Extended Kalman Filtering with Low-Rank Tensor Networks for MIMO Volterra System Identification

Batselier, Kim; Ko, Ching Yun; Wong, Ngai

DOI
10.1109/CDC40024.2019.9028895

Publication date
2019

Document Version
Final published version

Published in
Proceedings of the IEEE 58th Conference on Decision and Control, CDC 2019

2019 IEEE 58th Conference on Decision and Control (CDC)
Palais des Congrès et des Expositions Nice Acropolis
Nice, France, December 11-13, 2019

Extended Kalman Filtering with Low-Rank Tensor Networks for MIMO Volterra System Identification

Kim Batselier, Ching-Yun Ko and Ngai Wong

***Abstract*— This article reformulates the multiple-input-multiple-output Volterra system identification problem as an extended Kalman filtering problem. This reformulation has two advantages. First, it results in a simplification of the solution compared to the Tensor Network Kalman filter as no tensor filtering equations are required anymore. The second advantage is that the reformulation allows to model correlations between the parameters of different multiple-input-single-output Volterra systems, which can lead to better accuracy. The curse of dimensionality in the exponentially large parameter vector and covariance matrix is lifted through the use of low-rank tensor networks. The computational complexity of our tensor network implementation is compared to the conventional implementation and numerical experiments demonstrate the effectiveness of the proposed method.**

I. Introduction

Tensor decompositions were shown in [1]–[6] to be a useful paradigm to lift the curse of dimensionality in the identification of nonlinear Volterra models. The need to estimate an exponential number of model parameters is circumvented by representing these parameters as a particular tensor decomposition. Furthermore, the low-rank assumption in these tensor decompositions induces an implicit regularization on the model complexity. In both [5] and [6], the identification of a discrete-time multiple-input-multiple-output (MIMO) Volterra system was described and solved as a filtering problem. Indeed, a time-varying discrete-time p-input-l-output Volterra system of degree d and memory M is described by the following state space model

$$\begin{aligned} \boldsymbol{X}(t+1) &= \boldsymbol{A}(t)\ \boldsymbol{X}(t) + \boldsymbol{W}(t), \\ \boldsymbol{y}(t) &= \boldsymbol{u}_t{}^{\circledast}\ \boldsymbol{X}(t) + \boldsymbol{r}(t), \end{aligned} \tag{1}$$

where $\boldsymbol{X}(t) \in \mathbb{R}^{N^d \times l}$ is an exponentially large matrix containing the Volterra parameters of each of the l outputs, $\boldsymbol{y}(t) \in \mathbb{R}^{1\times l}$ is a **row vector** of l scalar measurements, $\boldsymbol{A}(t) \in \mathbb{R}^{N^d \times N^d}$ is a transition matrix that partly determines the time-varying character of the Volterra parameters and $\boldsymbol{W}(t) \in \mathbb{R}^{N^d \times l}$, $\boldsymbol{r}(t) \in \mathbb{R}^{1\times l}$ denote zero-mean independent Gaussian process and measurement noise, respectively. The row vector

$$\boldsymbol{u}_t := \begin{pmatrix} 1 & \boldsymbol{u}^T(t) & \cdots & \boldsymbol{u}^T(t-M+1) \end{pmatrix} \in \mathbb{R}^{1\times(pM+1)}$$

contains all p input values at times t down to $t-M+1$ and $\boldsymbol{u}_t{}^{\circledast}$ is defined as the d-times repeated Kronecker product

$$\boldsymbol{u}_t{}^{\circledast} := \overbrace{\boldsymbol{u}_t \otimes \boldsymbol{u}_t \otimes \cdots \otimes \boldsymbol{u}_t}^{d} \in \mathbb{R}^{1\times(pM+1)^d}. \tag{2}$$

In the remainder of the text we will use $N := (pM+1)$ for notational convenience. Unlike the conventional Kalman filter with a single state vector, (1) represents a more general setting by concatenating l state vectors into a matrix $\boldsymbol{X}(t)$ and l scalar outputs into a row vector $\boldsymbol{y}(t)$. These l state space models are "coupled" by a common state transition matrix $\boldsymbol{A}(t)$ and time-varying output model vector $\boldsymbol{u}_t{}^{\circledast}$. Estimating the states $\boldsymbol{X}(t)$ through a Kalman filter then requires rewriting the Kalman filter equations into tensor equations [5], [6]. The assumption is hereby made that all estimated Volterra kernels over the l outputs are uncorrelated. These tensor equations were implemented using low-rank tensor networks in order to lift the curse of dimensionality associated with the state vectors in $\boldsymbol{X}(t)$ having exponentially large dimensions.

This article represents an alternative representation of the discrete-time MIMO Volterra system. The key contribution is that all unknown Volterra kernels are described by one single unknown vector $\boldsymbol{\theta}(t) \in \mathbb{R}^{lN^d}$ in the following state space model

$$\begin{aligned} \boldsymbol{\theta}(t+1) &= \boldsymbol{\theta}(t) + \boldsymbol{w}(t), \\ \boldsymbol{y}(t) &= h\left(\boldsymbol{u}_t\,, \boldsymbol{\theta}(t)\right) + \boldsymbol{e}(t), \end{aligned} \tag{3}$$

where $\boldsymbol{y}(t)$ is a column vector and the smooth vector function $h : \mathbb{R}^N \times \mathbb{R}^{lN^d} \to \mathbb{R}^l$ maps all time-delayed input signals in $\boldsymbol{u}_t$ to the output vector $\boldsymbol{y}(t)$. Both the process noise $\boldsymbol{w}(t)$ and measurement noise $\boldsymbol{e}(t)$ are assumed to be stationary Gaussian white noise processes

$$\mathbb{E}\left(\begin{pmatrix}\boldsymbol{w}(t)\\ \boldsymbol{e}(t)\end{pmatrix}\left(\boldsymbol{w}(s)^T\ \boldsymbol{e}(s)^T\right)\right) = \begin{pmatrix}\boldsymbol{R}_w & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{R}_e\end{pmatrix}\delta_{ts}.$$

