Bayesian Framework for a MIMO Volterra Tensor Network

Memmel, Eva; Menzen, Clara; Batselier, Kim

DOI
10.1016/j.ifacol.2023.10.341

Publication date
2023

Document Version
Final published version

Published in
IFAC-PapersOnLine

Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 56-2 (2023) 7294–7299

Bayesian Framework for a MIMO Volterra Tensor Network ⋆

Eva Memmel * **Clara Menzen** * **Kim Batselier** *

* *Delft University of Technology , Delft, Netherlands*

(e-mail: {e.m.memmel, c.m.menzen, kim.batselier}@ tudelft.nl).

Abstract: This paper proposes a Bayesian Volterra tensor network (TN) to solve high-order discrete nonlinear multiple-input multiple-output (MIMO) Volterra system identification problems. Using a low-rank tensor network to compress all Volterra kernels at once, we avoid the exponential growth of monomials with respect to the order of the Volterra kernel. Our contribution is to introduce a Bayesian framework for the low-rank Volterra TN. Compared to the least squares solution for Volterra TNs, we include prior assumptions explicitly in the model. In particular, we show for the first time how a zero-mean prior with diagonal covariance matrix corresponds to implementing a Tikhonov regularization for the MIMO Volterra TN. Furthermore, adopting a Bayesian viewpoint enables simulations with Bayesian uncertainty bounds based on noise and prior assumptions. In addition, we demonstrate via numerical experiments how Tikhonov regularization prevents overfitting in the case of higher-rank TNs.

Keywords: Nonlinear system identification, Bayesian methods, Volterra series, Tensor networks

1. INTRODUCTION

One of the most challenging tasks in system identification is modelling nonlinear systems. A commonly used model structure for this purpose is the truncated Volterra series, a nonlinear extension of the linear finite impulse response model. The truncated Volterra series is useful when there is no precise information about the exact nonlinear system behaviour. However, a significant drawback of the Volterra series is the exponential growth of parameters with respect to the selected order of the model. This exponential growth is known as the curse of dimensionality. Consequently, the application of Volterra models is limited to weakly-nonlinear systems.

There are different ways of reducing the complexity of the Volterra kernel. One approach is to exploit prior knowledge about the structure of the Volterra kernel and include it in the model. Including prior knowledge can, for example, be achieved in a Bayesian framework (Chen et al., 2011). As the Volterra series is a nonlinear extension of the finite impulse response model, Volterra kernels are expected to be smoothly decaying. This knowledge can be encoded in a probabilistic Bayesian prior, as first shown for linear system identification by Pillonetto and De Nicolao (2010). This work has been extended to parametric Volterra models by introducing a smoothly decaying prior covariance matrix for each Volterra kernel by Birpoutsoukis et al. (2018). So far, applications of the parametric approach have been limited to third-order kernels. This approach has been extended to higher order non-parametric Volterra models (Dalla Libera et al., 2021), using a similar matrix structure, limiting the applications to models with a small number of samples. A different way of enforcing additional structure is assuming a sparse structure. For example, a sparse Bayesian learning algorithm can be introduced for a Volterra model (Miao et al., 2019). However, the authors have strong reasons to assume that most Volterra kernels are irrelevant; thus, the system can be represented with a sparse structure. Another way of adding structure is to compress the Volterra kernels by imposing a low-rank tensor structure onto them. Favier et al. (2012) apply both low-rank Tucker and canonical polyadic decompositions. Batselier et al. (2017) uses one tensor train decomposition (Oseledets, 2011) for all Volterra kernels at once to introduce a multiple-input-multiple-output (MIMO) Volterra tensor network (TN). The MIMO Volterra TN exploits the multi-linearity of the tensor train decomposition and solves the identification problem with the alternating least squares (ALS) method. This way, the exponential growth of the Volterra coefficients is completely avoided. This work showed how to estimate a 10th-order Volterra system within seconds on a standard personal computer.

Even though the MIMO Volterra system provides a considerable advantage due to the low computational complexity, it has a significant drawback: The TN model tends to overfit for larger TN ranks. Batselier et al. (2017) already discuss the hypothesis that a regularized TN-model will counteract the problem of overfitting. Unfortunately, the TN model does not provide an option to include prior knowledge about the Volterra model yet. Furthermore, available priors for Volterra kernels (Pillonetto and De Nicolao, 2010; Birpoutsoukis et al., 2018; Dalla Libera et al., 2021) cannot be straightforwardly applied to the TN-model because the MIMO Volterra TN compresses all kernels at once. The contribution of this paper is to address the limitations of the MIMO Volterra TN-model and test the hypothesis formulated by Batselier et al. (2017): For the first time, we introduce a Bayesian framework for the MIMO Volterra TN-model. Additionally, we provide a

⋆ This work is supported by the TU Delft AI Labs programme.

10.1016/j.ifacol.2023.10.341

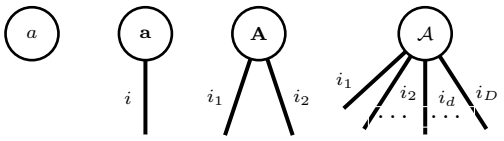

Fig. 1. Graphical notation of a scalar, a vector, a matrix and a D-way tensor. A node represents an object, an edge a specific index. A node with D edges depicts a D-way tensor.

