots\ u_P(n))^T \in \mathbb{R}^P.$$

Using the same notation as in Batselier et al. (2017a), for a given memory M the vector $\boldsymbol{u}_n$ is defined as

$$\boldsymbol{u}_n = \left(1\ \boldsymbol{u}(n)^T\ \cdots\ \boldsymbol{u}(n-M)^T\right)^T \in \mathbb{R}^{(1+(M+1)P)}.$$

Please not that the subscript n in $\boldsymbol{u}_n$ does not denote the nth entry of the vector $\boldsymbol{u}$. For notational convenience let $I = (1 + (M+1)P)$ and define the vector

$$\boldsymbol{u}_n^D := \overbrace{\boldsymbol{u}_n \otimes \boldsymbol{u}_n \otimes \cdots \otimes \boldsymbol{u}_n}^{D \text{ times}} \in \mathbb{R}^{I^D}. \tag{5}$$

where $\otimes$ denotes the matrix Kronecker product and D is the order of the Volterra system. The vector $\boldsymbol{u}_n^D$ contains all monomials in the lagged inputs from degree 0 up to D and can always be reshaped into a D-dimensional symmetric tensor $\boldsymbol{\mathcal{U}}_n^D$.

Example 1. For $D = 2, P = 1$ and $M = 1$ we have that

$$\boldsymbol{u}_n^2 = (1\ \boldsymbol{u}(n)\ \boldsymbol{u}(n-1))^T \otimes (1\ \boldsymbol{u}(n)\ \boldsymbol{u}(n-1))^T \in \mathbb{R}^9$$

contains 9 monomials. Since $D = 2$, we can reshape $\boldsymbol{u}_n^2$ into a 3×3 symmetric matrix

$$\begin{pmatrix} 1 & u_1(n) & u_1(n-1) \\ u_1(n) & u_1^2(n) & u_1(n)u_1(n-1) \\ u_1(n-1) & u_1(n)u_1(n-1) & u_1^2(n-1) \end{pmatrix}.$$

The output of a MIMO Volterra system can then be written as

$$\boldsymbol{y}(n)^T = \left(\boldsymbol{u}_n^D\right)^T \boldsymbol{H}\,, \tag{6}$$

where column l of the matrix $\boldsymbol{H} \in \mathbb{R}^{I^D \times L}$ contains all Volterra kernel coefficients from order 0 up to D responsible for output l. Writing out (6) for times $n = 1, \ldots, N$ results in a set of linear equations

$$\underbrace{\boldsymbol{Y}}_{N \times L} = \underbrace{\boldsymbol{U}}_{N \times I^D}\ \underbrace{\boldsymbol{H}}_{I^D \times L}. \tag{7}$$

The system identification problem is then: given a finite sequence of measured $\{(\boldsymbol{u}(n), \boldsymbol{y}(n)\}_{n=1}^N$, order D and memory M, solve (7) for the unknown $\boldsymbol{H}$ matrix.

Example 1 illustrates that each row of the $\boldsymbol{U}$ matrix contains repeated entries due to the symmetry. This symmetry limits the rank of the $\boldsymbol{U}$ matrix.

Theorem 1. (Lemma 4.1 (Batselier et al., 2017a, p. 30)) For the matrix $\boldsymbol{U}$ in (7) it holds that

$$\text{rank}(\boldsymbol{U}) \leq R = \binom{D+I-1}{I-1}.$$

The M input signals are then per definition persistently exciting of order D and memory M if the upper bound R of the rank is attained. A consequence of this rank-deficiency is that (7) has an infinite number of solutions. It was proven in Proposition 4.2 of (Batselier et al., 2017a, p. 30) that each column of the unique minimum-norm solution for $\boldsymbol{H}$ is the vectorization of a D-dimensional symmetric tensor. Enforcing symmetry to the Volterra kernels is therefore equivalent with finding the minimum-norm solution of (7).