Furthermore, we assume that $p, l \ll N^d$. The concatenation of all l MISO Volterra system parameters into one vector $\boldsymbol{\theta}(t)$ implies that one covariance matrix suffices, which opens up the possibility of modeling correlations between the parameters of different MISO Volterra systems. An additional benefit of this reformulation is that in contrast to the Kalman filter solutions in [5], [6] no tensor equations are required to formulate the Kalman filter, simplifying their implementation significantly. In this article an extended Kalman filter (EKF) will be developed to estimate the state vector $\boldsymbol{\theta}(t)$ from (3). A low-rank tensor network representation for both the estimate

K. Batselier is with the Delft Center for Systems and Control, Delft University of Technology, The Netherlands k.batselier@tudelft.nl

C.-Y Ko is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02142, USA

N. Wong is with the Department of Electrical and Electronic Engineering at The University of Hong Kong, Hong Kong, China

of $\boldsymbol{\theta}(t)$ and its covariance matrix $\boldsymbol{P}$ will be employed in our implementation to lift the curse of dimensionality.

II. Notation and Tensor network preliminaries

Tensors in this article are multi-dimensional arrays that generalize the notions of vectors and matrices to higher orders. A d-way or dth-order tensor is denoted $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_d}$ and hence each of its entries $\boldsymbol{\mathcal{A}}(n_1, n_2, \ldots, n_d)$ is determined by d indices. We use the MATLAB array index and colon notation to denote entries of tensors. The numbers $N_1, N_2, \ldots, N_d$ are called the dimensions of the tensor. For practical purposes, only real tensors are considered. We use boldface capital calligraphic letters $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \ldots$ to denote tensors, boldface capital letters $\boldsymbol{A}, \boldsymbol{B}, \ldots$ to denote matrices, boldface letters $\boldsymbol{a}, \boldsymbol{b}, \ldots$ to denote vectors, and Roman letters $a, b, \ldots$ to denote scalars. The transpose of a matrix $\boldsymbol{A}$ or vector $\boldsymbol{a}$ are denoted by $\boldsymbol{A}^T$ and $\boldsymbol{a}^T$, respectively. The unit matrix of order n is denoted $\boldsymbol{I}_n$ and $\boldsymbol{1}_n$ denotes the vector of ones of length n. The Kronecker product will play an important role in the construction of tensor networks and is denoted by $\otimes$. Before we can give the definitions of the tensor network structures used in this article, we first need to define how a single index can be expressed in terms of multiple indices. In general, the conversion of a given set of indices $n_1, n_2, \ldots, n_d$ with $1 \leq n_k \leq N_k \, (k = 1, \ldots, d)$ into a corresponding single index $1 \leq [n_1 n_2 \cdots n_d] \leq \prod_{k=1}^{d} N_k$ is defined by

$$[n_1 n_2 \cdots n_d] := n_1 + \sum_{k=2}^{d} (n_k - 1) \prod_{l=1}^{k-1} N_l. \tag{4}$$

Equation (4) allows us to rewrite the entries $a(n)$ of a vector $\boldsymbol{a} \in \mathbb{R}^{N^d}$ with $1 \leq n \leq N^d$ as $a([n_1 n_2 \cdots n_d])$ with $1 \leq n_k \leq N$ for all $k = 1, \ldots, d$. This rewriting enables us to use tensor networks for representing vectors and matrices of exponential size in an efficient manner. In this article, vectors will be represented by Matrix Product States (MPS, also called Tensor Trains [7]) and matrices by Matrix Product Operators (MPO, also called Tensor Train Matrix [8]). Without loss of generality, we provide the definition of an MPS for a vector of length N^d.

Definition 2.1: ([7, p. 2296]) For a vector $\boldsymbol{a} \in \mathbb{R}^{N^d}$ we define its MPS as the set of 3-way tensors $\boldsymbol{\mathcal{A}}^{(k)} \in \mathbb{R}^{R_k \times N \times R_{k+1}}$ for all $k = 1, \ldots, d$ such that the entry $\boldsymbol{a}([n_1 n_2 \cdots n_d])$ equals

$$\sum_{r_2, \ldots, r_d = 1}^{R_2, \ldots, R_d} \boldsymbol{\mathcal{A}}^{(1)}(1, n_1, r_2) \, \boldsymbol{\mathcal{A}}^{(2)}(r_2, n_2, r_3) \cdots \boldsymbol{\mathcal{A}}^{(d)}(r_d, n_d, 1).$$

The dimensions $R_2, R_3, \ldots, R_d$ are called the MPS-ranks. We have per definition that $R_1 = R_{d+1} = 1$.

Storage of the MPS requires the storage of each of the 3-way tensors, which leads to a storage complexity of approximately dNR^2 assuming all MPS-ranks are equal to R. In this way the curse of dimensionality associated with the storage of a vector of exponential length is lifted for small MPS-ranks R. A visual representation of an MPS is shown in Figure 1. Each node in the figure represents one of the tensors

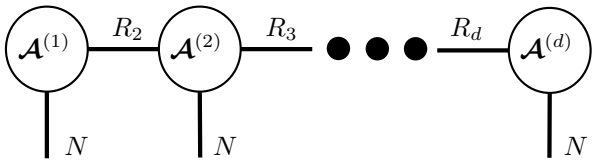

Fig. 1. Graphical representation of the MPS that represents a vector $\boldsymbol{a} \in \mathbb{R}^{N^d}$ with all dimensions labelled.