Bayesian perspective on the Tikhonov regularization in the MIMO Volterra TN. The Bayesian framework presented in this paper is an adaptation of the Bayesian alternating linear scheme presented by Menzen et al. (2022).

2. TENSOR BASICS

A D-dimensional array is a D-way or Dth-order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$. Each entry $\boldsymbol{\mathcal{A}}(i_1, i_2, \ldots, i_D)$ is completely determined by D indices $i_1, i_2, \ldots, i_D$. Commonly used in system identifications are scalars ($D = 0$), which are zero-order tensors, vectors ($D = 1$), which are first-order tensors and matrices ($D = 2$), which are second-order tensors. Higher-order tensors are tensors for which the order is three or higher. Tensors have proven to be a valuable tool to lift the curse of dimensionality in Volterra system identification. An extensive tutorial on low-rank TNs for nonlinear system identification is given by Batselier (2022). In this section, we revise some useful higher-order tensor basics, operations and decompositions, specifically focusing on nonlinear tensor-network Volterra system identification.

We denote scalars by the italic lowercase (e.g. a), vectors by boldface lowercase (e.g. $\mathbf{a}$) and matrices by boldface capital (e.g. $\mathbf{A}$) letters. Tensors are denoted with calligraphic capital letters (e.g. $\boldsymbol{\mathcal{A}}$). Indices are denoted with italic lowercase letters (e.g. i_d), and their upper bound is denoted with an italic uppercase letter (e.g. I_d). A graphical notation can be used to simplify the formulation of equations. Figure 1 depicts the graphical representation of a scalar, a vector, a matrix and a D-way tensor, where a node represents an object and an edge a specific index. It is possible to rewrite a tensor as a matrix or vector by matricization or vectorization (Kolda and Bader, 2009). The vectorization $\mathrm{vec}(\boldsymbol{\mathcal{A}}) \in \mathbb{R}^{I_1 I_2 \ldots I_D}$ of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$ is the vector obtained from combining all indices $i_1, i_2, \ldots, i_D$ into a single multi-index $[i_1 i_2 \ldots i_D]$ by

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

Equation (1) can also be used to reshape a vector $\mathrm{vec}(\boldsymbol{\mathcal{A}}) \in \mathbb{R}^{I_1 I_2 \ldots I_D}$ into a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$.

The size of the matrix containing all Volterra weights grows exponentially with the order of the Volterra system. It is possible to reduce the storage complexity of that matrix to linear in the order of the Volterra system by representing the Volterra weights in a tensor train matrix (TTM) form (Oseledets, 2010).

Definition 1. Let $\mathbf{A} \in \mathbb{R}^{I^D \times J^D}$, such that $I^D = I_1 I_2 \ldots I_D$, $J^D = J_1 J_2 \ldots J_D$. Then the TTM of $\mathbf{A}$ consists of D 4-way tensors $\boldsymbol{\mathcal{A}}^{(d)} \in \mathbb{R}^{R_d \times I_d \times J_d \times R_{d+1}}$, such that a matrix entry $A_{[i_1 i_2 \cdots D][j_1 j_2 \ldots j_D]}$is given by

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

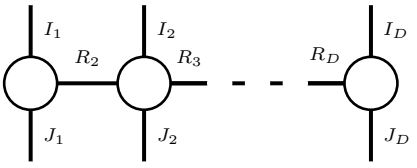

Fig. 2. Graphical notation of a TTM representing a matrix $\mathbf{A} \in \mathbb{R}^{I_1 I_2 \ldots I_D \times J_1 J_2 \ldots J_D}$. $R_1 = R_{D+1} = 1$ is usually not depicted.

The tensors $\boldsymbol{\mathcal{A}}^{(d)}$, $d = 1, \ldots, D$ are called core tensors. The dimensions $R_1, R_2, \ldots, R_{D+1}$ are called TTM-ranks, with $R_1 = R_{D+1} = 1$ per definition. Index-summations or contractions as in Equation (2) can also be denoted in the graphical notation by connecting the two corresponding nodes with the corresponding edge. The graphical notation for the TTM is shown in Figure 2. If $R_2, \ldots, R_D$ are small; the TTM is a low-rank representation of the original matrix. The advantage of representing the matrix $\mathbf{A}$ in a low-rank TTM format is the change in storage costs. With $I = \max(I_1, \ldots, I_D)$, and J, R defined accordingly, the storage costs change from $\mathcal{O}(I^D J^D)$ to $\mathcal{O}(DIJR^2)$. Thus, for a small R, the storage complexity reduces from an exponential dependency on the order D to a linear dependency.

