Nonlinear system identification with regularized Tensor Network B-splines

Karagoz, Ridvan; Batselier, Kim

DOI
10.1016/j.automatica.2020.109300

Publication date
2020

Document Version
Final published version

Published in
Automatica

Automatica 122 (2020) 109300

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Nonlinear system identification with regularized Tensor Network B-splines☆

Ridvan Karagoz, Kim Batselier *

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

ARTICLE INFO

Article history:
Received 17 March 2020
Received in revised form 22 June 2020
Accepted 28 August 2020
Available online xxxx

Keywords:
Nonlinear system identification
NARX
B-splines
Tensor network
Curse of dimensionality

ABSTRACT

This article introduces the Tensor Network B-spline (TNBS) model for the regularized identification of nonlinear systems using a nonlinear autoregressive exogenous (NARX) approach. Tensor network theory is used to alleviate the curse of dimensionality of multivariate B-splines by representing the high-dimensional weight tensor as a low-rank approximation. An iterative algorithm based on the alternating linear scheme is developed to directly estimate the low-rank tensor network approximation, removing the need to ever explicitly construct the exponentially large weight tensor. This reduces the computational and storage complexity significantly, allowing the identification of NARX systems with a large number of inputs and lags. The proposed algorithm is numerically stable, robust to noise, guaranteed to monotonically converge, and allows the straightforward incorporation of regularization. The TNBS-NARX model is validated through the identification of the cascaded watertank benchmark nonlinear system, on which it achieves state-of-the-art performance while identifying a 16-dimensional B-spline surface in 4 s on a standard desktop computer. An open-source MATLAB implementation is available on GitHub.

1. Introduction

B-splines are basis functions for the spline function space (De Boor, 1976), making them an attractive choice for approximating smooth continuous functions. B-spline curves offer great control of their flexibility and smoothness and are, compared to polynomials, numerically stable. Moreover, they evidence advantages w.r.t. neural networks in tuning the weights thanks to the linearity in the parameters: this allows to apply fast offline and online algorithms such as recursive least squares. For these reasons and more, B-splines have had numerous applications in system identification (Csurcsia, Schoukens, Kollár, & Lataire, 2014; Folgheraiter, 2016; Lightbody, O'Reilly, Irwin, McCormick, et al., 1997; Lin, Reay, Williams, & He, 2007; Mirea, 2014; dos Santos Coelho & Pessôa, 2009; Yiu, Wang, Teo, & Tsoi, 2001) and control (Cong & Song, 2000; Hong & Chen, 2012; Reay, 2003; Zhang, Zhao, Guo, Li, & Deng, 2017). The generalization of B-splines to multiple dimensions is done through tensor products of their univariate basis functions. The number of basis functions and weights that define a multivariate B-spline surface, therefore, increase exponentially with the number of dimensions, i.e. B-splines suffer from the curse of dimensionality. Previous attempts to avoid this limitation include strategies such as dimensionality reduction, ANOVA decompositions and hierarchical structures (Brown, Bossley, Mills, & Harris, 1995). The most effective method, i.e. hierarchical B-splines, relies on sparse grids (Zenger, 1991) and reduces the storage complexity from $\mathcal{O}(k^d)$ to $\mathcal{O}(k\log(k)^{d-1})$ (Garcke et al., 2006). This is still exponential in the number of dimensions d. A recently emerging way to alleviate the curse of dimensionality is through the concept of tensor networks. Originally developed in the context of quantum physics, tensor networks efficiently represent high-dimensional tensors as a set of sparsely interconnected low-order tensors (Cichocki et al., 2016). Combined with tensor algebra, tensor network structures can greatly decrease the computational complexity in nonlinear system identification (Batselier, Chen, & Wong, 2017a, 2017b; Batselier, Ko, & Wong, 2018). Due to their multilinear nature, multivariate B-splines easily admit a tensor network representation, which we call the Tensor Network B-splines (TNBS) model. Algorithms for optimization in the tensor network format make it possible to fit multivariate B-spline surfaces onto high-dimensional data by directly finding a low-rank tensor network approximation of the weight tensor, thereby overcoming the curse of dimensionality. This broadens the applicability of multivariate B-splines to high-dimensional problems that often occur in system identification and control. One particularly well-suited application of Tensor Network B-splines is

☆ The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Antonio Vicino under the direction of Editor Torsten Söderström.

* Corresponding author.
E-mail address: K.Batselier@tudelft.nl (K. Batselier).

https://doi.org/10.1016/j.automatica.2020.109300
0005-1098/

black-box nonlinear system identification. The Nonlinear Autoregressive eXogenous (NARX) model (Leontaritis & Billings, 1985) is able to represent a wide range of nonlinear systems and is useful when knowledge about the model structure of the system is limited. For the single-input single-output (SISO) case, the discrete-time NARX model is expressed by the following nonlinear difference equation:

$$y_n = f(y_{n-1}, y_{n-2}, \ldots, u_n, u_{n-1}, u_{n-2}, \ldots) + \varepsilon_n. \tag{1}$$

The function f is an unknown nonlinear mapping and u_n and y_n are the input and output samples at time step n. The error ε_n is assumed to be Gaussian white noise. The most common models used in approximating f are polynomials or neural networks (Billings, 2013). The applicability of polynomial NARX is, however, often limited to weakly nonlinear systems due to computational complexity. Neural networks, on the other hand, require a lot of data to generalize well and can be time consuming to train. Under the reasonable assumption that f is sufficiently smooth, the Tensor Network B-splines model is a suitable candidate to approximate the function from observed input and output data. The contributions of this paper are:

- Lift the curse of dimensionality of B-splines with Tensor Network theory.
- Present a regularized TNBS-NARX system identification algorithm.

The paper is structured as follows. Section 2 introduces relevant tensor and B-spline theory. Section 3 presents the TNBS model, the regularization technique and the NARX identification algorithm. Section 4 validates the TNBS-NARX approach through numerical experiments on synthetic and benchmark datasets. Section 5 concludes this paper and lists some possible extensions.

2. Preliminaries

This section provides the basic terminology and definitions for tensors and tensor decompositions, followed by an introduction to B-splines. Most of the introduced tensor network definitions are based on (Cichocki, 2014; Kolda & Bader, 2009; Oseledets, 2011; Penrose, 1971). A comprehensive treatment of B-splines is given in the book by de Boor (1978).

2.1. Tensor basics

A tensor is a multidimensional array of real numerical values, e.g. $\mathcal{A} \in \mathbb{R}^{k_1 \times k_2 \times \cdots \times k_d}$. Tensors can thus be considered generalizations of vectors and matrices. The order d of the tensor is the number of dimensions of the array. Unless stated otherwise, subscript indices indicate a single element of a tensor, e.g. $a = \mathcal{A}_{i_1,i_2,\ldots,i_d}$. The size of each dimension is indicated by $k_p, p \in \{1, 2, \ldots, d\}$, such that $i_p \in \{1, 2, \ldots, k_p\}$. In this paper, scalars are denoted by lowercase letters (a), vectors are denoted by bold lowercase letters ($\boldsymbol{a}$), matrices are denoted by bold uppercase letters ($\boldsymbol{A}$) and higher-order tensors are denoted by calligraphic letters ($\mathcal{A}$).