$\boldsymbol{\mathcal{A}}^{(k)} \, (k = 1, \ldots, d)$ in the MPS and the summations over the indices $r_2, \ldots, r_d$ are visualized by connecting edges between the nodes. The idea of an MPS can be extended to represent matrices. Analogous to Definition 2.1, the MPO of a matrix $\boldsymbol{A} \in \mathbb{R}^{N^d \times M^d}$ is defined as the set of 4-way tensors $\boldsymbol{\mathcal{A}}^{(k)} \in \mathbb{R}^{R_k \times N \times M \times R_{k+1}}$ for all $k = 1, \ldots, d$ such that the matrix entry $\boldsymbol{A}([n_1 n_2 \cdots n_d], [m_1 m_2 \cdots m_d])$ is represented by a summation of the MPO-tensors $\boldsymbol{\mathcal{A}}^{(k)}(r_k, n_k, m_k, r_{k+1})$ over the indices $r_1, \ldots, r_d$. Again, we have that per definition $R_1 = R_{d+1} = 1$. The following important theorem relates how a Kronecker product of matrices can be represented by a MPO with all unit MPO-ranks.

Theorem 2.1: ([9, p. 1225]) A matrix $\boldsymbol{A} \in \mathbb{R}^{N^d \times M^d}$ that satisfies

$$\boldsymbol{A} = \boldsymbol{A}^{(d)} \otimes \cdots \otimes \boldsymbol{A}^{(2)} \otimes \boldsymbol{A}^{(1)}$$

has an MPO representation where the kth MPO-tensor is $\boldsymbol{A}^{(k)} \in \mathbb{R}^{1 \times N \times M \times 1} \, (k = 1, \ldots, d)$ with unit MPO-ranks.

It is important to note that the order of the MPO-tensors is reversed with respect to the order of the factor matrices in the Kronecker product. This means that the last factor matrix $\boldsymbol{A}^{(1)}$ in the Kronecker product of Theorem 2.1 is the first tensor in the corresponding MPO representation.

III. Extended Kalman filter equations

The main idea of an EKF is to approximate the filtering density $p(\boldsymbol{\theta}(t) | \boldsymbol{y}_{1:t})$ with a Gaussian, characterized by a mean vector $\boldsymbol{m}(t) \in \mathbb{R}^{lN^d}$ and covariance matrix $\boldsymbol{P}(t) \in \mathbb{R}^{lN^d \times lN^d}$ [10]. The prediction step of an EKF for the state space model (3) is

$$\begin{aligned} \boldsymbol{m}(t)^- &= \boldsymbol{m}(t-1), \\ \boldsymbol{P}(t)^- &= \boldsymbol{P}(t-1) + \boldsymbol{R}_w. \end{aligned} \tag{5}$$

The corresponding update step is

$$\begin{aligned} \boldsymbol{v}(t) &= \boldsymbol{y}(t) - h\left(\boldsymbol{u}_t \, , \, \boldsymbol{m}(t)^-\right), \\ \boldsymbol{S}(t) &= \boldsymbol{C} \, \boldsymbol{P}(t)^- \, \boldsymbol{C}^T + \boldsymbol{R}_e, \\ \boldsymbol{K}(t) &= \boldsymbol{P}(t)^- \, \boldsymbol{C}^T \, \boldsymbol{S}(t)^{-1}, \\ \boldsymbol{m}(t) &= \boldsymbol{m}(t)^- + \boldsymbol{K}(t) \, \boldsymbol{v}(t), \\ \boldsymbol{P}(t) &= \boldsymbol{P}(t)^- - \boldsymbol{K}(t) \, \boldsymbol{S}(t) \, \boldsymbol{K}(T)^T, \end{aligned} \tag{6}$$

with

$$\boldsymbol{C} := \left. \frac{dh\left(\boldsymbol{u}_t \, , \, \boldsymbol{\theta}(t)\right)}{d\boldsymbol{\theta}(t)} \right|_{\boldsymbol{\theta}(t) = \boldsymbol{m}(t)^-} \in \mathbb{R}^{l \times lN^d}. \tag{7}$$

The exponential dimensions of $\boldsymbol{m}(t)$, $\boldsymbol{P}(t)$, $\boldsymbol{K}(t)$ and $\boldsymbol{C}$ will be resolved in the next section through the use of low-rank tensor network representations. Before proving the particular structure of the $\boldsymbol{C}$ matrix in (7) for the MIMO Volterra model, we first define the structure of the $\boldsymbol{\theta}(t)$ vector.

Definition 3.1: Let $\boldsymbol{\theta}_k(t) \in \mathbb{R}^{N^d}$ denote the vector containing the Volterra parameters such that

$$y_k(t) = \boldsymbol{u}_t{}^{\circledast} \, \boldsymbol{\theta}_k(t) \quad (1 \leq k \leq l),$$

where $y_k(t)$ denotes the kth output. Then we define

$$\boldsymbol{\theta}(t) := \begin{pmatrix} \boldsymbol{\theta}_1(t) \\ \boldsymbol{\theta}_2(t) \\ \vdots \\ \boldsymbol{\theta}_l(t) \end{pmatrix} \in \mathbb{R}^{lN^d}. \tag{8}$$

With the definition of the $\boldsymbol{\theta}(t)$ vector in place, we can now prove the particular structure of the $\boldsymbol{C}$ matrix (7) in the EKF.

