Extracting LPV and qLPV Structures From State-Space Functions: A TP Model Transformation Based Framework

Péter Baranyi

***Abstract*—This paper proposes a tensor product (TP) model transformation-based framework requiring minimal human intuition to numerically reconstruct linear time invariant, Takagi–Sugeno (T–S) fuzzy model-based linear parameter varying and quasi-linear parameter varying representations of state-space models. The proposed framework facilitates the manipulation of the structure of the system matrix, the parameter vector—including state elements—and the vertex systems. The motivation behind this capability is that all of these structural components strongly influence the control design and the resulting control performance. An important feature of the framework is that it is agnostic towards the formulation of the state-space model, i.e., whether it is given using soft-computing-based techniques or closed formulae. The proposed approach is an extension of the TP model-based control design framework and inherits all of its advantageous properties, e.g., it can be easily used to find minimal representations, including the higher order singular value-based canonical form, and it supports the clear formulation of complexity/accuracy tradeoffs and allows for conversions to various types of convex representations, making for a flexible way to manipulate the weighting and antecedent functions. This paper gives examples to show how the framework can be used in a routine-like fashion and to highlight how it can be applied to the problem of finding useful T–S fuzzy model variations of a given model.**

***Index Terms*—LMI control design, polytop model, TP model, TP model transformation, TS model.**

I. Introduction

THIS paper proposes the tensor product (TP) model transformation [1]–[3]-based framework to numerically reconstruct linear time invariant (LTI), linear parameter varying (LPV), and quasi-LPV (qLPV) representations of state-space models such that the reconstructed models are in TP model form. (Note that in the present discourse the TP model form is equivalent to the T–S fuzzy model form [1]–[4], therefore, whenever this paper makes an assertion concerning the TP model, it will also be applicable to the T–S fuzzy model.) The proposed framework incorporates all advantageous properties of the TP model transformation framework, such as the ability to reconstruct canonical [5], [6] as well as various other types of convex TP model (or T–S fuzzy model) forms [1], [7], and the ability to find the minimal number of vertices (or fuzzy rules) corresponding to a specified approximation accuracy [1], [8]. The goal of this proposed framework is to replace the complicated (or even intractable) derivations of closed formulae with straightforward, tractable numerically appealing routine-like solutions that can be executed in a reasonable amount of time. A further goal is that it should be irrelevant how the state-space model is given (as a set of closed formulae, as a neural network, as a black-box model), and indeed, the numerical steps of the proposed framework can be executed in all of these cases.

The motivation behind exploring variations of the T–S fuzzy model structure can be outlined as follows. Polytopic model-based state-space control design has three key steps: A) defining the state-space model; B) finding the optimal polytopic model (TP model or T–S fuzzy model); and C) deriving the controller. The crucial point is that the structure of the model obtained after Steps A and B has a key role in determining the effectiveness of the controller design obtained in Step C. Thus, it is important to structure the elements of the system matrix in Step A well, since ultimately, the ordering of the elements influences the whole design process in a strong sense. The convex hull defined by the polytopic structure in Step B also directly influences the design. Therefore, the applied control design strategy must take into account these points in order to guarantee that the best controller is obtained for the task at hand [2], [9]–[11].

This paper proposes a TP model transformation-based solution that can be automatically executed and used to derive models for Steps A and B. More importantly, the key novelty of the paper is that it extends the TP model transformation-based control design framework to state-space models whose matrix structure is unknown. Further novelties of the proposed extension include its capability of dealing with the internal structure of the vertices or system matrix in a systematic way, that it allows for the flexible manipulation of this structure in a way that considerably improves the controller design, and that it allows for the specification and design of the external parameters and state vector element contained in the parameter vector.

Much work has been carried out in the recent past on the TP model transformation. Computational analyses and improvements to the original formulation were recently proposed in [12] and [13]. It was also proved in [2], [9]–[11] that linear matrix

Manuscript received January 11, 2019; revised March 8, 2019; accepted March 28, 2019. Date of publication May 8, 2019; date of current version March 2, 2020. This work was supported by the FIEK Program of the Center for Cooperation between Higher Education and the Industries, Széchenyi István University, under Grant GINOP-2.3.4-15-2016-00003.