A convenient way of expressing tensors and their operations is using the graphical notation introduced by Roger Penrose in 1972 (Penrose, 1971). Fig. 1 shows the representation of a scalar, vector, matrix and third-order tensor using this notation. Every node represents a tensor, the edges represent the indices and the number of edges, therefore, corresponds to its order. The vectorization of a tensor $\mathcal{A} \in \mathbb{R}^{k_1 \times k_2 \times \cdots \times k_d}$ is the reordering of its elements into a column vector, denoted by $\boldsymbol{vec}(\mathcal{A}) = \boldsymbol{a} \in \mathbb{R}^{k_1 k_2 \cdots k_d}$. The elements of $\boldsymbol{a}$ are denoted as:

$$\boldsymbol{a}_{i_1+(i_2-1)k_1+\cdots+(i_d-1)k_1k_2\ldots k_{d-1}} = \mathcal{A}_{i_1,i_2,\ldots,i_d}.$$

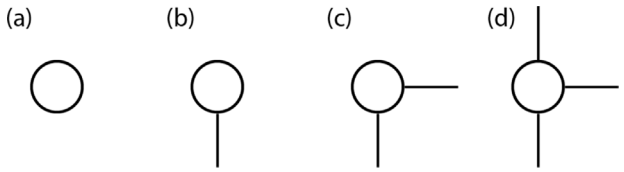

Fig. 1. Graphical notation of a (a) scalar, (b) vector, (c) matrix and (d) third-order tensor.

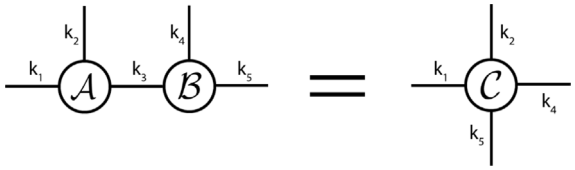

Fig. 2. Tensor contraction in graphical notation.

A tensor $\mathcal{T} \in \mathbb{R}^{k_1 \times k_2 \times \cdots \times k_d}$ is of rank one if it can be decomposed into the outer product of d vectors $\boldsymbol{b}^{(p)} \in \mathbb{R}^{k_p}$, e.g:

$$\mathcal{T} = \boldsymbol{b}^{(1)} \circ \boldsymbol{b}^{(2)} \circ \cdots \circ \boldsymbol{b}^{(d)},$$

where $\circ$ denotes the outer product operation. The most essential operation in tensor algebra is contraction, which is the summing of elements over equal-sized indices. Given the tensors $\mathcal{A} \in \mathbb{R}^{k_1 \times k_2 \times k_3}$ and $\mathcal{B} \in \mathbb{R}^{k_3 \times k_4 \times k_5}$, contracting the index i_3 results in a tensor $\mathcal{A} \times_3^1 \mathcal{B} = \mathcal{C} \in \mathbb{R}^{k_1 \times k_2 \times k_4 \times k_5}$ whose elements are given by:

$$\mathcal{C}_{i_1,i_2,i_4,i_5} = \sum_{i_3} \mathcal{A}_{i_1,i_2,i_3}\, \mathcal{B}_{i_3,i_4,i_5}. \tag{2}$$

Contraction is indicated by the left-associative $\times_n^m$ operator (Cichocki et al., 2016), where n and m indicate the position of the indices of the first and second tensor respectively. In the graphical notation, contraction is indicated by connecting corresponding edges, as illustrated for (2) in Fig. 2. A useful equation (Batselier et al., 2017a) that relates contraction of a d-dimensional tensor with d matrices to a linear operation is the following:

$$\boldsymbol{vec}\left(\mathcal{A} \times_1^2 \boldsymbol{C}^{(1)} \times_2^2 \cdots \times_d^2 \boldsymbol{C}^{(d)}\right) = \left(\boldsymbol{C}^{(d)} \otimes \cdots \otimes \boldsymbol{C}^{(1)}\right) \boldsymbol{vec}(\mathcal{A}), \tag{3}$$

where $\otimes$ denotes the Kronecker product. The outer product operation is a special case of contraction where the contracted indices have singleton dimensions. The outer product is depicted in the graphical notation by a dashed line connecting two nodes. The inner product between two equal-sized tensors is the sum of their entry-wise products, equivalent to contraction of the tensors over all pairs of indices. Given two tensors $\mathcal{A} \in \mathbb{R}^{k_1 \times k_2 \times k_3}$ and $\mathcal{B} \in \mathbb{R}^{k_1 \times k_2 \times k_3}$, their inner product is given by:

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1,i_2,i_3} \mathcal{A}_{i_1,i_2,i_3} \mathcal{B}_{i_1,i_2,i_3} = \boldsymbol{vec}(\mathcal{A})^T \boldsymbol{vec}(\mathcal{B}).$$

The Frobenius norm of a tensor is defined as the square root of the sum of squares of its entries:

$$\|\mathcal{A}\|_2 = \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}.$$

2.2. Tensor trains

The tensor train (TT) decomposition is a widely used tensor network format, popular for its low parametric format and the numerical stability of related optimization algorithms (Oseledets, 2011). A tensor train expresses a tensor $\mathcal{W} \in \mathbb{R}^{k_1 \times k_2 \times \cdots \times k_d}$ of order d in terms of third-order tensors $\mathcal{G}_{\mathcal{W}}^{(p)} \in \mathbb{R}^{r_{p-1} \times k_p \times r_p}$, also known

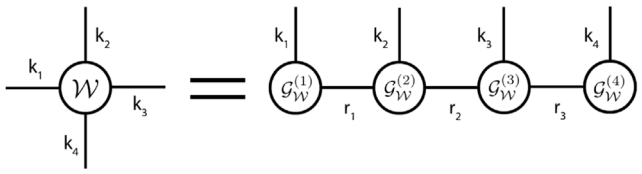

Fig. 3. Graphical notation of the tensor train decomposition for a fourth-order tensor.

as the TT-cores. Fig. 3 shows the TT-decomposition of a four-dimensional tensor in graphical notation. The dimensions of the contracted indices, r_p, are called TT-ranks. The first and last TT-ranks, r_0 and r_d, are by definition equal to one. Keeping in mind that the $\times_n^m$-mode product operator is left-associative, the tensor train in Fig. 3 can be expressed as:

$$\mathcal{W} = \mathcal{G}_{\mathcal{W}}^{(1)} \times_2^1 \mathcal{G}_{\mathcal{W}}^{(2)} \times_3^1 \mathcal{G}_{\mathcal{W}}^{(3)} \times_4^1 \mathcal{G}_{\mathcal{W}}^{(4)}. \tag{4}$$

There exists a set of TT-ranks $r_p = R_p$ for which the decomposition is exact. When $r_p < R_p$, the tensor train represents an approximation of the original tensor. The lower the TT-ranks, the less accurate the decomposition, but the better the compression. When all r_p and dimensions k_p are equal, the storage complexity of the tensor train representation is $\mathcal{O}(kdr^2)$. A TT-decomposition with low TT-ranks can thus significantly reduce the memory footprint of high-dimensional data. For a prescribed set of TT-ranks or a prescribed accuracy, the TT-decomposition of a tensor can be computed with the TT-SVD (Oseledets, 2011) or the TT-Cross (Oseledets & Tyrtyshnikov, 2010) algorithm. An important notion for TT-cores is orthogonality. A TT-core $\mathcal{G}_{\mathcal{W}}^{(p)}$ is left-orthogonal if it can be reshaped (Batselier et al., 2017a) into a matrix $\boldsymbol{G}^{(p)} \in \mathbb{R}^{r_{p-1}k_p \times r_p}$ for which:

$$\boldsymbol{G}^{(p)T}\boldsymbol{G}^{(p)} = \boldsymbol{I}.$$

Likewise, $\mathcal{G}_{\mathcal{W}}^{(p)}$ is right-orthogonal if it can be reshaped into a matrix $\boldsymbol{G}^{(p)} \in \mathbb{R}^{r_{p-1} \times k_p r_p}$ for which:

$$\boldsymbol{G}^{(p)}\boldsymbol{G}^{(p)T} = \boldsymbol{I}.$$

A tensor train is in site-k-mixed-canonical form (Schollwöck, 2011) when for its TT-cores the following applies:

$$\mathcal{G}_{\mathcal{W}}^{(p)} = \begin{cases} \text{left-orthogonal}, & 1 \leq p \leq k-1 \\ \text{right-orthogonal}, & k+1 \leq p \leq d. \end{cases} \tag{5}$$