Theorem 3.1: For the time-varying MIMO Volterra state space model (3) and $\boldsymbol{\theta}(t)$ vector (8) we have that

$$\boldsymbol{C} := \left. \frac{dh\left(\boldsymbol{u}_t\,,\,\boldsymbol{\theta}(t)\right)}{d\boldsymbol{\theta}(t)} \right|_{\boldsymbol{\theta}(t)=\boldsymbol{m}(t)^-} = \left(\boldsymbol{I}_l \otimes \boldsymbol{u}_t{}^{\circledast}\right).$$

Proof: From Definition 3.1 it follows that the kth output can be rewritten in terms of the whole $\boldsymbol{\theta}(t)$ vector as

$$y_k(t) = h_k(\boldsymbol{u}_t, \boldsymbol{\theta}(t)) = \left(\boldsymbol{e}_k^T \otimes \boldsymbol{u}_t{}^{\circledast}\right) \, \boldsymbol{\theta}(t),$$

where $\boldsymbol{e}_k \in \mathbb{R}^l$ denotes the kth canonical basis vector in $\mathbb{R}^l$. We can therefore write

$$\frac{dh_k(\boldsymbol{u}_t, \boldsymbol{\theta}(t))}{d\boldsymbol{\theta}(t)} = \left(\boldsymbol{e}_k^T \otimes \boldsymbol{u}_t{}^{\circledast}\right) \quad (1 \leq k \leq l),$$

such that

$$\boldsymbol{C} = \left. \frac{dh\left(\boldsymbol{u}_t\,,\,\boldsymbol{\theta}(t)\right)}{d\boldsymbol{\theta}(t)} \right|_{\boldsymbol{\theta}(t)=\boldsymbol{m}(t)^-} = \begin{pmatrix} \boldsymbol{e}_1^T & \otimes & \boldsymbol{u}_t{}^{\circledast} \\ \boldsymbol{e}_2^T & \otimes & \boldsymbol{u}_t{}^{\circledast} \\ & \vdots & \\ \boldsymbol{e}_l^T & \otimes & \boldsymbol{u}_t{}^{\circledast} \end{pmatrix}. \tag{9}$$

The right-hand side of equation (9) can be written as

$$\begin{aligned} \sum_{k=1}^{l} \boldsymbol{e}_k \otimes \boldsymbol{e}_k^T \otimes \boldsymbol{u}_t{}^{\circledast} &= \left(\sum_{k=1}^{l} \boldsymbol{e}_k \otimes \boldsymbol{e}_k^T\right) \otimes \boldsymbol{u}_t{}^{\circledast} \\ &= \boldsymbol{I}_l \otimes \boldsymbol{u}_t{}^{\circledast}, \end{aligned}$$

which concludes the proof. ∎

The significance of Theorem 3.1 is that it implies through Theorem 2.1 that the $\boldsymbol{C}$ matrix in (7) is exactly represented by the rank-1 tensor network structure shown in Figure 2. As the $\boldsymbol{C}$ matrix does not depend on $\boldsymbol{\theta}$, the filtering density $p(\boldsymbol{\theta}(t)|\boldsymbol{y}_{1:t})$ is exactly Gaussian and using an EKF will therefore not introduce any approximation errors.

IV. Tensor network implementation of the EKF

The EKF equations in (5) and (6) require the manipulation of vectors and matrices of exponential dimensions. The explicit computation of these filter equations is therefore limited to small values of lN^d. For this reason all vector and matrix quantities will be represented by a low-rank MPS and MPO, respectively. The following lemmas explain how the initial mean vector $\boldsymbol{m}(0) \in \mathbb{R}^{lN^d}$ and initial covariance matrix $\boldsymbol{P}(0) \in \mathbb{R}^{lN^d \times lN^d}$ can be directly initialized in their corresponding MPS/MPO formats. The assumption is hereby made that all entries of the initial mean $\boldsymbol{m}(0)$ are identical.

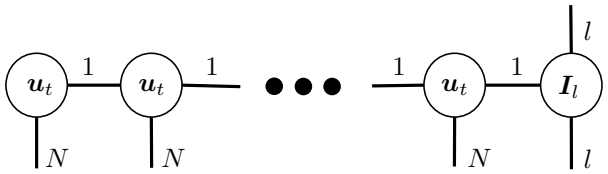

Fig. 2. Graphical representation of the $\boldsymbol{C}$ matrix as a rank-1 MPO with all dimensions labelled.

Lemma 1: Suppose we have a vector $\boldsymbol{m}(0) = m\,\mathbf{1}_{lN^d}$ where $m \in \mathbb{R}$. The corresponding MPS is unit-rank and consists of the following tensors

$$\begin{aligned} \boldsymbol{\mathcal{M}}^{(1)} &= \mathbf{1}_N \in \mathbb{R}^{1\times N\times 1}, \\ \boldsymbol{\mathcal{M}}^{(2)} &= \mathbf{1}_N \in \mathbb{R}^{1\times N\times 1}, \\ &\vdots \\ \boldsymbol{\mathcal{M}}^{(d)} &= \mathbf{1}_N \in \mathbb{R}^{1\times N\times 1}, \\ \boldsymbol{\mathcal{M}}^{(d+1)} &= m\mathbf{1}_l \in \mathbb{R}^{1\times l\times 1}. \end{aligned}$$

Lemma 1 follows directly from applying Theorem 2.1 to

$$\boldsymbol{m}(0) = m\,\mathbf{1}_l \;\otimes\; \mathbf{1}_N{}^{\circledast}.$$

The scalar m can be moved to any of the $d+1$ MPS tensors. The total storage cost of $\boldsymbol{m}(0)$ is through Lemma 1 reduced from lN^d to $l+N$ as the vector $\mathbf{1}_N$ needs to be stored only once. For the initial covariance matrix $\boldsymbol{P}(0)$ we can assume that it is a diagonal matrix with a uniform diagonal.