3. TENSOR NETWORK MIMO VOLTERRA SYSTEM

In this section, we give a brief introduction to the TN MIMO Volterra System presented by Batselier et al. (2017). To simplify the notation, consider the following truncated, D-th order SISO Volterra system with memory M

$$y(n) = \sum_{d=0}^{D} \sum_{m_1, \ldots, m_d}^{M-1} \boldsymbol{\mathcal{W}}_d(m_1, \ldots m_d) \prod_{j=1}^{d} u(n - m_j) + e(n)\,. \tag{3}$$

From this equation, it immediately follows that the size of Volterra kernels $\boldsymbol{\mathcal{W}}_d$ grows exponentially with the order of the system D. This curse of dimensionality can be lifted by imposing a TN structure on all Volterra kernels at once (Batselier et al., 2017). To derive the general MIMO Volterra TN, let us now consider a MIMO system with P inputs and L outputs, which are given by

$$\mathbf{u}(n) = (u_1(n)\ u_2(n) \ldots u_P(n))^\top \in \mathbb{R}^P$$
$$\mathbf{y}(n) = (y_1(n)\ y_2(n) \ldots y_L(n))^\top \in \mathbb{R}^L\,,$$

for $n = 1, \ldots, N$. The objective is to derive a set of linear matrix equations that we can decompose into a low-rank TN and thus lift the curse of dimensionality. The first step is to rewrite the inputs $\mathbf{u}(n)$ corresponding to Batselier (2022) as

$$\mathbf{u}_n = \left(1\ \mathbf{u}(n)^\top \ldots \mathbf{u}(n - M + 1)^\top\right)^\top \in \mathbb{R}^I, \tag{4}$$

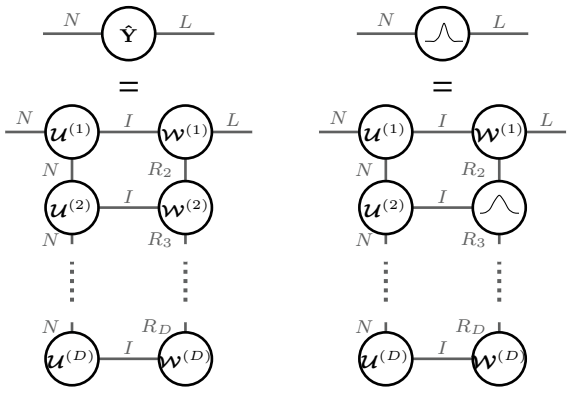

Fig. 3. Left: TN MIMO Volterra system for times $n = 1, \dots, N$, where row indices point to the left and column indices to the right. Right: Bayesian framework for TN MIMO Volterra system, the elements of one TN-core are now random variables, indicated by a bell curve. With the introduction of the Bayesian framework, the simulations $\hat{\mathbf{Y}}$ also become random variables.

with $I := PM + 1$. Including the 1 allows us to define a vector $\mathbf{u}_n^{D\otimes}$, which contains all monomials of the inputs from degree 0 up to D as

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

where D is the order of the Volterra system and $\otimes$ denotes the Kronecker product. With this definition, we can write the output of a MIMO Volterra system as

$$\hat{\mathbf{y}}(n)^\top = (\mathbf{u}_n^{D\otimes})^\top \mathbf{W}, \tag{6}$$

where $\mathbf{W} \in \mathbb{R}^{I^D \times L}$ contains all Volterra kernel coefficients. We now obtain a set of linear matrices by writing out equation (6) for all measurements $n = 1, \dots, N$

$$\underbrace{\hat{\mathbf{Y}}}_{N\times L} = \underbrace{\mathbf{U}}_{N\times I^D} \underbrace{\mathbf{W}}_{I^D\times L}. \tag{7}$$

The size of the matrix $\mathbf{W}$ is problematic: Solving equation (6) for $\mathbf{W}$, has the computational complexity $\mathcal{O}(LI^{3D})$. Since the computational complexity scales exponentially with D, solving for $\mathbf{W}$ becomes quickly infeasible as D grows. The computational complexity of solving equation (7) can be reduced to linear in D by solving it iteratively with the alternating linear scheme (ALS) (Batselier et al., 2017). The corresponding algorithm is called MIMO Volterra ALS (MVALS). Applying the ALS requires imposing a low-rank TN structure on equation (7), such that $\mathbf{U}$ and $\mathbf{W}$ are TTMs with TTM-cores $\boldsymbol{\mathcal{U}}^{(d)}$ and $\boldsymbol{\mathcal{W}}^{(d)}$ for $d = 1, ..., D$, respectively. The TTMs representing $\mathbf{U}$ and $\mathbf{W}$ form the MIMO Volterra TN. Figure 3 on the left is a graphical representation of the MIMO Volterra TN. To simplify the notation, we will assume equal rank R for all ranks $R_2 = \cdots = R_D = R$. Note that it is a low-rank compression of all Volterra kernels combined. Thus, directly linking one specific TN-core to one specific Volterra kernel is impossible. Due to the multilinear structure of the TTM, it is possible to solve for one TTM-core $\boldsymbol{\mathcal{W}}^{(d)}$ at a time. To derive the update equation for one core, we first define the matrix $\mathbf{U}_{<d}$ and the vector $\mathbf{u}_{>d}$ for one sample according to (Batselier, 2022)

$$\mathbf{U}_{<d} := \left(\boldsymbol{\mathcal{W}}^{(1)} \times_2 \mathbf{u}_n\right) \dots \left(\boldsymbol{\mathcal{W}}^{(d-1)} \times_2 \mathbf{u}_n\right) \in \mathbb{R}^{L\times R}$$
$$\mathbf{u}_{>d} := \left(\boldsymbol{\mathcal{W}}^{(d+1)} \times_2 \mathbf{u}_n\right) \dots \left(\boldsymbol{\mathcal{W}}^{(D)} \times_2 \mathbf{u}_n\right) \in \mathbb{R}^{R\times 1}.$$