4. PSEUDOINVERSE IN TENSOR TRAIN MATRIX FORM

One way of finding the minimum-norm solution is via the Moore-Penrose pseudoinverse. Assume $N \geq R$ and let

$$\begin{aligned}\boldsymbol{U} &= (\boldsymbol{Q}_1 \ \boldsymbol{Q}_2) \begin{pmatrix} \boldsymbol{S} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{V}_1^T \\ \boldsymbol{V}_2^T \end{pmatrix}, \\ &= \boldsymbol{Q}_1 \, \boldsymbol{S} \, \boldsymbol{V}_1^T. \end{aligned} \tag{8}$$

be the singular value decomposition (SVD) of $\boldsymbol{U}$, where $\boldsymbol{Q} \in \mathbb{R}^{N \times N}$, $\boldsymbol{V} \in \mathbb{R}^{I^D \times I^D}$ are orthogonal matrices and $\boldsymbol{S} \in \mathbb{R}^{R \times R}$ is a diagonal matrix. The symmetric Volterra kernel coefficients can then be retrieved as

$$\boldsymbol{H}_{\text{symm}} = \boldsymbol{V}_1 \, \boldsymbol{S}^{-1} \, \boldsymbol{Q}_1^T \, \boldsymbol{Y}. \tag{9}$$

4.1 Tensor train matrix identification

The exponential number of columns of $\boldsymbol{U}$ make the SVD computationally inefficient even for moderate values of I and D. Fortunately, this curse of dimensionality can be lifted by rewriting the linear problem in terms of tensor train matrices. Figure 4 shows the tensor diagram of (7) where both $\boldsymbol{U}$ and $\boldsymbol{H}$ are represented by tensor train matrices. These tensor train matrices are defined such that

$$\begin{aligned}\boldsymbol{\mathcal{U}}^{(1)} &\in \mathbb{R}^{1 \times N \times I \times R_2}, \\ \boldsymbol{\mathcal{H}}^{(1)} &\in \mathbb{R}^{1 \times I \times L \times R_2}.\end{aligned}$$

The second index of the remaining $\boldsymbol{\mathcal{U}}^{(d)}$ tensors is always one-dimensional, and likewise for the third index of the remaining $\boldsymbol{\mathcal{H}}^{(d)}$ tensors. Summing over all indices that correspond with connected edges in the diagram results in the $N \times L$ matrix $\boldsymbol{Y}$. The symmetry that is present in the $\boldsymbol{U}$ also results in upper bounds for the corresponding tensor train matrix-ranks. These upper bounds were proven for a slightly different matrix but with similar structure.

Theorem 2. (Theorem 4.1 (Batselier et al., 2018, p. 191)) The tensor train matrix-ranks of the matrix $\boldsymbol{U}$ in (7) satisfy

$$R_d \leq \binom{D-d+I-1}{I-1}. \tag{10}$$

Theorem 2 allows for an alternative definition of persistently exciting input signals as inputs for which these upper bounds are all attained. The tensor train matrix

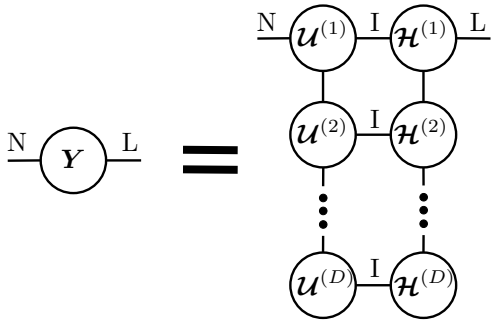

Fig. 3. Tensor diagram of (7).Both the matrices $\boldsymbol{U}$ and $\boldsymbol{H}$ are represented by tensor train matrices. Row indices point to the left and column indices point to the right.

for $\boldsymbol{U}$ can be constructed one core at a time. Defining the $N \times I$ matrix

$$\tilde{\boldsymbol{U}} = \begin{pmatrix} \boldsymbol{u}_1^T \\ \boldsymbol{u}_2^T \\ \vdots \\ \boldsymbol{u}_N^T \end{pmatrix},$$

then we have that

$$\boldsymbol{U} = \overbrace{\tilde{\boldsymbol{U}} \odot \tilde{\boldsymbol{U}} \odot \cdots \odot \tilde{\boldsymbol{U}}}^{D \text{ times}}, \tag{11}$$

where the $\odot$ operator take the row-wise Kronecker product of two matrices. Algorithm 1 is a modified version of Algorithm 2 in (Batselier et al., 2018, p. 191), where a matrix $\boldsymbol{U}$ with identical structure appears in the context of polynomial state space models. The assumption is made that the input signals are persistently exciting such that the tensor train matrix-ranks are given by (10). If the inputs are not persistently exciting, then a numerical rank needs to be determined. Once the tensor train matrix of

Algorithm 1 Construct tensor train matrix of repeated row-wise Kronecker product of matrices. (Algorithm 2 (Batselier et al., 2018, p. 191))