Lemma 2: Suppose we have a matrix $\boldsymbol{P}(0) = p\,\boldsymbol{I}_{lN^d}$ where $p > 0$ is a real scalar. The corresponding MPO is unit-rank and consists of the following tensors

$$\begin{aligned} \boldsymbol{\mathcal{P}}^{(1)} &= \boldsymbol{I}_N \in \mathbb{R}^{1\times N\times N\times 1}, \\ \boldsymbol{\mathcal{P}}^{(2)} &= \boldsymbol{I}_N \in \mathbb{R}^{1\times N\times N\times 1}, \\ &\vdots \\ \boldsymbol{\mathcal{P}}^{(d)} &= \boldsymbol{I}_N \in \mathbb{R}^{1\times N\times N\times 1}, \\ \boldsymbol{\mathcal{P}}^{(d+1)} &= p\boldsymbol{I}_l \in \mathbb{R}^{1\times l\times l\times 1}. \end{aligned}$$

Lemma 2 follows directly from applying Theorem 2.1 to

$$\boldsymbol{P}(0) = p\,\boldsymbol{I}_l \;\otimes\; \boldsymbol{I}_N{}^{\circledast},$$

where the scalar p can also be moved to any of the $d+1$ MPO tensors. The total storage cost of $\boldsymbol{P}(0)$ is through Lemma 2 reduced from l^2N^{2d} to l^2+N^2. A diagonal covariance matrix, however, does not model possible correlations between the parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_l$ of the l MISO Volterra models. Suppose, for example, that $l = 2$ and that we want to model that the coefficients of the matrix $(\boldsymbol{\theta}_1(0)\;\boldsymbol{\theta}_2(0))$ are row-wise correlated. This implies that the initial covariance matrix $P(0)$ has the following structure

$$\boldsymbol{P}(0) = \begin{pmatrix} \boldsymbol{P}_{11} & \boldsymbol{P}_{12} \\ \boldsymbol{P}_{12} & \boldsymbol{P}_{22} \end{pmatrix} \tag{10}$$

where each of the $\boldsymbol{P}_{11}, \boldsymbol{P}_{22}, \boldsymbol{P}_{12} \in \mathbb{R}^{N^d}$ matrices are diagonal. The construction of this $\boldsymbol{P}(0)$ in an MPO-form is given by the following lemma.

Lemma 3: The matrix as described in (10) can be constructed as a rank-4 MPO through

$$\begin{aligned}\boldsymbol{P}(0) =& \boldsymbol{e}_1 \otimes \boldsymbol{e}_1^T \otimes \boldsymbol{P}_{11} + \boldsymbol{e}_1 \otimes \boldsymbol{e}_2^T \otimes \boldsymbol{P}_{12} + \boldsymbol{e}_2 \otimes \boldsymbol{e}_1^T \otimes \boldsymbol{P}_{12} \\ &+ \boldsymbol{e}_2 \otimes \boldsymbol{e}_2^T \otimes \boldsymbol{P}_{22}.\end{aligned}$$

where $\boldsymbol{e}_1, \boldsymbol{e}_2$ are the canonical basis vectors in $\mathbb{R}^2$, $\boldsymbol{P}_{11}, \boldsymbol{P}_{22}, \boldsymbol{P}_{12}$ are rank-1 MPOs according to Lemma 2 and the addition of the four MPOs can be computed through Theorem 4.1.

It is possible to generate a rank-1 MPO for this specific case. By redefining $\boldsymbol{\theta}$ as the columnwise-vectorization of $(\boldsymbol{\theta}_1 \ \cdots \ \boldsymbol{\theta}_l)^T$, one can show that $\boldsymbol{P}(0)$ has the following rank-1 MPO

$$\boldsymbol{P}(0) = \boldsymbol{I}_N{}^{\circledast} \otimes \begin{pmatrix} p_{11} & \cdots & p_{1l} \\ \vdots & \ddots & \vdots \\ p_{1l} & \cdots & p_{ll} \end{pmatrix},$$

where p_{ij} denotes the covariance of all coefficients between Volterra models i and j. This redefinition of $\boldsymbol{\theta}$ also implies that $\boldsymbol{C} := \boldsymbol{u}_t{}^{\circledast} \otimes \boldsymbol{I}_l$.

With the initial mean vector and covariance matrix initialized as tensor networks, we can now explain how each of the EKF equations can be implemented directly in the tensor network format. Two major operations play an important role in each of the Kalman filter equations in (5) and (6): matrix addition and multiplication. We will explain how the results of each of these operations can be computed directly in tensor network form.