Now, we can define the matrix $\mathbf{U}_{\setminus d}$ in a row-wise Kronecker product structure for all samples $n = 1, \dots, N$

$$\mathbf{U}_{\setminus d} := \begin{pmatrix} \mathbf{u}_{>d}^\top \otimes \mathbf{u}_1^\top \otimes \mathbf{U}_{<d} \\ \mathbf{u}_{>d}^\top \otimes \mathbf{u}_2^\top \otimes \mathbf{U}_{<d} \\ \vdots \\ \mathbf{u}_{>d}^\top \otimes \mathbf{u}_N^\top \otimes \mathbf{U}_{<d} \end{pmatrix} \in \mathbb{R}^{NL\times R^2 I}. \tag{8}$$

To simplify the notation further, we define $\hat{\mathbf{y}} := \mathrm{vec}(\hat{\mathbf{Y}})$ and $\mathbf{w}_d := \mathrm{vec}(\boldsymbol{\mathcal{W}}^{(d)})$. Updating one core $\boldsymbol{\mathcal{W}}^{(d)}$ is now equivalent to solving the linear system

$$\hat{\mathbf{y}} = \mathbf{U}_{\setminus d}\mathbf{w}_d, \tag{9}$$

with $\hat{\mathbf{y}} \in \mathbb{R}^{NL}$ and $\mathbf{w}_d \in \mathbb{R}^{RIR}$. In this context, one iteration of the ALS is defined as sequentially updating all cores once. Thus, one iteration of the ALS has the computational complexity of $\mathcal{O}(D(R^2I)^3)$ and the algorithm converges after a few iterations (Batselier et al., 2017). Choosing a low-rank representation of $\mathbf{W}$, thus choosing a small value for the rank R, has the potential to lift the curse of dimensionality and achieve a significant reduction of the computational complexity, from exponential in D with $\mathcal{O}(LI^{3D})$ to linear in D with $\mathcal{O}(D(R^2I)^3)$.

4. BAYESIAN TN MIMO VOLTERRA SYSTEM IDENTIFICATION

The key contribution of this paper is to propose a framework that allows to include prior knowledge into the MIMO Volterra TN, namely the Bayesian MVALS (BAMVALS). In the first subsection, we derive the proposed algorithm for imposing a Bayesian prior on one TN-core. In the second section, we choose a specific prior and show, how this prior can be reinterpreted for the MIMO Volterra TN as Tikhonov regularization. The theoretical considerations from this section are supported by numerical experiments in the next section.

4.1 General framework

To simplify the notation, we now consider the SISO-system

$$\underbrace{\mathbf{y}}_{N} = \underbrace{\mathbf{U}}_{N\times I^D} \underbrace{\mathbf{w}}_{I^D} + \underbrace{\mathbf{e}}_{N} \tag{10}$$

and assume Gaussian white noise $\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. Our proposed method is summarised in Algorithm 1. The algorithm relies on three central steps:
In the first step we solve Equation (10) for $\mathbf{w}$ with the MVALS identification algorithm (Batselier et al., 2017) by updating one TN-core at a time with Equation (9).
The second step of Algorithm 1 enables to incorporate prior knowledge into the system. It requires a more detailed derivation to show how to include prior knowledge into the MIMO Volterra TN with our proposed method. We begin with the TTM that represents the vector $\mathbf{w} \in \mathbb{R}^{I^D}$ in Equation (10). We use the multi-linear structure of TTMs and rewrite the vector $\mathbf{w}$ as a function that is linear in the core $\mathbf{w}_d \in \mathbb{R}^{RIR}$ given by

$$\mathbf{w} = \mathbf{W}_{\setminus d}\mathbf{w}_d. \tag{11}$$

The matrix $\mathbf{W}_{\setminus d} \in \mathbb{R}^{I^D \times RIR}$ is a contraction over all but the dth core of the TN representing the tensor $\boldsymbol{\mathcal{W}}_d$. From this it follows directly that $\mathbf{W}_{\setminus d}$ is computed with

$$\mathbf{W}_{\setminus d} = \mathbf{W}_{>d} \otimes \mathbf{I}_{RIR} \otimes \mathbf{W}_{<d}^\top \in \mathbb{R}^{I^D \times RIR},$$

where $\mathbf{W}_{>d}$ and $\mathbf{W}_{<d}$ are matrices computed by contracting all cores, whose ordinal number is smaller or bigger than d, respectively. The number of rows of $\mathbf{W}_{\setminus d}$ grows exponentially with D and thus, $\mathbf{W}_{\setminus d}$ can become very large. Consequently, it is important to mention that $\mathbf{W}_{\setminus d}$ is never computed explicitly. Instead, we implement an efficient computation exploiting a row-wise Kronecker product structure similar to the computation of $\mathbf{U}_{\setminus d}$ in Equation (8). Additionally, note that Equation (11) can be interpreted as a linear projection of Volterra kernels $\mathbf{w}$ onto a smaller subspace. With this reinterpretation of the TTM as a linear projection, we have laid the foundation for the next steps.