Input: $N \times I$ matrix $\tilde{\boldsymbol{U}}$, factor D
Output: tensor train matrix $\boldsymbol{\mathcal{U}}^{(1)}, \ldots, \boldsymbol{\mathcal{U}}^{(D)}$ of $\boldsymbol{U}$ in (11)

$\boldsymbol{\mathcal{U}}^{(D)} \leftarrow \text{reshape}(\tilde{\boldsymbol{U}}, [1, N, I, 1])$
for $d = D : -1 : 2$ **do**
  $\boldsymbol{T} \leftarrow \text{reshape}(\boldsymbol{\mathcal{U}}^{(d)}, [N, IR_{d+1}])$ % $R_{D+1} = 1$
  $\boldsymbol{T} \leftarrow \boldsymbol{T} \odot \tilde{\boldsymbol{U}}$
  $\boldsymbol{T} \leftarrow \text{reshape}(\boldsymbol{T}, [NI, IR_{d+1}])$
  $[\boldsymbol{Q}, \boldsymbol{S}, \boldsymbol{V}] \leftarrow \text{SVD}(\boldsymbol{T})$
  Truncate $\boldsymbol{Q}, \boldsymbol{S}, \boldsymbol{V}$ to a rank $R_d = \binom{d-1+I-1}{I-1}$
  $\boldsymbol{\mathcal{U}}^{(d)} \leftarrow \text{reshape}(\boldsymbol{V}^T, [R_d, 1, I, R_{d+1}])$
  $\boldsymbol{\mathcal{U}}^{(d-1)} \leftarrow \text{reshape}(\boldsymbol{U}\boldsymbol{S}, [1, N, I, R_d])$
end for

$\boldsymbol{U}$ is obtained through Algorithm 1, then its pseudoinverse can also be readily computed. The thin SVD as in (8) is obtained through one SVD computation of the first tensor train matrix core tensor. This core tensor $\boldsymbol{\mathcal{U}}^{(1)}$ is first reshaped into an $N \times IR_2$ matrix $\boldsymbol{U}_1$ and then its SVD is computed. If the input signals are persistently exciting, then the rank of $\boldsymbol{U}_1$ is guaranteed to be $R = \binom{D+I-1}{I-1}$. The first core of the tensor train matrix of $\boldsymbol{V}_1$ is obtained through reshaping the orthogonal $\boldsymbol{V}$ matrix

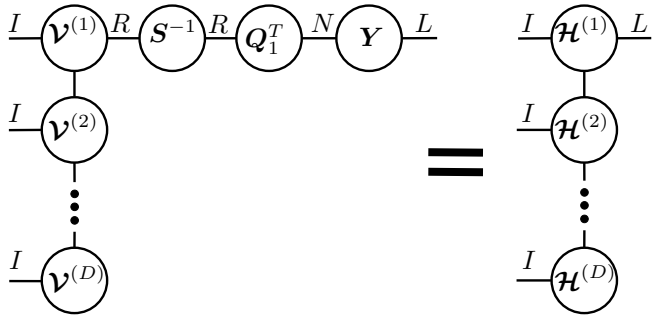

Fig. 4. Tensor diagram of (9). The pseudoinverse of $\boldsymbol{U}$ is computed in tensor train matrix form and applied to the matrix $\boldsymbol{Y}$, resulting in a tensor train matrix for $\boldsymbol{H}$. Row indices point to the left and column indices point to the right.

from the SVD and retaining all other tensor train matrix cores from $\boldsymbol{U}$. The thin SVD computation from (8) is summarized in pseudocode in Algorithm 2. The symmetric minimum-norm solution $\boldsymbol{H}_{\text{symm}}$ is obtained through the diagram shown in Figure 4. The matrices $\boldsymbol{S}^{-1}, \boldsymbol{Q}_1^T$ and $\boldsymbol{Y}$ are all "absorbed" by $\boldsymbol{\mathcal{V}}^{(1)}$. The difference between

Algorithm 2 Thin SVD of $\boldsymbol{U}$ in tensor train matrix form

Input: tensor train matrix of $\boldsymbol{U}$ from Algorithm 1
Output:matrices $\boldsymbol{Q}_1, \boldsymbol{S}$ and tensor train matrix $\boldsymbol{V}_1$ in (8)