Theorem 4.1: ([7, p. 2308] Addition of two MPS) Let $\boldsymbol{\mathcal{A}}^{(1)}, \ldots, \boldsymbol{\mathcal{A}}^{(d)}$ be the MPS of a vector $\boldsymbol{a} \in \mathbb{R}^{N^d}$ with uniform MPS-ranks R_A and likewise for a vector $\boldsymbol{b}$. Then the MPS of $\boldsymbol{c} = \boldsymbol{a} + \boldsymbol{b}$ consists of the tensors

$$\boldsymbol{\mathcal{C}}^{(1)}(1, n_1, :) = \begin{pmatrix} \boldsymbol{\mathcal{A}}^{(1)}(1, n_1, :) & \boldsymbol{\mathcal{B}}^{(1)}(1, n_1, :) \end{pmatrix},$$

$$\boldsymbol{\mathcal{C}}^{(d)}(:, n_d, 1) = \begin{pmatrix} \boldsymbol{\mathcal{A}}^{(d)}(:, n_d, 1) \\ \boldsymbol{\mathcal{B}}^{(d)}(:, n_d, 1) \end{pmatrix},$$

and

$$\boldsymbol{\mathcal{C}}^{(k)}(:, n_k, :) = \begin{pmatrix} \boldsymbol{\mathcal{A}}^{(k)}(:, n_k, :) & 0 \\ 0 & \boldsymbol{\mathcal{B}}^{(k)}(:, n_k, :) \end{pmatrix},$$

when $k = 2, \ldots, d-1$. The MPS-ranks of $\boldsymbol{c}$ are $R_A + R_B$. The extension of Theorem 4.1 to the MPO-case involves the concatenation of both a row index n_k and column index m_k into one multi-index $[n_k m_k]$ and applying Theorem 4.1. The second important operation in the Kalman filter equations is matrix multiplication. We will work out this operation with tensor networks for the matrix vector case. The extension to matrix matrix multiplication involves the addition of an extra (column) index.

Theorem 4.2: (Matrix vector multiplication in MPS form) Let $\boldsymbol{\mathcal{A}}^{(1)}, \ldots, \boldsymbol{\mathcal{A}}^{(d)}$ be the MPO of a matrix $\boldsymbol{A} \in \mathbb{R}^{N^d \times N^d}$ and $\boldsymbol{\mathcal{X}}^{(1)}, \ldots, \boldsymbol{\mathcal{X}}^{(d)}$ the MPS for a vector $\boldsymbol{x} \in \mathbb{R}^{N^d}$. Then the entries $\boldsymbol{\mathcal{B}}^{(k)}([r_A r_X], n_k, [s_A s_X])$ of the MPS that represents $\boldsymbol{b} = \boldsymbol{A}\,\boldsymbol{x}$ are computed as

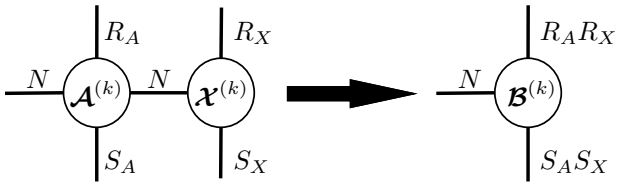

Fig. 3. The computation of the kth MPS tensor of $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}$ with all dimensions labelled.

$$\sum_{m_k=1}^{N} \boldsymbol{\mathcal{A}}^{(k)}(r_A, n_k, m_k, s_A)\, \boldsymbol{\mathcal{X}}^{(k)}(r_X, m_k, s_X) \tag{11}$$

for all $k = 1, \ldots, d$. The MPS-ranks of $\boldsymbol{b}$ are the product of the MPO-ranks of $\boldsymbol{A}$ with the corresponding MPS-ranks of $\boldsymbol{x}$.

The summation (11) is graphically represented in Figure 3 by the connected edge between the $\boldsymbol{\mathcal{A}}^{(k)}$ and $\boldsymbol{\mathcal{X}}^{(k)}$ nodes. With the initialization of $\boldsymbol{m}(0)$ and $\boldsymbol{P}(0)$ as described in Lemmas 1, 2, and 3, together with the two major operations in Theorems 4.1 and 4.2 one can implement all EKF equations directly in the MPS/MPO format. In the next section we investigate the benefit of using tensor networks by doing a detailed analysis of the computational complexity.

V. Computational complexity analysis

In this section a comparison is made of the computational complexity of the tensor network EKF implementation proposed in this article with the conventional implementation. It will be shown that using tensor networks for the implicit representation of all vectors and matrices lifts the curse of dimensionality and enables the identification of highly nonlinear systems. We assume that the MPO-ranks of the covariance matrix $\boldsymbol{P}(t)$ are all identical and denoted by R_P. The MPO that represents the Kalman gain $\boldsymbol{K}$ is obtained through the multiplication of $\boldsymbol{P}(t)^-$ with $\boldsymbol{C}^T$. Since all MPO-ranks of $\boldsymbol{C}$ are equal to one, this implies that the MPO-ranks of $\boldsymbol{K}$ will be identical to R_P. The computational complexity of the conventional EKF and tensor network-implementation are shown for the various filter equations in Table I. Similar to the tensor network Kalman filter described in [5], [6], the exponential complexity of the conventional implementation is also here completely lifted through the use of tensor networks. The MPO-rank of the covariance matrix $\boldsymbol{P}(t)$ is the only MPO-rank that appears in Table I and is therefore **the** determining factor for the tensor network computational complexity. Theorems 4.1 and 4.2 tell us how the implementation of each filter equation will invariably lead to increased MPS-ranks of $\boldsymbol{m}(t)$ and MPO-ranks of $\boldsymbol{P}(t)$, resulting in the following lemma.