For a site-k-mixed-canonical tensor train holds that its norm is contained in the kth TT-core, i.e.:

$$\|\mathcal{W}\|_2 = \|\mathcal{G}_{\mathcal{W}}^{(k)}\|_2.$$

2.3. B-splines

A univariate spline S is a piecewise polynomial function that maps values from an interval $[a, b]$ to the set of real numbers, e.g. $S : [a, b] \in \mathbb{R} \to \mathbb{R}$. Any spline of degree ρ can be expressed as a unique linear combination of B-splines of the same degree:

$$S(x) = \sum_{i=1}^{k} B_i(x) w_i = \boldsymbol{b}^T \boldsymbol{w} \tag{6}$$

$$= \begin{bmatrix} B_{1,\rho}(x) & B_{2,\rho}(x) & \cdots & B_{k,\rho}(x) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix}. \tag{7}$$

The B-spline basis functions $B_{i,\rho}(x)$ are defined by the knot sequence and degree ρ, and they are contained in the basis vector $\boldsymbol{b}$. A knot sequence $\boldsymbol{t} = \{t_0, t_1, \ldots, t_{m-1}, t_m\}$ is defined as a non-decreasing and finite sequence of real numbers that define the partitioning of the domain [a,b], i.e. $a = t_0 \leq t_1 \leq \cdots \leq t_{m-1} \leq t_m = b$, such that $S(x)$ is a polynomial on any interval $[t_i, t_{i+1}]$. B-spline basis functions of arbitrary degree ρ can be recursively constructed by means of the Cox–de Boor formula (de Boor, 1978):

$$B_{i,0}(x) = \begin{cases} 1 & \text{if} \quad t_{i-1} \leq x < t_i \\ 0 & \text{otherwise} \end{cases}, \tag{8}$$

$$B_{i,\rho+1}(x) = \frac{x - t_{i-1}}{t_{i+\rho-1} - t_{i-1}} B_{i,\rho}(x) + \frac{t_{i+\rho} - x}{t_{i+\rho} - t_i} B_{i+1,\rho}(x).$$

The number of B-spline basis functions k, therefore, relates to the degree ρ and number of knots $m + 1$ by $k = m - \rho$. If the knots are equidistantly distributed over the domain of the spline, the spline is called uniform. If the uniform knot sequence is also a subset of $\mathbb{Z}$, i.e. a sequence of integers, the spline is referred to as a cardinal spline (Schoenberg, 1973). In this article, all knot sequences will be considered uniform, as they allow for efficient evaluation of $\boldsymbol{b}$ using a matrix expression (Qin, 2000) instead of (8). Fig. 4 illustrates B-splines of degree 1 to 3 on the cardinal knot sequence $\boldsymbol{t} = \{0, 1, 2, 3, 4, 5\}$. The dashed purple lines represent the sum of the basis functions. The shape of a B-spline curve $S(x)$ is only fully adjustable within its natural domain $\mathcal{D}_n = [t_\rho, t_{m-\rho}]$, because the sum of the B-spline basis functions at any point within this domain equals one. It is desirable to have full control over the shape of the B-spline curve over the whole range of data samples. The knot sequences in this article will be chosen such that $\mathcal{D}_n$ coincides with the unit interval [0, 1].

2.4. Multivariate B-splines

B-splines generalize to multiple input dimensions through tensor products of univariate basis functions. One can construct a d-dimensional spline S as a linear combination of multivariate B-splines:

$$\begin{aligned} &S(x_1, x_2, \ldots, x_d) \\ &= \sum_{i_1=1}^{k_1} \sum_{i_2=1}^{k_2} \cdots \sum_{i_d=1}^{k_d} B_{i_1}(x_1) B_{i_2}(x_2) \cdots B_{i_d}(x_d) \mathcal{W}_{i_1 i_2 \cdots i_d} \\ &= \langle \mathcal{B}, \mathcal{W} \rangle . \end{aligned} \tag{9}$$

For notational convenience, we omitted the degrees ρ. The B-spline tensor $\mathcal{B}$ contains the multivariate basis functions and is defined as:

$$\mathcal{B} = \boldsymbol{b}^{(1)} \circ \boldsymbol{b}^{(2)} \circ \cdots \circ \boldsymbol{b}^{(d)},$$

where $\boldsymbol{b}^{(p)}$ is the univariate basis vector of the pth input variable, i.e.

$$\boldsymbol{b}^{(p)} = \begin{bmatrix} B_{1,\rho}(x_p) & B_{2,\rho}(x_p) & \cdots & B_{k_p,\rho}(x_p) \end{bmatrix}^T. \tag{10}$$

We will assume equal knots and degree for each dimension, hence $k_p = k$, $\forall p$. The representation of B-spline surfaces in (9) is severely limited by the exponential increase in the number of basis functions and weights, $\mathcal{O}(k^d)$.

3. Tensor Network B-splines

For our purposes, the input variables x_p are the lagged inputs and outputs of (1). For a large number of lags or inputs, it can therefore quickly become computationally infeasible to store or operate on the tensors $\mathcal{B}$ and $\mathcal{W}$. Using tensor network theory, the multivariate B-spline surface can be represented in a low-parametric format. In this section, we derive the TNBS model and use it to approximate the function f in (1) from observed input and output data.

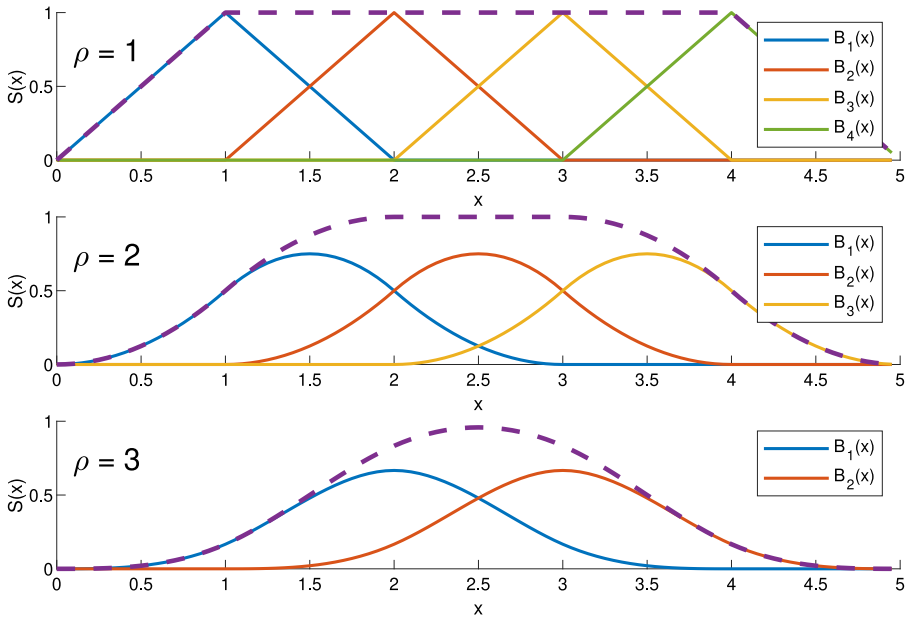

Fig. 4. Cardinal B-splines of degrees 1 to 3. The dashed purple lines represent the sum of the B-splines.