The author is with Doctoral School of Multidisciplinary Engineering Sciences, Széchenyi István University, Hungary (e-mail: prof.peter.baranyi@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2019.2908770

inequality (LMI)-based control design theories are very sensitive to the properties of the convex hulls (both the shape of the weighting functions and the number of vertices) defined by the TP models; hence, the convex hull manipulation capability of the TP model transformation is an important and necessary step in LMI-based control design. Very effective convex hull manipulation methods were incorporated into the TP model transformation in [7], [14], and [15]. Further useful control approaches and applications were published in the field of control theory [16]–[23], including in the area of sliding mode control in [18], [24], and [25]. For further theories and applications, readers are referred to [26]–[59].

The rest of this paper is organized as follows. Section II defines the notation and the basic concepts of the TP model transformation in a modified form to better fit the proposed framework. Section III outlines the key points of the paper. It highlights the main goal, namely, to extend the TP model transformation to state-space functions where the linear matrix product structure (product of the system matrix and the state vector) is unknown. Sections IV to VII present the novelty of this paper. These sections show how to numerically reconstruct various state-space forms in a reasonable amount of time, even in cases when the exact closed formulae are unknown, but the model can be sampled over a grid. Section IV shows how to numerically reconstruct a global linearization, and also shows how to make the linearization in a given point. Section V presents one of the key points of the paper and shows how to execute the extended TP model transformation even in cases when we do not know the inner linear parameter dependent matrix product-based structure of the given state-space model. Sections VI and VII give some idea of how to manipulate the parameter space and the structure of the system matrix. Section VIII presents some examples to show that the MATLAB implementations of the proposed framework are very simple. The performance of the algorithms are demonstrated on the example of the inverted pendulum. Section IX discusses the contrast between the proposed extension and previous solutions. Finally, Section X concludes this paper.

II. Notations and Preliminary Concepts

This section provides the notations used in the paper and introduces the basic concepts of the TP model transformation in a modified and rather more compact form that is better suited to the proposed framework.

A. Notations

The following notations are used in this paper:

1) Scalar: a;
2) Vector: $\mathbf{a}$ contains elements a_i;
3) Matrix: $\mathbf{A}$ contains elements $a_{i,j}$;
4) Tensor: $\mathcal{A}$ contains elements $a_{i,j,k,\ldots}$;
5) $\mathbf{x} \in \mathbb{R}^N$ denotes that the vector $\mathbf{x}$ contains N elements, and the values of these elements are real numbers;
6) $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ means that the values of the N-dimensional tensor are real numbers. I_n, $n = 1 \ldots N$ denotes the number of elements in the nth dimension of tensor $\mathcal{A}$.;
7) Index i: The upper bounds of the indices are denoted by the uppercase letter, e.g., $i = 1 \ldots I$;
8) Interval: $\omega = [\omega_{\min},\ \omega_{\max}]$;
9) Space: $\Omega : \omega_1 \times \omega_2 \times \cdots \times \omega_N$ is an N-dimensional hypercube;
10) $\mathbf{x} \in \Omega$ expresses the fact that vector $\mathbf{x}$ is within the space Ω. The dimensions of $\mathbf{x}$ and Ω are the same;
11) Ω^x and Ω^p: We frequently define vectors $\mathbf{x} \in \Omega$ and $\mathbf{p} \in \Omega$. In order to denote that these spaces of $\mathbf{x}$ and $\mathbf{p}$ are different, we use superscript x and p as Ω^x and Ω^p. This simply means that $\mathbf{x} \in \Omega^x$ and $\mathbf{p} \in \Omega^p$;
12) $\sqsubset$ refers to a dimensionality reduced subset in general as:
    a) in the case of spaces: $\Theta \sqsubset \Omega$ states that Θ is a hypercube with the same-sized intervals as Ω, but has a smaller number of dimensions;
    b) in the case of vectors: $\mathbf{a} \sqsubset \mathbf{b}$, where $\mathbf{a} \in \Theta \subset \mathbb{R}^N$ and $\mathbf{b} \in \Omega \subset \mathbb{R}^M$ means that $N < M$ and $\Theta \sqsubset \Omega$;
    c) in the case of tensors, $\mathcal{A} \sqsubset \mathcal{B}$ means, for instance, that $\mathcal{A}$ is obtained by deleting complete dimensions from tensor $\mathcal{B}$;
13) $\mathbf{A} = \mathcal{A}_{[n]}$ denotes the layout of tensor $\mathcal{A}$. $\mathbf{A}$ is a matrix with the size of $I_n \times \prod_{i=1, i \neq n}^{N} I_i$, where the size of $\mathcal{A}$ is $I_1 \times I_2 \times \cdots \times I_N$. The column vectors of matrix $\mathbf{A}$ are the vectors of the nth dimension of tensor $\mathcal{A}$. For further details, see [60];
14) $\mathcal{S} \boxtimes_{n=1}^{N} \mathbf{U}_n$ is the TP. It defines the tensor product between tensor $\mathcal{S}$ and matrices $\mathbf{U}$. $\mathcal{S}$ is the N-dimensional core tensor and $\mathbf{U}_n$ are matrices assigned to each dimension. For details refer to [1]–[3];
15) $f(\mathbf{x}) = \mathcal{S} \boxtimes_{n=1}^{N} \mathbf{w}_n(x_n)$ represents the TP function $\mathbf{x} \in \mathbb{R}^N$ where $\mathbf{w}_n(x_n) = [w_{n,1}(x_n)\ \ w_{n,2}(x_n)\ \ \ldots\ \ w_{n,I_n}(x_n)]$ is called the weighting function system. This is equivalent to the very frequently used transfer function of the T–S fuzzy model given in classical form such as