Now we can include prior knowledge in a low-rank MIMO Volterra TN. In the second step of Algorithm 1, we perform a Bayesian update (Menzen et al., 2022). A central difference to the algorithm presented by Menzen et al. (2022) is that we include the prior knowledge on only one core $\mathbf{w}_d$ and use the deterministic matrix $\mathbf{W}_{\setminus d}$ to project the prior knowledge on the vector $\mathbf{w}$ to make simulations. Thanks to the linear projection, we are able to bypass the computational bottleneck, namely the expensive nonlinear transformation required to make simulations in a full Bayesian TN as discussed by Menzen et al. (2022). For the core $\mathbf{w}_d$, we consider a prior Gaussian distribution

$$p(\mathbf{w}_d) = \mathcal{N}(\mathbf{m}_d^0, \mathbf{P}_d^0),$$

where $\mathbf{m}_d^0$ is the prior mean and $\mathbf{P}_d^0$ is the prior covariance of the d-th core $\mathbf{w}_d$, respectively. A graphical representation of the new MIMO Volterra TN for $d = 2$ is depicted in Figure 3 on the right. In Equation (10), we assume the noise $\mathbf{e}$ to be Gaussian. It follows that the likelihood is Gaussian, too, given by

$$p(\hat{\mathbf{y}} \mid \mathbf{w}_d) = \mathcal{N}(\mathbf{m}_{\hat{\mathbf{y}}}, \sigma^2 \mathbf{I}),$$

where $\mathbf{m}_{\hat{\mathbf{y}}} = \mathbf{U}\mathbf{W}_{\setminus d}\mathbf{w}_d$. Then the mean $\mathbf{m}_d^+$ and covariance $\mathbf{P}_d^+$ of the posterior distribution

$$p(\mathbf{w}_d \mid \hat{\mathbf{y}}) = \mathcal{N}(\mathbf{m}_d^+, \mathbf{P}_d^+)$$

of the updated core $\mathbf{w}_d$ is given by

$$\mathbf{P}_d^+ = \left[(\frac{\mathbf{W}_{\setminus d}^\top \mathbf{U}^\top \mathbf{U} \mathbf{W}_{\setminus d}}{\sigma^2} + (\mathbf{P}_d^0)^{-1} \right]^{-1} \tag{12}$$

$$\mathbf{m}_d^+ = \mathbf{P}_d^+ \left[\frac{\mathbf{W}_{\setminus d}^\top \mathbf{U}^\top \hat{\mathbf{y}}}{\sigma^2} + (\mathbf{P}_d^0)^{-1} \mathbf{m}_d^0 \right]. \tag{13}$$

With the third step and final step of Algorithm 1, we can first project the posterior distribution from the smaller subspace $p(\mathbf{w}_d \mid \hat{\mathbf{y}})$ to $p(\mathbf{w} \mid \hat{\mathbf{y}}) = \mathcal{N}(\mathbf{m}_\mathbf{w}, \mathbf{P}_\mathbf{w})$ with

$$\mathbf{P}_\mathbf{w} = \mathbf{W}_{\setminus d} \mathbf{P}_d^+ \mathbf{W}_{\setminus d}^\top \tag{14}$$

$$\mathbf{m}_\mathbf{w} = \mathbf{W}_{\setminus d} \mathbf{m}_d^+ \tag{15}$$

Then we use the distribution $p(\mathbf{w} \mid \hat{\mathbf{y}})$ to compute $p(\hat{\mathbf{y}}) = \mathcal{N}(\mathbf{m}_{\hat{y}}^+, \boldsymbol{\Sigma})$ for unseen inputs with

$$\boldsymbol{\Sigma} = \mathbf{U}\mathbf{P}_\mathbf{w}\mathbf{U}^\top = \mathbf{U}\mathbf{W}_{\setminus d}\mathbf{P}_d^+\mathbf{W}_{\setminus d}^\top\mathbf{U}^\top, \tag{16}$$

$$\mathbf{m}_{\hat{y}}^+ = \mathbf{U}\mathbf{m}_\mathbf{w} = \mathbf{U}\mathbf{W}_{\setminus d}\mathbf{m}_d^+. \tag{17}$$

Again, it is important to note that the matrices $\mathbf{U}$ and $\mathbf{W}_{\setminus d}$ are not constructed explicitly. The product $\mathbf{U}\mathbf{W}_{\setminus d}$ is computed efficiently by exploiting the row-wise Kronecker product structure presented in Equation (8). That yields the computational complexity $\mathcal{O}(DNR^2I)$ for predicting the mean and $\mathcal{O}(DNI^2R^4)$ for predicting the confidence bounds.

Algorithm 1 BAMVALS: Including prior knowledge to Volterra system identification

Require: N samples of input u and output y, degree D, memory M, TN ranks $R_2, \ldots, R_D$, prior mean $\mathbf{m}_d^0$, prior covariance $\mathbf{P}_d^0$, noise variance σ^2

Ensure: Simulations of $\hat{y}$ with confidence bounds

1: Compute $\boldsymbol{\mathcal{W}}^{(1)}, \boldsymbol{\mathcal{W}}^{(2)}, ..., \boldsymbol{\mathcal{W}}^{(D)}$ with MVALS using Equation (9).
2: Compute $p(\mathbf{w}_d \mid \hat{\mathbf{y}})$ with Bayesian update in projected subspace with Equation (12) and (13).
3: Simulate unseen data to obtain mean with confidence bounds, using Equation (16) and (17).