1: $\boldsymbol{U}_1 \leftarrow \text{reshape}(\boldsymbol{\mathcal{U}}^{(1)}, [N, IR_2])$
2: $[\boldsymbol{Q}_1, \boldsymbol{S}, \boldsymbol{V}] \leftarrow \text{SVD}(\boldsymbol{U}_1)$
3: Truncate $\boldsymbol{Q}_1, \boldsymbol{S}, \boldsymbol{V}$ to a rank $R = \binom{D+I-1}{I-1}$
4: $\boldsymbol{\mathcal{V}}^{(1)} \leftarrow \text{reshape}(\boldsymbol{V}, [1, R, I, R_2])$
5: $\boldsymbol{\mathcal{V}}^{(d)} \leftarrow \boldsymbol{\mathcal{U}}^{(d)}$ $(d = 2, \ldots, D)$

the tensor train matrix for $\boldsymbol{H}_{\text{symm}}$ and $\boldsymbol{V}_1$ therefore lies only in the first core. The resulting tensor train matrix-ranks of $\boldsymbol{H}_{\text{symm}}$ will therefore be identical to the ranks of $\boldsymbol{V}_1$. However, the column dimension has been reduced from R to L, which implies that lower ranks are likely to exist. The ranks can be truncated to smaller values, without loss of accuracy, through a a sequence of SVD computations (Oseledets, 2011, p. 2305). Upper bounds for the resulting ranks can be deduced by taking the symmetry of $\boldsymbol{H}_{\text{symm}}$ into account.

Theorem 3. The tensor train matrix representation of the matrix $\boldsymbol{H}_{\text{symm}}$ has tensor train matrix-ranks R_d that satisfy

$$R_d \leq \min\left(L\binom{d+I-1}{I-1}, \binom{D-d+I-1}{I-1}\right). \quad (12)$$

Proof. Define the matrix $\boldsymbol{H}_d \in \mathbb{R}^{LI^{d-1} \times I^{D-d+1}}$ such that

$$\boldsymbol{H}_d([li_1 \cdots i_{d-1}], [i_d i_{d+1} \cdots i_D]) = \boldsymbol{H}_{\text{symm}}([i_1 i_2 \cdots i_D], l).$$

The tensor train matrix-rank R_d is then per definition

$$\text{rank}\,(\boldsymbol{H}_d)\,.$$

The symmetry of $\boldsymbol{H}_{\text{symm}}$ implies that any entry remains invariant under any permutation of the $i_1, \ldots, i_D$ indices. As a consequence, $\boldsymbol{H}_d$ has at most $L\binom{d+I-1}{I-1}$ linearly independent rows and at most $\binom{D-d+I-1}{I-1}$ linearly independent columns.

Volterra kernels that are decaying and/or sparse will lead to ranks that are smaller than the upper bounds of (12).

4.2 Influence of measurement noise

Noise on the output measurements will affect the identified model in the sense that the solution of the perturbed system

$$\boldsymbol{Y} + \boldsymbol{E} = \boldsymbol{U}\,\boldsymbol{H} \quad (13)$$

is found. The matrix $\boldsymbol{E}$ contains the measurement noise and does not affect the $\boldsymbol{U}$ matrix. The minimum-norm solution of (13)

$$\hat{\boldsymbol{H}}_{\text{symm}} = \boldsymbol{H}_{\text{symm}} + \boldsymbol{V}_1\,\boldsymbol{S}^{-1}\,\boldsymbol{Q}_1^T\,\boldsymbol{E},$$

therefore consists of $\boldsymbol{H}_{\text{symm}}$ upon which a symmetric perturbation $\boldsymbol{V}_1\,\boldsymbol{S}^{-1}\,\boldsymbol{Q}_1^T\,\boldsymbol{E}$ is applied. If we assume that $\boldsymbol{E}$ consists of mutually uncorrelated noise samples, then it follows from Theorem 3 that all tensor train matrix-ranks of this symmetric perturbation term will attain their upper bounds. As a result, the tensor train matrix-ranks of $\hat{\boldsymbol{H}}_{\text{symm}}$ will also attain these upper bounds, thus destroying any inherent low-rank structure in $\boldsymbol{H}_{\text{symm}}$. Low-rank symmetric Volterra kernel identification is therefore an ill-conditioned problem, as any level of measurement noise completely destroys the low-rank property.