Lemma 4: For the MIMO Volterra system identification problem in (3) we have that any MPS-rank $R_M(t)$ of $\boldsymbol{m}(t)$ and any MPO-rank $R_P(t)$ of $\boldsymbol{P}(t)$ follows the recursion

TABLE I

COMPUTATIONAL COMPLEXITY (FLOPS) OF CONVENTIONAL AND TENSOR NETWORK EFK.

	conventional EKF	tensor network EKF
$\boldsymbol{S}$	$O(l^3N^{2d})$	$O(dR_P^2N^2)$
$\boldsymbol{K}$	$O(l^3 + l^3N^{2d})$	$O(l^3 + dR_P^2N^2)$
$\boldsymbol{m}$	$O(l^2N^d)$	$O(l^2R_P)$
$\boldsymbol{P}$	$O(l^3N^{2d})$	$O(l^3 + (d-1)R_P^4N^2 + R_P^2N^2)$

TABLE II

IMPROVEMENTS OF TNEKF OVER TNKALMAN IN TERMS OF RELATIVE VALIDATION ERRORS IN 25 TRIALS.

	SNR (dB)	Improvement (×)
max	11.5036	4.3496
mean	9.6660	3.0735
min	8.0624	2.2850

equations

$$R_M(t+1) = R_M(t) + R_P(t),$$
$$R_P(t+1) = R_P(t) + R_P(t)^2 \quad \text{for all } t \geq 0.$$

The linear growth of $R_M(t)$ is due to the update step $\boldsymbol{m}(t)^- + \boldsymbol{K}\boldsymbol{v}$ as the MPS-rank of the Kalman gain $\boldsymbol{K}$ is R_P. Similarly, the quadratic growth of R_P results from the product $\boldsymbol{K}\boldsymbol{S}\boldsymbol{K}^T$ in the update of the covariance matrix. A rounding step [7, p. 2305] is required to truncate $R_M(t), R_P(t)$ through successive singular value decompositions on each of the MPS/MPO-tensors. This rounding step has a total computational complexity of $O(dNR^3)$.

VI. EXPERIMENT

In this section, we investigate two particular issues through numerical experiments. First, we illustrate the advantage of the proposed tensor network extended Kalman filter (TNEKF) in modeling correlations between the model parameters over the tensor network Kalman filter (TNKalman) [1]. In a second series of experiments we investigate the effect of the maximal MPO-rank of the covariance matrix $\boldsymbol{P}(t)$ on both the convergence of the EKF filter and its total runtime. For each of the experiments the state space model (3) was used. As described in detail in [11], [12], when using Kalman filtering for parameter estimation one can choose the covariance matrix of the system noise as $\boldsymbol{R}_w(t) := (\lambda^{-1} - 1)\boldsymbol{P}(t)$ with a forgetting factor $\lambda = 1$, which implies that all past data is given equal weight. All algorithms were implemented in MATLAB and ran on a desktop computer with 4 cores running at 2.7 GHz and 8 GB RAM. A MATLAB implementation of the proposed TNEKF can be downloaded from `https://github.com/IRENEKO/TNEKF`.

A. TNEKF vs. TNKalman for correlated model parameters

In contrast to the existing TNKalman, the proposed TNEKF is capable of modeling correlations of different

[1]A MATLAB implementation is freely available at `https://github.com/kbatseli/TNKalman`

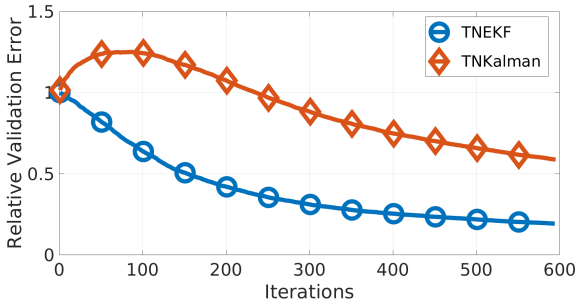

Fig. 4. Average relative validation errors of TNEKF and TNKalman over 100 trials.

Volterra models through its initial covariance matrix $\boldsymbol{P}(0)$. To evaluate the advantage of the proposed TNEKF over the existing TNKalman, we consider a single-input-single-output (SISO) Volterra system of degree 3 and memory 6. The model parameters were sampled from a standard normal distribution. We generate 10 output signals using an identical white noise standard normal input signal of 600 samples and different realizations of the measurement noise, which is Gaussian distributed with zero mean and a variance of 10^3. These 10 output signals are then combined and used in both the TNEKF and TNKalman filter to estimate the 3430 parameters of 10 SISO Volterra systems. The rounding tolerance used for truncating the MPO-ranks of the covariance matrices was set to 10^{-10}. With the TNKalman filter it is not possible to model the correlation between these 10 identical models and all covariance matrices are therefore diagonal matrices with a variance of 10. The initial covariance matrix of the TNEKF is constructed as described in Lemma 3 with all covariances set to 10. At each iteration of the TNEKF and TNKalman filters, we take the current estimate of the model parameters and use these estimates to run a simulation with a validation signal that consists again of a white noise input signal of 600 samples. The relative validation error is then defined as

$$\frac{\sqrt{\sum_{t=0}^{599} ||\boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t)||^2}}{\sqrt{\sum_{t=0}^{599} ||\boldsymbol{y}(t)||^2}}, \tag{12}$$

where $\boldsymbol{y}(t)$ contains the 10 output signals uncorrupted by measurement noise and $\hat{\boldsymbol{y}}(t)$ are the simulated outputs from the current model parameter estimates. We define the improvement of the relative validation error as $e_{\text{TNKalman}}/e_{\text{TNEKF}}$. A Monte-Carlo simulation of 100 runs was performed. The average runtime per iteration for TNEKF and TNKalman were 0.0190 and 0.0253 seconds, respectively. Detailed information on the signal-to-noise (SNR) ratios and relative validation error improvements are shown in Table II. The SNR is defined as

$$20 \log_{10}\left(\frac{A_h}{A_e}\right),$$

where A_h is the root mean square amplitude of $h(\boldsymbol{u}_t, \boldsymbol{\theta}(t))$ and A_e is the root mean square amplitude of the measurement noise $\boldsymbol{e}(t)$. The average SNR over the 100 runs

TABLE III

TOTAL RUNTIME AND AVERAGE RUNTIME PER ITERATION OF TNEKF FOR DIFFERENT MAXIMAL COVARIANCE MPO-RANKS.