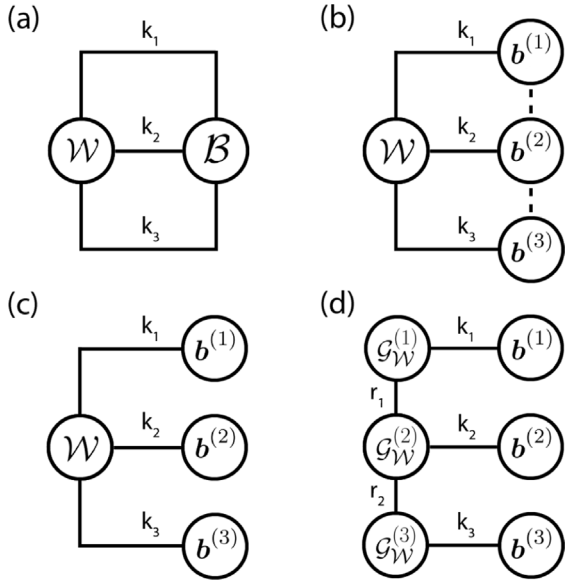

Fig. 5. Derivation of the Tensor Network B-splines model of order 3.

3.1. Model structure

We illustrate the model structure using a three-dimensional Tensor Network B-spline surface as an example, which is derived as follows:

$$S(x_1, x_2, x_3)$$
$$= \langle \mathcal{B}, \mathcal{W} \rangle \tag{11}$$
$$= \mathcal{W} \times_1^1 \boldsymbol{b}^{(1)} \times_2^1 \boldsymbol{b}^{(2)} \times_3^1 \boldsymbol{b}^{(3)} \tag{12}$$
$$= (\mathcal{G}_{\mathcal{W}}^{(1)} \times_2^1 \boldsymbol{b}^{(1)})(\mathcal{G}_{\mathcal{W}}^{(1)} \times_2^1 \boldsymbol{b}^{(2)})(\mathcal{G}_{\mathcal{W}}^{(1)} \times_2^1 \boldsymbol{b}^{(3)}). \tag{13}$$

Fig. 5 will be used as a visual reference to walk through these equations. Given the weight tensor $\mathcal{W} \in \mathbb{R}^{k_1 \times k_2 \times k_3}$ and B-spline tensor $\mathcal{B} \in \mathbb{R}^{k_1 \times k_2 \times k_3}$ in (11), their inner product is equal to the contraction over all pairs of indices, as seen in Fig. 5a. As $\mathcal{B}$ is a rank one tensor, it can be decomposed into the outer product of three B-spline vectors $\boldsymbol{b}^{(p)}$ (Fig. 5b). The outer product operation is a special case of contraction where the contracted indices have singleton dimensions. Singleton contractions that close a loop in a tensor network are redundant, and hence omitted in Fig. 5c. Now $S(x_1, x_2, x_3)$ in (12) is simply the contraction of $\mathcal{W}$ with the B-spline basis vectors. Finally, $\mathcal{W}$ is decomposed into a tensor train in Fig. 5d. A point (x_1, x_2, x_3) on the TNBS surface in (13) is evaluated by constructing the B-spline vectors $\boldsymbol{b}^{(p)}$, contracting them with the corresponding tensor train cores and finally multiplying the sequence of resulting matrices. Due to the constraints, $r_0 = r_d = 1$, and this results in a scalar output. Extending the TNBS model to l outputs can be realized by removing one of these constraints, e.g. $r_0 = l$. In general, a d-dimensional TNBS surface is represented by:

$$S(x_1, x_2, \ldots, x_d) = \prod_{p=1}^{d} (\mathcal{G}_{\mathcal{W}}^{(p)} \times_2^1 \boldsymbol{b}^{(p)}). \tag{14}$$

3.2. Identification algorithm

We illustrate, without loss of generality, the proposed identification algorithm by means of the following example. Suppose we have the following NARX system model:

$$y_n = f(u_n, y_{n-1}, u_{n-1}) + \varepsilon_n. \tag{15}$$

We want to identify this model from a set of observed input and output data $\{(y_n, u_n)\}_{n=1}^{N}$. We approximate the function f with the three-dimensional TNBS from Fig. 5d, by minimizing the least-squared cost function:

$$\min_{\mathcal{W}} \ \|\boldsymbol{y} - \boldsymbol{s}\|_2^2 \tag{16}$$
$$\text{s.t. TT-rank}(\mathcal{W}) = (r_1, r_2),$$

where

$$\boldsymbol{y} = \begin{bmatrix} y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix}, \qquad \boldsymbol{s} = \begin{bmatrix} f(u_2, y_1, u_1) \\ f(u_3, y_2, u_2) \\ \vdots \\ f(u_N, y_{N-1}, u_{N-1}) \end{bmatrix}.$$

To solve (16) directly for the TT-cores, we use the alternating linear scheme (ALS) (Holtz, Rohwedder, & Schneider, 2012). The

TT-ranks are chosen beforehand and the TT-cores are initialized randomly. ALS then iteratively optimizes one tensor core at a time while holding the others fixed. Optimizing one core is equal to solving a small linear subsystem. Suppose we wish to update the second core from Fig. 5d. The idea is to contract everything in the network up until the nodes adjacent to $\mathcal{G}_{\mathcal{W}}^{(2)}$ (Fig. 6a), whereupon (3) is used to rewrite the network as an inner product of two vectors (Fig. 6b):

$$\begin{aligned} y_n &= \mathcal{G}_{\mathcal{W}}^{(2)} \times_1^1 \boldsymbol{v}_<^{(2)} \times_2^1 \boldsymbol{b}^{(2)} \times_3^2 \boldsymbol{v}_>^{(2)} \\ &= \mathcal{G}_{\mathcal{W}}^{(2)} \times_1^2 \boldsymbol{v}_<^{(2)T} \times_2^2 \boldsymbol{b}^{(2)T} \times_3^2 \boldsymbol{v}_>^{(2)} \\ &= \left(\boldsymbol{v}_>^{(2)T} \otimes \boldsymbol{b}^{(2)T} \otimes \boldsymbol{v}_<^{(2)}\right) \boldsymbol{vec}\left(\mathcal{G}_{\mathcal{W}}^{(2)}\right) \\ &= \boldsymbol{a}^{(2)T} \boldsymbol{g}^{(2)}. \end{aligned} \tag{17}$$