4.2 Tikhonov regularization

For this paper, we showcase how to include a prior to the MIMO Volterra TN and analyse its effect. The TN-model in Figure 3 on the left is prone to overfitting when the TN-ranks are chosen too large. This can be resolved by implementing a Bayesian Tikhonov prior. To achieve a Tikhonov regularization, we choose the entries of the core $\mathbf{w}_d$ to be normally distributed with

$$p(\mathbf{w}_d) = \mathcal{N}(\mathbf{0}, \frac{1}{\lambda}\mathbf{I}). \tag{18}$$

Inserting the selected prior into Equations (12) and (13) yields the new update rules for the posterior distribution $p(\mathbf{w}_d \mid \hat{\mathbf{y}}) = \mathcal{N}(\mathbf{m}_d^+, \mathbf{P}_d^+)$ for the core $\mathbf{w}_d$

$$\mathbf{P}_d^+ = \left[\frac{\mathbf{W}_{\setminus d}^\top \mathbf{U}^\top \mathbf{U} \mathbf{W}_{\setminus d}}{\sigma^2} + \lambda \mathbf{I} \right]^{-1} \tag{19}$$

$$\mathbf{m}_d^+ = \mathbf{P}_d^+ \left[\frac{\mathbf{W}_{\setminus d}^\top \mathbf{U}^\top \hat{\mathbf{y}}}{\sigma^2} \right]. \tag{20}$$

For FIR models, the relationship between a Bayesian perspective and regularized problems has been discussed before, e.g. by Chen et al. (2011). We revisit the discussion in the specific context of the MIMO Volterra TN and the prior chosen in this paper. First, it is essential to point out that the MIMO Volterra ALS proposed by Batselier et al. (2017) computes the cores $\boldsymbol{\mathcal{W}}^{(1)}, \boldsymbol{\mathcal{W}}^{(2)}, ..., \boldsymbol{\mathcal{W}}^{(D)}$ in such a way that the matrix $\mathbf{W}_{\setminus d}$ is orthogonal. Then, it can be shown that implementing Equation (19) and (20) corresponds to updating the core $\mathbf{w}_d$ with choosing a quadratic loss function combined with a Tikhonov regularization. The resulting least squares problem to update the mean of $\mathbf{w}_d$ is then given as

$$\min_{\mathbf{w}_d} \|\hat{\mathbf{y}} - \mathbf{U}\mathbf{W}_{\setminus d}\mathbf{w}_d\|_2^2 + \tilde{\lambda}\|\mathbf{W}_{\setminus d}\mathbf{w}_d\|_2^2, \tag{21}$$

where $\tilde{\lambda} = \sigma^2/\lambda$ and $\|\cdot\|_2$ indicates the L^2-norm.

In fact, by choosing (18) as a prior, we can address a problem of the MVALS discussed in the work by Batselier et al. (2017). There, Equation (9) is solved by computing the pseudo-inverse of $\mathbf{U}_{\setminus d}$ since it is a singular matrix. This pseudo-inverse has a regularizing effect but still tends to overfit for relatively large TN ranks. This is especially

problematic since the TN ranks are hyperparameters of the MVALS and, therefore, not known in advance. This way, the Tikhonov regularization allows the user to not worry about choosing all the TN ranks too high and still achieve good simulation performance. Furthermore, the choice of the $(D-1)$ potentially different TN ranks $R_2, \dots, R_D$ becomes less important over choosing the single hyperparameter $\tilde{\lambda}$ when a strong preference on the prior is assumed.

5. NUMERICAL EXPERIMENTS

In this section, we want to discuss the effect of including the prior stated in Equation (18) into the MIMO Volterra TN and give an example of simulations with Bayesian uncertainty bounds. With a limited number of training measurement samples, an excessive number of parameters is expected to lead to an increased root mean squared error (RMSE) for the validation data. We show how this effect can be countered with regularization. For this, we conduct numerical experiments with the Cascaded Tanks Benchmark presented by Schoukens et al. (2016). All computations were done on a Dell Inc. Latitude 7410 laptop with 16 GB of RAM and an Intel Core i7-10610U CPU running at 1.80 GHz. For all experiments presented in the paper, we use a grid search to fix the order $D = 2$, the memory $M = 95$, and, if applicable, the noise $\sigma^2 = 0.0165$. For all experiments, we impose the prior knowledge on the core $d = 2$. The choice is arbitrary in the example since both TN cores have the same number of coefficients. For multiple cores with different sizes, choosing the core containing the most elements would be sensible to achieve the highest constraining effect. All implementations are done in MATLAB R2020b; the code can be found at `https://github.com/blixi/BAMVALS`.