5. ALTERNATING LINEAR SCHEME IDENTIFICATION

An alternative method for solving (7) is the alternating linear scheme (ALS, also called alternating least squares), which is fully described in Batselier et al. (2017a). The main idea of the ALS algorithm is to initialize a tensor train matrix for $\boldsymbol{H}$ with desired tensor train matrix-ranks and then update each core tensor $\boldsymbol{\mathcal{H}}^{(d)}$ in an iterative manner while keeping the other cores fixed. The tensor train matrix-ranks for each core need to be chosen in advance and in order to ensure symmetry Theorem 3 must be kept in mind. Defining

$$\boldsymbol{U}_{<d} = (\boldsymbol{\mathcal{H}}^{(1)} \times_2 \boldsymbol{u}_n) \cdots (\boldsymbol{\mathcal{H}}^{(d-1)} \times_2 \boldsymbol{u}_n) \in \mathbb{R}^{L \times R_d},$$
$$\boldsymbol{u}_{>d} = (\boldsymbol{\mathcal{H}}^{(d+1)} \times_2 \boldsymbol{u}_n) \cdots (\boldsymbol{\mathcal{H}}^{(D)} \times_2 \boldsymbol{u}_n) \in \mathbb{R}^{R_{d+1} \times 1},$$

where $\boldsymbol{\mathcal{H}} \times_2$ denotes the summation over the second index of $\boldsymbol{\mathcal{H}}$, then the d-th tensor train matrix core $\boldsymbol{\mathcal{H}}^{(d)}$ can be updated by solving the following linear system

$$\begin{pmatrix} \boldsymbol{y}(1) \\ \boldsymbol{y}(2) \\ \vdots \\ \boldsymbol{y}(N) \end{pmatrix} = \begin{pmatrix} \boldsymbol{u}_{>d}^T \otimes \boldsymbol{u}_1^T \otimes \boldsymbol{U}_{<d} \\ \boldsymbol{u}_{>d}^T \otimes \boldsymbol{u}_2^T \otimes \boldsymbol{U}_{<d} \\ \vdots \\ \boldsymbol{u}_{>d}^T \otimes \boldsymbol{u}_N^T \otimes \boldsymbol{U}_{<d} \end{pmatrix} \text{vec}(\boldsymbol{\mathcal{H}}^{(d)}). \quad (14)$$

The matrix in (14) has dimensions $N \times R_d I R_{d+1}$, which means that the ranks R_d, R_{d+1} should be chosen such that $R_d\, I\, R_{d+1} \leq N$. The rank-deficiency of (6) also results in a rank-deficiency at cores 2 and $D-1$. In these cases the minimum-norm solution of (14) can be computed through the pseudoinverse.

6. EXPERIMENTS

In this section we demonstrate both Algorithm 2 and the ALS algorithm of Batselier et al. (2017a) in order to retrieve low-rank symmetric Volterra kernels. The effect

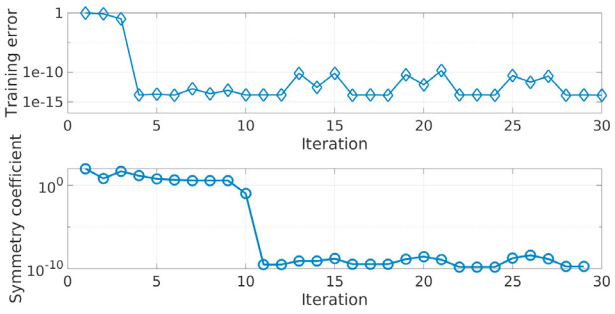

Fig. 5. ALS identification of a low-rank Volterra system without measurement noise. Both the relative training error and symmetry coefficient are shown for 30 iterations. The relative training error becomes numerically zero at iteration 4, while the symmetric solution is found at iteration 11.

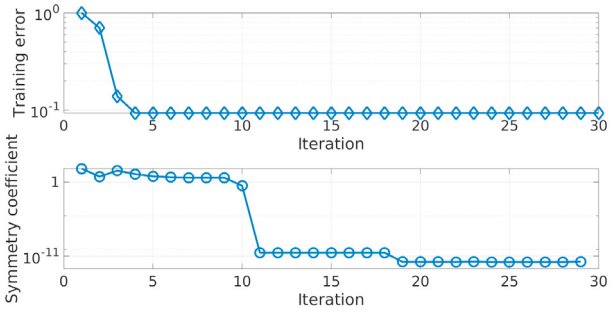

Fig. 6. ALS identification of a low-rank Volterra system with measurement noise. Both the relative training error and symmetry coefficient are shown for 30 iterations. The relative training error stabilizes at iteration 4, while the symmetric solution is found at iteration 11.

of measurement noise to the results is also demonstrated. All experiments were run on a laptop running an Intel i7 processor at 1.90 GHz and 8 GB RAM. A Matlab implementation to reproduce these experiments can be freely downloaded from `https://github.com/kbatseli/SymmetricVolterra`.