More generally, rewriting (14) for the nth data sample as a linear function of the elements of the pth core gives:

$$y_n = \left(\boldsymbol{v}_{>,n}^{(p)T} \otimes \boldsymbol{b}_n^{(p)T} \otimes \boldsymbol{v}_{<,n}^{(p)}\right) \boldsymbol{vec}\left(\mathcal{G}_{\mathcal{W}}^{(p)}\right), \tag{18}$$

where

$$\boldsymbol{v}_{<,n}^{(p)} = \prod_{j=1}^{p-1} (\mathcal{G}_{\mathcal{W}}^{(j)} \times_2^1 \boldsymbol{b}_n^{(j)}) \in \mathbb{R}^{1 \times r_{p-1}}$$

$$\boldsymbol{v}_{>,n}^{(p)} = \prod_{j=p+1}^{d} (\mathcal{G}_{\mathcal{W}}^{(j)} \times_2^1 \boldsymbol{b}_n^{(j)}) \in \mathbb{R}^{r_p}$$

for $2 \leq p \leq d-1$, and $\boldsymbol{v}_{<,n}^{(1)} = \boldsymbol{v}_{>,n}^{(d)} = 1$. Computing (18) for all N data samples results in a system of linear equations. The subproblem for updating the pth core thus becomes:

$$\min_{\boldsymbol{g}^{(p)}} \left\| \boldsymbol{y} - \boldsymbol{A}^{(p)} \boldsymbol{g}^{(p)} \right\|_2^2, \tag{19}$$

where

$$\boldsymbol{A}^{(p)} = \begin{bmatrix} \boldsymbol{v}_{>,1}^{(p)T} \otimes \boldsymbol{b}_1^{(p)T} \otimes \boldsymbol{v}_{<,1}^{(p)} \\ \boldsymbol{v}_{>,2}^{(p)T} \otimes \boldsymbol{b}_2^{(p)T} \otimes \boldsymbol{v}_{<,2}^{(p)} \\ \vdots \\ \boldsymbol{v}_{>,N}^{(p)T} \otimes \boldsymbol{b}_N^{(p)T} \otimes \boldsymbol{v}_{<,N}^{(p)} \end{bmatrix}, \quad \boldsymbol{g}^{(p)} = \boldsymbol{vec}\left(\mathcal{G}_{\mathcal{W}}^{(p)}\right). \tag{20}$$

The optimum is found by solving the normal equation:

$$\left(\boldsymbol{A}^{(p)T} \boldsymbol{A}^{(p)}\right) \boldsymbol{g}^{(p)} = \boldsymbol{A}^{(p)T} \boldsymbol{y}. \tag{21}$$

Reshaping (Batselier et al., 2017a) $\boldsymbol{g}^{(p)}$ back into a third-order tensor results in the updated core $\mathcal{G}_{\mathcal{W}}^{(p)}$. The ALS algorithm sweeps back and forth, iterating from the first to the last core and back, until convergence. At each iteration, (21) is solved for $\boldsymbol{g}^{(p)}$. Numerical stability is ensured by keeping the tensor train in site-p-mixed-canonical form through an additional orthogonalization step. To illustrate this, consider again the TNBS in Fig. 5d. Assume that we are iterating from left to right and the tensor train is in site-2-mixed-canonical form. After solving $\boldsymbol{g}^{(p)}$ it is reshaped into a matrix $\boldsymbol{G}^{(p)} \in \mathbb{R}^{r_{p-1} k_p \times r_p}$, which is then decomposed through a QR decomposition (Francis, 1961). The tensor network is now in the form of Fig. 7. Finally, Q is reshaped back into a third-order left-orthogonal tensor $\boldsymbol{G}^{(2)}$ and R is contracted with the next core. The tensor train is now in site-3-mixed-canonical form, and the next iteration starts. More details about the orthogonalization step are given in Holtz et al. (2012). The optimization with ALS exhibits local linear convergence under the assumption that the TT-ranks are correctly estimated (Rohwedder & Uschmajew, 2013). In practice, ALS convergences monotonically, so a possible stopping criterion is:

$$\left\| J_h^{(1)} - J_{h+1}^{(1)} \right\|_2 \leq \epsilon, \tag{22}$$

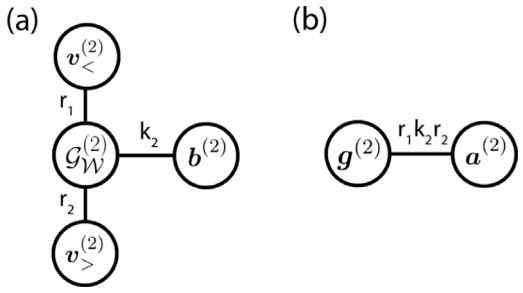

Fig. 6. The tensor network written as a vector inner product.

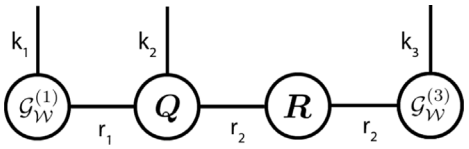

Fig. 7. QR decomposition of the second core during a left to right sweep.

where $J_h^{(1)}$ is the cost of the objective function in (19) during the first core update of the hth sweep. A modified version of ALS method, MALS (Holtz et al., 2012), updates two cores simultaneously and is computationally more expensive, but is able to adaptively determine the optimal TT-ranks for a specified accuracy. Another adaptive method is the tensor network Kalman filter (Batselier et al., 2017b), which can be used for online optimization of the cores.

3.3. Regularization

In addition to decreasing computational burden, the TT-rank constraints serve as a regularization mechanism. This regularization is however insufficient for high-dimensional B-splines, as the volume of the domain of the TNBS increases exponentially. The available estimation data becomes sparse and scattered, which can lead to an ill-posed optimization problem. B-spline curves inherently possess the ability to regularize by adjustment of their degree or knot placement. The choice of knots has been a subject of much research (Eubank, 1999), but due to lack of an attractive all-purpose scheme, we opt for a non-parametric approach known as P-splines (Eilers & Marx, 1996). P-splines induce smoothness by combining uniform B-splines with a discrete penalty placed on the α-th difference between adjacent weights. For univariate splines, the following penalty function is added to the cost function:

$$R(\boldsymbol{w}) = \|\boldsymbol{D}_\alpha \boldsymbol{w}\|_2^2. \tag{23}$$

The matrix $\boldsymbol{D}_\alpha \in \mathcal{R}^{(k+1-\alpha)\times(k+1)}$ is the α-th order difference matrix such that $\boldsymbol{D}_\alpha \boldsymbol{w} = \Delta^\alpha \boldsymbol{w}$ results in a vector of α-th order differences of $\boldsymbol{w}$. This matrix can be constructed by using the difference operator α times consecutively on the identity matrix. For $\alpha = 0$ this is equal to Tikhonov regularization and for $\alpha = 1$ we get Total Variation regularization. For example, given are a weight vector and the first-order difference matrix:

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}, \qquad \boldsymbol{D}_1 = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}.$$

The penalty term then equals:

$$\begin{aligned} \|\boldsymbol{D}_1 \boldsymbol{w}\|_2^2 &= (\boldsymbol{D}_1 \boldsymbol{w})^T (\boldsymbol{D}_1 \boldsymbol{w}) \\ &= \begin{bmatrix} (w_1 - w_2) & (w_2 - w_3) \end{bmatrix} \begin{bmatrix} (w_1 - w_2) \\ (w_2 - w_3) \end{bmatrix} \\ &= (w_1 - w_2)^2 + (w_2 - w_3)^2. \end{aligned}$$

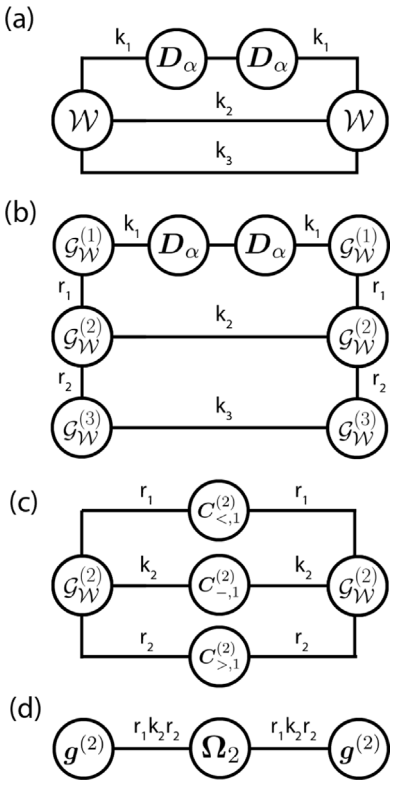

Fig. 8. Derivation of the Tensor Network P-spline penalty.