MPO-rank	Total runtime (s)	Average runtime per iteration (s)
1	72	0.0072
2	109	0.0109
6	311	0.0311
10	1475	0.1475

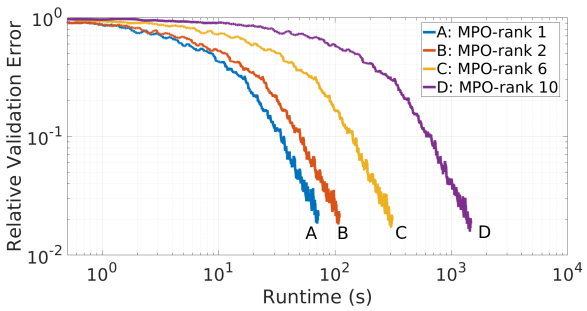

Fig. 5. Relative validation error as a function of the runtime for different maximal values of the MPO-ranks R_P.

is about 9.7 dB, while the relative validation error of the proposed TNEKF is on average 3 times smaller than that of TNKalman. In Figure 4, we plot the average relative validation errors of TNEKF and TNKalman as a function of the number of iterations. The ability of the TNEKF to model correlations between the models results in better accuracy. After 600 iterations, TNKalman converges to a relative validation error that can be reached by the TNEKF in about 100 iterations.

B. Effect of covariance MPO-ranks on convergence/runtime

In this experiment we investigate the effect of the MPO-rank R_P on both the total runtime and relative validation error of the proposed EKF. A single-input-two-output Volterra system of degree 4 and memory 9 was constructed, where all 20000 parameters were sampled from a standard normal distribution. A white noise input of 10^4 samples was used to generate the output signals, corrupted by a measurement noise with variance 10^{-2} resulting in a SNR of 66 dB. The rounding algorithm was adapted such that it truncates all MPO-ranks to a fixed given value in each of the experiments. Experiments were then run for fixed maximal covariance MPO-ranks $R_P = \{1, 2, 6, 10\}$. The relative validation error was computed according to (12) on a validation signal of 1000 samples. The relative validation errors as a function of the runtime for each of the fixed MPO-ranks are shown in Figure 5. A first observation is that all four filters converge to a relative validation error of about 10^{-2} after 10^4 iterations, irrespective of their MPO-ranks. As expected by the computational complexity analysis in Table I, both the total runtime and average runtime per iteration grow with an increasing maximal MPO-rank, as shown in Table III. A 10-fold increase of the maximal MPO-rank results in a 20 times higher runtime, which implies that there is no benefit in choosing an MPO-rank other than 1 if the maximal allowable rank is 10.

VII. CONCLUSION

A reformulation of the MIMO Volterra system identification problem as an extended Kalman filtering problem was presented. This reformulation lead to a simplification of the filter equations, together with the newly added capability of modeling correlations between different models. The curse of dimensionality associated with the number of model parameters that need to be identified has been lifted through the use of low-rank tensor networks. Numerical experiments have demonstrated the advantage of being able to model correlated models compared to the Tensor Network Kalman filter, together with the practical need for low-rank MPO representations of the covariance matrix.

REFERENCES

[1] R. Nowak and B. Van Veen, "Tensor product basis approximations for Volterra filters," *IEEE Transactions on Signal Processing*, vol. 44, no. 1, pp. 36–50, JAN 1996.

[2] G. Favier and T. Bouilloc, "Parametric complexity reduction of Volterra models using tensor decompositions," in *17th European Signal Processing Conference (EUSIPCO)*, Aug 2009.

[3] G. Favier, A. Y. Kibangou, and T. Bouilloc, "Nonlinear system modeling and identification using Volterra-PARAFAC models," *Int. J. Adapt. Control Signal Process*, vol. 26, no. 1, pp. 30–53, Jan. 2012.

[4] K. Batselier, Z. M. Chen, and N. Wong, "Tensor Network alternating linear scheme for MIMO Volterra system identification," *Automatica*, vol. 84, pp. 26–35, 2017.

[5] ——, "A Tensor Network Kalman filter with an application in recursive MIMO Volterra system identification," *Automatica*, vol. 84, pp. 17–25, 2017.

[6] K. Batselier and N. Wong, "Matrix output extension of the tensor network Kalman filter with an application in MIMO Volterra system identification," *Automatica*, vol. 95, pp. 413–418, 2018.

[7] I. V. Oseledets, "Tensor-train decomposition," *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2295–2317, 2011.

[8] ——, "Approximation of $2^d \times 2^d$ matrices using tensor decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 31, no. 4, pp. 2130–2145, Jun. 2010.

[9] K. Batselier and N. Wong, "Computing low-rank approximations of large-scale matrices with the Tensor Network randomized SVD," *SIAM J. Matrix Anal. Appl.*, vol. 39, pp. 1221–1244, 2018.

[10] S. Särkkä, *Bayesian Filtering and Smoothing*. New York, NY, USA: Cambridge University Press, 2013.

[11] A. T. Nelson, "Nonlinear estimation and modeling of noisy time-series by dual Kalman filtering methods," *Doctor of Philosopy, Oregon Graduate Institute of Science and Technology*, 2000.

[12] S. Haykin, *Kalman filtering and neural networks*. John Wiley & Sons, 2004, vol. 47.