5.1 Benchmark description

The Cascaded Tanks Benchmark is a nonlinear benchmark with a relatively short data record ($N = 1024$), combining both soft and hard (due to overflow) nonlinearities (Schoukens et al., 2016). It consists of two water tanks. Water is pumped into the upper tank from a reservoir, from where it can flow into the lower tank and back into the reservoir. The input signal controls the water pump, and the output signal measures the water level in the lower tank. If the input signal is too large, both tanks can overflow. The input-output behaviour of the system is modelled with a Volterra series. For the estimation, both the MVALS algorithm (Batselier et al., 2017) and the proposed BAMVALS algorithm, see Algorithm 1, is used for the training data set to estimate the Volterra kernels. The performance of each model is evaluated on the transient-free validation data set with the RMSE

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(\hat{\mathbf{y}} - \mathbf{y})^2} \tag{22}$$

as error-index, where N is the number of transient-free data points of the validation data, $\hat{\mathbf{y}}$ is the simulated output and $\mathbf{y}$ is the validation output.

R	MVALS		BAMVALS	
	time (s)	RMSE	time (s)	RMSE
3	$\mathbf{0.5 \pm 0.1}$	$\mathbf{0.96 \pm 0.06}$	0.5 ± 0.2	0.99 ± 0.51
6	1.6 ± 0.3	1.63 ± 0.35	1.7 ± 0.3	0.78 ± 0.11
9	5.5 ± 0.9	5.02 ± 1.87	5.6 ± 0.9	0.96 ± 0.35
12	8.2 ± 0.3	256 ± 70	8.3 ± 0.3	0.82 ± 0.05
24	15.9 ± 0.4	8471 ± 6295	16.4 ± 0.4	0.67 ± 0.01
48	24.9 ± 1	2498 ± 1	$\mathbf{27.2 \pm 1.1}$	$\mathbf{0.63 \pm 0.00}$
72	32.6 ± 1.2	2501 ± 0	38.0 ± 2.5	0.63 ± 0.00
96	43.3 ± 0.6	3073 ± 0	54.2 ± 0.8	0.63 ± 0.00

Table 1. Training time and RMSE on validation data for MVALS and BAMVALS ($M = 95$, D = 2, $\tilde{\lambda} = 330$ with $1/\lambda = 20,000$, $\sigma^2 = 0.0165$) for an increasing rank R, averaged over 10 iterations. The results show a clear improvement of the RMSE with the BAMVALS with only a small increase in training time.

5.2 Regularizing effect

At first, we compare the RMSE and runtime of the MVALS and BAMVALS to show the regularizing effect of the zero-mean prior. For this, we consider the training time and the RMSE for the MVALS and the BAMVALS averaged over ten iterations. We set the hyperparameters of the BAMVALS to $1/\lambda = 20,000$, $\sigma^2 = 0.0165$. The results listed in Table 1 show the BAMVALS leads to a clear improvement of the RMSE compared to the MVALS: The high RMSE of the MVALS is a result of the overparametrization, the zero-mean prior given in Equation (18) counteracts the overfitting and leads to the expected regularizing effect. Thus, the results confirm the hypothesis of Batselier et al. (2017). The additional computational steps for the BAMVALS, corresponding to Equations (19) and (20), lead to a small increase in the training time up to 11 s.

5.3 Choosing λ and computing Bayesian simulations

Next, we conduct a more detailed investigation of the interaction between the chosen rank R and the hyperparameter $1/\lambda$. We apply Algorithm 1 on the data set and vary both R and $1/\lambda$ while keeping all other hyperparameters fixed. The results are shown in Figure 4 The number of parameters that need to be estimated grows with an increasing R. It follows that for larger R we also need to emphasise the zero mean prior. That corresponds to setting a higher value for $1/\lambda$. If $1/\lambda$ is chosen too high, it results in underfitting, and the RMSE increases again. Last, we apply Algorithm 1 to make predictions with Bayesian uncertainty bounds, as shown in Figure 5. The uncertainty bounds are calculated with Equation 16.

We compare the results of Table 1 and Figure 4 with existing literature on Volterra models for the Cascaded Nonlinear Benchmark, and we see a clear improvement in the runtime with 27 seconds compared to 6.5 hours (Birpoutsoukis et al., 2018) and 2 minutes (Dalla Libera et al., 2021). With a RMSE = 0.54 (Birpoutsoukis et al., 2018) and RMSE = 0.48 (Dalla Libera et al., 2021), it also becomes evident that the exponentially decaying priors for the Volterra kernels of Birpoutsoukis et al. (2018); Dalla Libera et al. (2021) yield better results with respect to the RMSE compared to the zero-mean prior obtaining