We wish to extend the penalty in (23) to the TNBS format. Without loss of generality, Fig. 8 visualizes the necessary steps in graphical notation for a three-dimensional B-spline surface. In the multivariate case, the differences in adjacent weights in the weight tensor $\mathcal{W}$ have to be penalized along each dimension individually. This is done by contracting the second index of the difference matrix $\boldsymbol{D}_\alpha$ with the dimension of the weight tensor $\mathcal{W}$ along which the penalty is applied, then taking the norm of the result. For a B-spline curve with d inputs, the penalty on the α-th order differences along the jth dimension is given by:

$$R(\mathcal{W}) = \|\mathcal{W} \times_j^2 \boldsymbol{D}_\alpha\|_2^2 = \left\langle (\mathcal{W} \times_j^2 \boldsymbol{D}_\alpha), (\mathcal{W} \times_j^2 \boldsymbol{D}_\alpha) \right\rangle. \tag{24}$$

This is illustrated in 8a, where the penalty is applied along the first dimension, e.g. $j = 1$. Decomposing $\mathcal{W}$ into a tensor train results in the network depicted in 8b. To write this penalty again as a linear function of the pth core, we contract everything in the network except these cores (Fig. 8c). In this example, $\boldsymbol{C}_{>,1}^{(2)}$ and $\boldsymbol{C}_{-,1}^{(2)}$ are simply identity matrices. Then, using (3), the penalty function can be rewritten in the form of Fig. 8d:

$$R\left(\mathcal{G}_{\mathcal{W}}^{(p)}\right) = \boldsymbol{g}^{(p)T} \boldsymbol{\Omega}_j^{(p)} \boldsymbol{g}^{(p)}, \tag{25}$$

where

$$\boldsymbol{\Omega}_j^{(p)} = \left(\boldsymbol{C}_{>,j}^{(p)} \otimes \boldsymbol{C}_{-,j}^{(p)} \otimes \boldsymbol{C}_{<,j}^{(p)}\right). \tag{26}$$

The matrix $\boldsymbol{\Omega}_j^{(p)}$ in (26) is constructed for every dimension j. Due to the site-p-mixed-canonical form of the tensor train, the contraction of two out of the three matrices $\boldsymbol{C}_{>,j}^{(p)}$, $\boldsymbol{C}_{-,j}^{(p)}$ and $\boldsymbol{C}_{<,j}^{(p)}$ result in identity matrices. This knowledge can be utilized for efficient implementation. Adding the penalties to the cost function results in the following regularized optimization problem:

$$\min_{\mathcal{W}} \quad \|\boldsymbol{y} - \boldsymbol{s}\|_2^2 + \sum_{j=1}^{d} \lambda_j \|\mathcal{W} \times_j^2 \boldsymbol{D}_\alpha\|_2^2 \tag{27}$$
$$\text{s.t. TT-rank}(\mathcal{W}) = (r_1, r_2, \ldots, r_{d-1}).$$

The smoothing parameter $\lambda_j \geq 0$ controls the penalization of the roughness along dimension j. The subproblem for updating the pth core becomes:

$$\min_{\boldsymbol{g}^{(p)}} \quad \|\boldsymbol{y} - \boldsymbol{A}^{(p)}\boldsymbol{g}^{(p)}\|_2^2 + \sum_{j=1}^{d} \lambda_j \boldsymbol{g}^{(p)T} \boldsymbol{\Omega}_j^{(p)} \boldsymbol{g}^{(p)}. \tag{28}$$

The normal equation is then:

$$\left(\boldsymbol{A}^{(p)T}\boldsymbol{A}^{(p)} + \sum_{j=1}^{d} \lambda_j \boldsymbol{\Omega}_j^{(p)}\right) \boldsymbol{g}^{(p)} = \boldsymbol{A}^{(p)T}\boldsymbol{y}. \tag{29}$$

The whole procedure of identifying a TNBS model from measured data is summarized as pseudo-code in Algorithm 1.

Table 1
Computational complexities of significant operations.

Operation	Complexity
Construct $\left\{\boldsymbol{b}_n^{(p)}\right\}_{n=1}^{N}$	$O\left(Nn^2\right)$
Construct $\left\{\boldsymbol{\Omega}_j^{(p)}\right\}_{j=1}^{d}$	$O\left((d+(m-\rho)^4)r^4\right)$
Construct $\boldsymbol{A}^{(p)}$	$O\left(N(m-\rho)r^2\right)$
Solve $\boldsymbol{g}^{(p)}$	$O\left(N(m-\rho)^2r^4 + (m-\rho)^3r^6\right)$
Evaluate f	$O\left((\rho^2+(m-\rho)r^2)d\right)$

Algorithm 1 TNBS-NARX identification

Input: Data $\{(y_n, u_n)\}_{n=1}^{N}$, TT-ranks $\{r_p\}_{p=1}^{d}$, number of knots m, degree ρ, regularization parameters $\{\lambda_j\}_{j=1}^{d}$
Output: TT-cores $\left\{\mathcal{G}_{\mathcal{W}}^{(p)}\right\}_{p=1}^{d}$

1: Initialize random TT-cores
2: Construct $\left\{\left\{\boldsymbol{b}_n^{(p)}\right\}_{n=1}^{N}\right\}_{p=1}^{d}$ from data
3: **while** stopping criteria (22) not satisfied **do**
4: **for** $p = 1, 2, \ldots, d-1$ **do**
5: Construct $\boldsymbol{A}^{(p)}$ (20) and $\left\{\boldsymbol{\Omega}_j^{(p)}\right\}_{j=1}^{d}$ Eq. (26)
6: $\boldsymbol{g}^{(p)} \leftarrow$ Solve (29)
7: $\mathcal{G}_{\mathcal{W}}^{(p)} \leftarrow$ Orthogonalize and reshape $\boldsymbol{g}^{(p)}$
8: **end for**
9: **for** $p = d, d-1, \ldots, 2$ **do**
10: Repeat lines 5-7
11: **end for**
12: **end while**

Table 1 summarizes relevant computational complexities concerning the TNBS-NARX method. While the complexities scale only linearly in the dimensions, it is important to realize that high TT-ranks easily degrade the performance of optimization of the cores. There is, therefore, a tradeoff between accuracy and speed. The number of data samples N also appears linearly in the complexities but may become a bottleneck for large datasets. A modification for this scenario is to use a small random batch of the data when updating $\boldsymbol{g}^{(p)}$. This can speed up estimation time without significant loss of accuracy.

4. Experiments

In this section, we demonstrate the proposed system identification method. The algorithm is implemented in MATLAB and executed on a personal computer with a 4.2 GHz Intel Core i5-7600K processor and 16 GB of random access memory (RAM). An open-source MATLAB implementation can be found at https://github.com/Ridvanz/Tensor-Network-B-splines, which includes demos on three additional benchmark datasets.