6.1 Low-rank Volterra system

For both experiments a single-input-single-output system with low-rank symmetric Volterra kernels was generated. The order and memory are set to $D = 7$ and $M = 3$, respectively such that $\boldsymbol{H}$ contains $4^7 = 16384$ kernel coefficients of which only $R = \binom{7+4-1}{4-1} = 120$ are unique. The kernel coefficients were constructed by first initializing a 4×100 matrix $\boldsymbol{A}$ where each column i consists of a sampled decaying exponential function

$$\boldsymbol{A}(:, i) = |\alpha_i| \, \exp(-\beta_i \, [1 : I])$$

where α_i is sampled from a standard normal distribution and β_i is an integer sampled uniformly from the range $[1, 10]$. Taking $D - 1 = 6$ column-wise Kronecker products of $\boldsymbol{A}$ with itself and summing over the columns then results in the desired $\boldsymbol{H}$ vector. The resulting tensor train matrix-ranks of $\boldsymbol{H}$ were uniformly 3 with a relative approximation error of 1.2×10^{-13}.

The input signal was chosen to be standard normal Gaussian white noise and is therefore persistently exciting of any order. The measurement noise is also Gaussian white noise with a variance chosen such that a signal-to-noise ratio (SNR) of 20 dB was obtained. A total of 1300 samples were generated for which the first 1000 samples were used for identification and the remaining 300 samples for validation. The relative training error is defined as

$$\frac{||\boldsymbol{Y} - \boldsymbol{U}\boldsymbol{H}||_F}{||\boldsymbol{Y}||_F}$$

where $||\cdot||_F$ denotes the Frobenius norm.

6.2 No measurement noise

Algorithms 1 and 2 required 30.4 seconds to compute the minimum-norm solution. The resulting tensor train matrix-ranks are equal to the upper bounds in (10). Truncating these ranks via a sequence of SVD computations such that the relative error is smaller than 10^{-12} result in tensor train matrix-ranks that are uniformly 3. The relative error between the computed kernel coefficients $\boldsymbol{H}_{\text{pseudo}}$ and the ground truth $\boldsymbol{H}$ is then

$$\frac{||\boldsymbol{H}_{\text{pseudo}} - \boldsymbol{H}||_2}{||\boldsymbol{H}||_2} = 2.75 \times 10^{-13}.$$

The symmetry of the obtained solution can be verified by computing the symmetry coefficient, which is defined as

$$s = \sum_p ||\boldsymbol{\mathcal{H}} - \text{permute}(\boldsymbol{\mathcal{H}}, \boldsymbol{p})||_F,$$

where the tensor $\boldsymbol{\mathcal{H}}$ is obtained by reshaping $\boldsymbol{H}$ into a D-dimensional tensor. The symmetry coefficient s compares $\boldsymbol{\mathcal{H}}$ to all possible index permutations $\boldsymbol{p}$ and is therefore exactly zero when $\boldsymbol{\mathcal{H}}$ is symmetric. The symmetry coefficient for the result $\boldsymbol{H}_{\text{pseudo}}$ from Algorithms 1 and 2 is $s = 6.15 \times 10^{-13}$.

Unexpectedly, the ALS algorithm is not able to retrieve the desired symmetric low-rank solution when the ranks are chosen to be uniformly 3. After 100 iterations the relative training error is 3.5×10^{-3}, the symmetry coefficient equals 8.8 and does not decrease anymore with further iterations. If the ALS algorithm is applied with ranks equal to the upper bounds (12) then the desired solution is found after 11 iterations. Both the relative training error and symmetry coefficient for 30 iterations are shown in Figure 5. Each iteration took 0.34 seconds. The relative training error is seen to drop to numerical zero ($\approx 10^{-15}$) after 4 iterations. The obtained solution is however not symmetric as the symmetry coefficient at that point is $s = 19.1$. At iteration 11 the symmetry coefficient also suddenly drops down to numerical zero and at that point the desired symmetric minimum-norm solution is obtained. All ranks of the resulting tensor train matrix can be truncated to 3 with a relative approximation error of 10^{-11}, which shows that the desired symmetric solution has been found. The reason why the ALS is not able to retrieve the true low-rank symmetric solution when initialized with the correct ranks is not understood and requires future research.