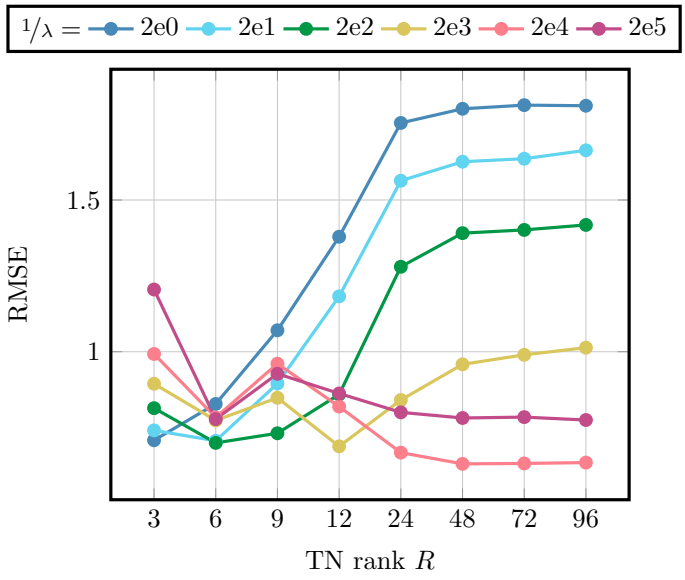

Fig. 4. RMSE for BAMVALS ($D = 2$, $M = 95$, $\sigma^2 = 0.0165$) for different values of R and $1/\lambda$ on validation data, averaged over 10 iterations. The number of parameters to be estimated increases with a larger R, thus selecting a higher $1/\lambda$ with an increasing R results in a smaller RMSE.

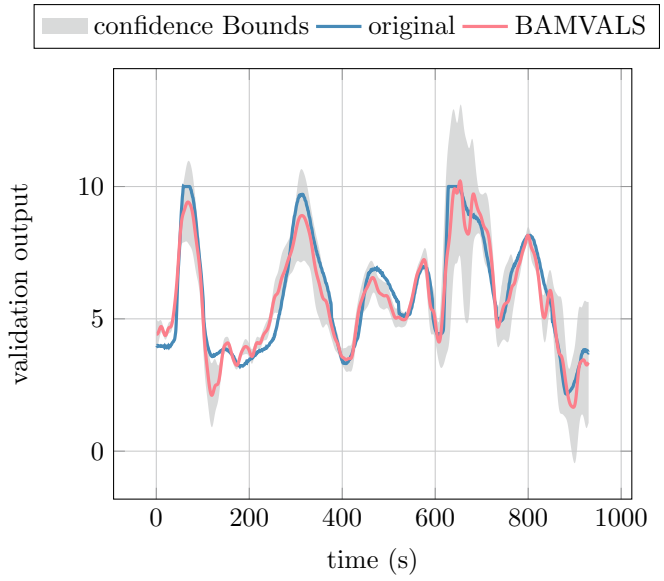

Fig. 5. True (blue) and mean of simulated (red) system output (BAMVALS: $M = 95$, $R = 48$, $1/\lambda = 20,000$, $\sigma^2 = 0.0165$), confidence bounds are depicted with one standard deviation of the mean.

RMSE = 0.63 or one core the low-rank TN representation in Equation (18).

6. CONCLUSION AND FUTURE WORK

With this paper, we develop an algorithm that allows to include prior knowledge into the MIMO Volterra TN. Furthermore, with the selected prior, we can confirm the hypothesis of (Batselier et al., 2017) that regularization in the MVALS counteracts overfitting. Last, we provide a framework for computing simulations with Bayesian uncertainty bounds. A clear future direction of the work is to derive an exponentially decaying prior for the MIMO Volterra TN, as an improvement of the RMSE is expected. A second future direction of the work is to adapt the Bayesian ALS of Menzen et al. (2022) and treat all cores of the TN as random variables. Third, in future work, we will include hyperparameter optimization or investigate the influence of the BAMVALS algorithm's hyperparameters, such as the core's position containing the prior knowledge in larger TNs or the impact of σ^2 on the confidence bounds.

REFERENCES

Batselier, K. (2022). Low-rank tensor decompositions for nonlinear system identification: A tutorial with examples. *IEEE Control Systems Magazine*, 42(1), 54–74.

Batselier, K., Chen, Z., and Wong, N. (2017). Tensor network alternating linear scheme for mimo volterra system identification. *Automatica*, 84, 26–35.

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

Chen, T., Ohlsson, H., and Ljung, L. (2011). On the estimation of transfer functions, regularizations and gaussian processes–revisited. *IFAC Proceedings Volumes*, 44(1), 2303–2308.

Dalla Libera, A., Carli, R., and Pillonetto, G. (2021). Kernel-based methods for volterra series identification. *Automatica*, 129, 109686.

Favier, G., Kibangou, A.Y., and Bouilloc, T. (2012). Nonlinear system modeling and identification using volterra-parafac models. *International Journal of Adaptive Control and Signal Processing*, 26(1), 30–53.

Kolda, T.G. and Bader, B.W. (2009). Tensor decompositions and applications. *SIAM review*, 51(3), 455–500.

Menzen, C., Kok, M., and Batselier, K. (2022). Alternating linear scheme in a bayesian framework for low-rank tensor approximation. *SIAM Journal on Scientific Computing*, 44(3), A1116–A1144.

Miao, P., Qi, C., Jin, Y., Song, K., and Yu, T. (2019). Kernels pruning for volterra digital predistortion using sparse bayesian learning. In *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, 1–6. IEEE.

Oseledets, I.V. (2010). Approximation of 2ˆd\times2ˆd matrices using tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(4), 2130–2145.

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

Pillonetto, G. and De Nicolao, G. (2010). A new kernel-based approach for linear system identification. *Automatica*, 46(1), 81–93.

Schoukens, M., Mattson, P., Wigren, T., and Noel, J.P. (2016). Cascaded tanks benchmark combining soft and hard nonlinearities. In *Workshop on nonlinear system identification benchmarks*, 20–23.