4.1. Synthetic dataset

First, we validate the proposed methods through the identification of an artificial nonlinear dynamical system that is exactly representable in the TNBS-NARX format. The lagged inputs and outputs are chosen as $u_{n-\mu}$ and $y_{n-\mu}$ respectively, where $\mu \in (1, 2, 3, 4)$, such that the system equation is of the following form:

$$y_n = f(y_{n-1}, y_{n-2}, y_{n-3}, y_{n-4}, u_{n-1}, u_{n-2}, u_{n-3}, u_{n-4}) \tag{30}$$

The nonlinear mapping f is modeled as an 8-dimensional TNBS. We choose the degree of the B-splines $\rho = 2$ and the number of knots per dimension $m = 6$. A random weight tensor $\mathcal{W}$ of size $(m - \rho)^d = 4^8$ is generated of which the elements equal either $w_{min} = -4$ or $w_{max} = 5$ with equal probability. The generated tensor is decomposed using the TT-SVD algorithm, truncating the TT-ranks to a value of 5 uniformly. The resulting tensor train represents the true weights of our nonlinear system. For the input signal $\boldsymbol{u}$ we generate a random sequence of length 3000, with values uniformly distributed in the range [0, 1]. This sequence is smoothed with a Gaussian window of size 5 to dampen higher frequencies. We initialize the output signal $\boldsymbol{y}$ with 4 zeros and recursively evaluate the next output with (30), until we have a signal of length 3000. The signals are split in an identification set of 2000 samples and a test set of 1000 samples.

We test the performance of our TNBS-NARX identification algorithm with different levels of Gaussian white noise on the estimation data. Noise is only added to the output signal. The variances for the white noise signals are chosen based on the desired signal to noise ratios SNR. The signal powers are determined after subtracting their means. For simplicity, we penalize the second difference ($\alpha = 2$) of the weights equally for each dimension, e.g. $\lambda_p = \lambda$, $\forall p$. The experiment is run using three different values for lambda. All other model parameters are set to the true values of the synthetic model. The TT-cores are estimated using Algorithm 1. For consistency, we simply choose a max number (16) of sweeps as stopping criteria. The root mean squared error (RMSE) is used as the performance metric to evaluate the accuracy on the test set for both prediction and simulation.

$$e_{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

Fig. 9 plots the RMSE of the different experiments as a function of the SNR in dB. The prediction errors are consistently lower than the simulation errors. The effect of the regularization is in line with expectations, i.e. for increasing SNR values, more regularization is needed to avoid overfitting to noise, so larger penalties give better performance. Overall, the TNBS is able to identify the system accurately, even for relatively noisy estimation data.

4.2. Cascaded tanks dataset

The cascaded tanks system is a benchmark dataset for nonlinear system identification. A detailed description of the system and the data is given in Schoukens, Mattson, Wigren, and Noël (2016).

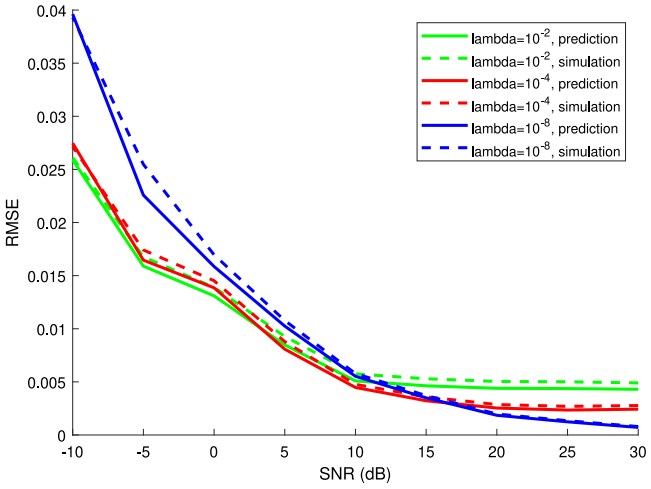

Fig. 9. Prediction and simulation performance on synthetic test set.

The system consists of two tanks, a water reservoir and a pump. The water in the reservoir is pumped in the upper tank, from which it flows to the lower tank through a small opening and then back into the reservoir. The system input u_n is the pump voltage and the system output y_n is the water level of the lower tank. If too much water is pumped into the upper tank it overflows, causing a hard saturation nonlinearity in the system dynamics. The input signals are low-frequency multisine signals. Both the estimation and test set have a length of $N = 1024$ samples. The major challenges of this benchmark are the hard saturation nonlinearity and the relatively small size of the estimation set. The performance metric used is again RMSE.

The original data is first normalized to the interval [0,1]. Both input and output lags are chosen as $u_{n-\mu}$ and $y_{n-\mu}$ respectively, where $\mu \in \{1, 2, 3, 4, 8, 12, 16, 32\}$. The lags were chosen heuristically, based on partial auto-correlations . The large lags are included to capture the relevant slow system dynamics. We choose the degree of the B-splines $\rho = 3$ and the number of knots $m = 7$. We penalize the first-order difference only, i.e. $\alpha = 1$, and set the TT-ranks to 8 uniformly. We choose λ through 3-fold cross-validation on the estimation set. A total of 12 sweeps are performed in the optimization with Algorithm 1. After tuning λ, the full identification set is used to identify the final model, which takes about 2 seconds. Using TNBS, the number of weights to represent the 16-dimensional B-spline surface is reduced from approximately 4.3×10^9 to 3648. The performances on prediction and simulation are listed and compared in Table 2. To the best of our knowledge, the algorithm slightly outperforms the current state-of-the-art results on both prediction and simulation. Fig. 10 shows the true and simulated output on the test set. It is apparent that the TNBS-NARX model was able to accurately capture the nonlinear system dynamics with relatively sparse estimation data. Significantly similar performances on this benchmark were obtained using different combinations of lags, as long as the memory of the input was chosen sufficiently large. The performance of our model is therefore quite robust to the choice of lags, due to the regularization from both the TT-ranks and P-spline penalties.

5. Conclusions

This article presents a new algorithm for nonlinear system identification using a NARX model of which the nonlinear mapping is approximated using the introduced Tensor Network

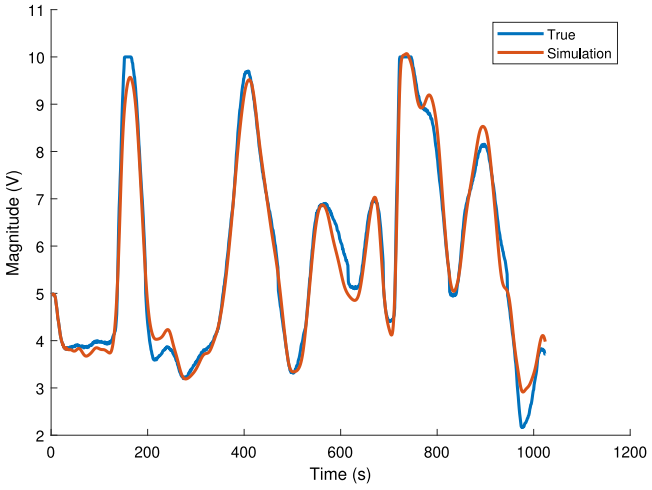

Fig. 10. Simulation on cascaded tanks dataset.

Table 2
Comparison of methods on Cascaded tanks benchmark.

Method	Prediction error (RMSE)	Simulation error (RMSE)
LTI (Schoukens & Scheiwe, 2016)	0.056	0.588
Volterra FB (Schoukens & Scheiwe, 2016)	0.049	0.397
Flexible SS (Svensson & Schön, 2017)	–	0.45
NOMAD (Brunot, Janot, & Carrillo, 2017)	–	0.376
PWARX (Mattsson, Zachariah, & Stoica, 2018)	–	0.350
Sparse Bay. DNN (Zhou, Ibrahim, & Pan, 2019)	0.0472	0.344
TNBS-NARX	0.0461	0.3018

B-splines. Tensor Network theory enables to work with B-spline surfaces directly in a high-dimensional feature space, allowing the identification of NARX systems with a large number of lags and inputs. The identification algorithm is guaranteed to monotonically converge and numerical stability is ensured through orthogonality of the TT-cores. The efficiency and accuracy of the algorithm is demonstrated through numerical experiments on SISO nonlinear systems. Extension of TNBS-NARX to multiple inputs is straightforward through the addition of input variables. Multiple outputs can be realized efficiently by adding an index to one of the TT-cores, as done in Batselier et al. (2017a). Future work includes the implementation of an online optimization scheme, as an alternative to ALS, and the development of control strategies for identified TNBS-NARX systems.