6.3 With measurement noise

As expected, the measurement noise destroys the inherent low-rank structure of $\boldsymbol{H}_{\text{symm}}$. Algorithms 1 and 2 retrieve

a tensor train matrix with ranks that are equal to their upper bounds (12). Further truncation of the ranks is not possible without destroying the symmetry of the solution. The symmetry coefficient is $s = 9.5\times10^{-13}$ and the relative validation error is 0.0432. A similar observation is made when using the ALS algorithm, as shown in Figure 6. The measurement noise prevents the training error to drop below a value of 10^{-1}, while the symmetric solution is found again after 11 iterations. The ranks, once again, need to be set to their upper bounds.

7. CONCLUSIONS

The proposed algorithm was shown to being able to identify the exact symmetric Volterra kernels. The exact reason why the iterative ALS algorithm requires a full-rank initialization to retrieve the true underlying low-rank solution is not understood and requires further research. Another interesting avenue of research is to find a low-rank symmetric approximation of $\hat{\boldsymbol{H}}_{\text{symm}}$ in (13), rather than $\hat{\boldsymbol{H}}_{\text{symm}}$ itself. In addition, alternative tensor decompositions could be considered, such as the canonical polyadic decomposition as used in Boussé et al. (2018). An alternative structure that can be imposed on the Volterra kernels is the triangular structure Rugh (1981). Whether this structure can also be imposed in tensor network form remains to be investigated.

REFERENCES

Batselier, K., Chen, Z.M., and Wong, N. (2017a). Tensor Network alternating linear scheme for MIMO Volterra system identification. *Automatica*, 84, 26–35.

Batselier, K., Chen, Z.M., and Wong, N. (2017b). A Tensor Network Kalman filter with an application in recursive MIMO Volterra system identification. *Automatica*, 84, 17–25.

Batselier, K., Ko, C.Y., and Wong, N. (2018). Tensor network subspace identification of polynomial state space models. *Automatica*, 95, 187–196.

Birpoutsoukis, G., Csurcsia, P.Z., and Schoukens, J. (2018). Efficient multidimensional regularization for Volterra series estimation. *Mechanical Systems and Signal Processing*, 104, 896–914.

Birpoutsoukis, G., Marconato, A., Lataire, J., and Schoukens, J. (2017). Regularized nonparametric Volterra kernel estimation. *Automatica*, 82, 324–327.

Boussé, M., Vervliet, N., Domanov, I., Debals, O., and De Lathauwer, L. (2018). Linear systems with a canonical polyadic decomposition constrained solution: Algorithms and applications. *Numerical Linear Algebra with Applications*, 25(6), e2190.

Campello, R.J.G.B., Favier, G., and do Amaral, W.C. (2004). Optimal expansions of discrete-time Volterra models using Laguerre functions. *Automatica*, 40(5), 815 – 822.

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

de Silva, V. and Lim, L.H. (2008). Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM J. on Matrix Anal. Appl.*, 30(3), 1084–1127.

Diouf, C., Telescu, M., Cloastre, P., and Tanguy, N. (2012). On the Use of Equality Constraints in the Identification of Volterra-Laguerre Models. *IEEE Signal Processing Letters*, 19(12), 857–860.

Favier, G., Kibangou, A.Y., and Bouilloc, T. (2012). Nonlinear system modeling and identification using Volterra-PARAFAC models. *Int. J. Adapt. Control Signal Process*, 26(1), 30–53.

Harshman, R.A. (1970). Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16(1), 84.

Håstad, J. (1990). Tensor rank is NP-complete. *Journal of Algorithms*, 11(4), 644–654.

Oseledets, I.V. (2011). Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5), 2295–2317.

Oseledets, I. (2010). Approximation of $2^d \times 2^d$ Matrices Using Tensor Decomposition. *SIAM J. Matrix Anal. Appl.*, 31(4), 2130–2145.

Rugh, W. (1981). *Nonlinear System Theory – The Volterra-Wiener Approach*. Baltimore, MD: Johns Hopkins Univ. Press.

Shi, T. and Townsend, A. (2021). On the compressibility of tensors. *SIAM Journal on Matrix Analysis and Applications*, 42(1), 275–298.

Tucker, L.R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3), 279–311.