References

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

Batselier, K., Chen, Z., & 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., & Wong, N. (2018). Tensor network subspace identification of polynomial state space models. *Automatica*, *95*, 187–196.

Billings, S. A. (2013). *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons.

de Boor, C. (1978). *A practical guide to splines, vol. 27*. springer-verlag New York.

Brown, M., Bossley, K. M., Mills, D. J., & Harris, C. J. (1995). High dimensional neurofuzzy systems: overcoming the curse of dimensionality. In *Proceedings of 1995 IEEE international conference on fuzzy systems, vol. 4* (pp. 2139–2146). IEEE.

Brunot, M., Janot, A., & Carrillo, F. (2017). Continuous-time nonlinear systems identification with output error method based on derivative-free optimisation. *IFAC-PapersOnLine*, *50*(1), 464–469.

Cichocki, A. (2014). Era of big data processing: A new approach via tensor networks and tensor decompositions. arXiv preprint arXiv:1403.2048.

Cichocki, A., Lee, N., Oseledets, I., Phan, A., Zhao, Q., Mandic, D. P., et al. (2016). Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends® in Machine Learning*, *9*(4–5), 249–429.

Cong, S., & Song, R. (2000). An improved B-spline fuzzy-neural network controller. In *Proceedings of the 3rd world congress on intelligent control and automation (Cat. No. 00EX393), vol. 3* (pp. 1713–1717). IEEE.

Csurcsia, P., Schoukens, J., Kollár, I., & Lataire, J. (2014). Nonparametric time-domain identification of linear slowly time-variant systems using B-splines. *IEEE Transactions on Instrumentation and Measurement*, *64*(1), 252–262.

De Boor, C. (1976). *Splines as linear combinations of B-splines. A survey: Technical report*, Wisconsin Univ Madison Mathematics Research Center.

Eilers, P. H. C., & Marx, B. D. (1996). Flexible smoothing with B-splines and penalties. *Statistical Science*, 89–102.

Eubank, R. L. (1999). *Nonparametric regression and spline smoothing*. CRC Press.

Folgheraiter, M. (2016). A combined B-spline-neural-network and ARX model for online identification of nonlinear dynamic actuation systems. *Neurocomputing*, *175*, 433–442.

Francis, J. G. (1961). The QR transformation a unitary analogue to the LR transformation—Part 1. *The Computer Journal*, *4*(3), 265–271.

Garcke, J., et al. (2006). *Sparse grid tutorial* (p. 7). Mathematical Sciences Institute, Australian National University, Canberra Australia.

Holtz, S., Rohwedder, T., & Schneider, R. (2012). The alternating linear scheme for tensor optimization in the tensor train format. *SIAM Journal on Scientific Computing*, *34*(2), A683–A713.

Hong, X., & Chen, S. (2012). The system identification and control of Hammerstein system using non-uniform rational B-spline neural network and particle swarm optimization. *Neurocomputing*, *82*, 216–223.

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

Leontaritis, I. J., & Billings, S. A. (1985). Input-output parametric models for non-linear systems part I: deterministic non-linear systems. *International Journal of Control*, *41*(2), 303–328.

Lightbody, G., O'Reilly, P., Irwin, G. W., McCormick, J., et al. (1997). Neural modelling of chemical plant using MLP and B-spline networks. *Control Engineering Practice*, *5*(11), 1501–1515.

Lin, Z., Reay, D. S., Williams, B. W., & He, X. (2007). Online modeling for switched reluctance motors using B-spline neural networks. *IEEE Transactions on Industrial electronics*, *54*(6), 3317–3322.

Mattsson, P., Zachariah, D., & Stoica, P. (2018). Identification of cascade water tanks using a PWARX model. *Mechanical Systems and Signal Processing*, *106*, 40–48.

Mirea, L. (2014). Dynamic multivariate B-spline neural network design using orthogonal least squares algorithm for non-linear system identification. In *2014 18th international conference on system theory, control and computing* (pp. 720–725). IEEE.

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

Oseledets, I., & Tyrtyshnikov, E. (2010). TT-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, *432*(1), 70–88.

Penrose, R. (1971). Applications of negative dimensional tensors. *Combinatorial Mathematics and Its Applications*, *1*, 221–244.

Qin, K. (2000). General matrix representations for B-splines. *The Visual Computer*, *16*(3), 177–186.

Reay, D. S. (2003). CMAC and B-spline neural networks applied to switched reluctance motor torque estimation and control. In *IECON'03. 29th annual conference of the IEEE Industrial Electronics Society (IEEE Cat. No. 03CH37468), vol. 1* (pp. 323–328). IEEE.

Rohwedder, T., & Uschmajew, A. (2013). On local convergence of alternating schemes for optimization of convex problems in the tensor train format. *SIAM Journal on Numerical Analysis*, *51*(2), 1134–1162.

dos Santos Coelho, L., & Pessôa, M. W. (2009). Nonlinear identification using a B-spline neural network and chaotic immune approaches. *Mechanical Systems and Signal Processing*, *23*(8), 2418–2434.

Schoenberg, I. J. (1973). *Cardinal spline interpolation, vol. 12*. SIAM.

Schollwöck, U. (2011). The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, *326*(1), 96–192.

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

Schoukens, J., & Scheiwe, F. G. (2016). Modeling nonlinear systems using a Volterra feedback model. In *Workshop on nonlinear system identification benchmarks*. http://www.nonlinearbenchmark.org/FILES/SLIDES/2016-NSIB-Schoukensb.pdf. Online; Accessed 1 March 2020.

Svensson, A., & Schön, T. B. (2017). A flexible state-space model for learning nonlinear dynamical systems. *Automatica*, *80*, 189–199.

Yiu, K. F. C., Wang, S., Teo, K. L., & Tsoi, A. C. (2001). Nonlinear system modeling via knot-optimizing B-spline networks. *IEEE Transactions on Neural Networks*, *12*(5), 1013–1022.

Zenger, C. (1991). In W. Hackbusch (Ed.), *Notes on numerical fluid mechanics*: *vol. 31*, *Sparse grids, parallel algorithms for partial differential equations: Proceedings of the sixth GAMM-seminar*. Braunschweig: Vieweg.

Zhang, X., Zhao, Y., Guo, K., Li, G., & Deng, N. (2017). An adaptive B-spline neural network and its application in terminal sliding mode control for a mobile satcom antenna inertially stabilized platform. *Sensors*, *17*(5), 978.

Zhou, H., Ibrahim, C., & Pan, W. (2019). A sparse Bayesian deep learning approach for identification of cascaded tanks benchmark. arXiv preprint arXiv:1911.06847.

Ridvan Karagoz was born in Rotterdam and received his B.S. degree in Mechanical Engineering and M.S. degree in Systems & Control from the Delft University of Technology in 2017 and 2020 respectively. He is currently working in industry and aims to introduce systems and control theory to new fields. His research interests include nonlinear system identification, tensor networks, optimization, and model predictive control.

Kim Batselier received the M.S. degree in Electro-Mechanical Engineering and the Ph.D. Degree in Applied Sciences from the KULeuven, in 2005 and 2013 respectively. He worked as a research engineer at BioRICS on automated performance monitoring until 2009. He is currently an assistant professor at the Delft University of Technology, The Netherlands. His current research interests include linear and nonlinear system theory/identification, algebraic geometry, tensors, and numerical algorithms.