$$y = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N}^{I_N} \prod_{n=1}^{N} w_{n,i_n}(x_n) b_{i_1,i_2,\ldots,i_N} \qquad (1)$$

16) Types of the weighting functions:
    a) SN: Sum normalized;
    b) NN: Nonnegative;
    c) NO: Normalized;
    d) CNO: Close to normalized;
    e) RNO: Relaxed normalized;
    f) INO: Inverse normalized;
    g) IRNO: Inverse relaxed normalized;

    for further details refer to [1] and [2].

B. Concepts

In this section, slight modifications are proposed to the basic concepts used in the TP model transformation. This will allow us to more easily address the key steps of the proposed extensions.

Definition 1. Grid: An equidistant hyper rectangular grid with I dimensions is determined by space $\Omega = \mathbb{R}^{\omega_1 \times \omega_2 \times \cdots \times \omega_I}$ and the density of the grid denoted by $M : M_1 \times M_2 \times \cdots \times$

M_I. Thus the grid, per dimension, is $[g_{i,1}\ g_{i,2}\ \dots\ g_{i,M_i}]$, where $\forall i : g_{i,m} \leq g_{i,m+1}, g_{i,1} = \omega_{\min}, g_{i,M_i} = \omega_{\max}$.

Definition 2. Grid tensor: A grid tensor, denoted as $\mathcal{G}$, contains the coordinates of the hyper rectangular grid (given by Ω and M) in each entry. Each element of the tensor is a vector pointing to the location of one grid. Thus, $\mathbf{g}_{m_1,m_2,\dots,m_I} = [g_{1,m_1}\ g_{2,m_2}\ \dots\ g_{I,m_I}]$, $m_i = 1,\dots,M_i$. A grid tensor is obtained when we store $\mathbf{g}_{m_1,m_2,\dots,m_I}$ in $\mathcal{G}$. This means that $\mathcal{G}$ has $I+1$ dimensions and $\mathcal{G} \in \mathbb{R}^{M_1 \times M_2 \times \dots \times M_I \times I}$.

Definition 3. Multiple operations: Assume a given function $\mathbf{y} = f(\mathbf{x})$, where $\mathbf{y} \in \mathbb{R}^O$ and $\mathbf{x} \in \mathbb{R}^I$. The multiple operation

$$\mathcal{Y} = f(*\mathcal{X}) \tag{2}$$

takes an input tensor $\mathcal{X} \in \mathbb{R}^{M_1 \times M_2 \times \dots \times M_I \times I}$, and returns an output tensor $\mathcal{Y} \in \mathbb{R}^{M_1 \times M_2 \times \dots \times O}$ whose entries are $\mathbf{y}_{m_1,m_2,\dots,m_I} = f(\mathbf{x}_{m_1,m_2,\dots,m_I})$, where $\mathbf{x}_{m_1,m_2,\dots,m_I}$ are the elements of tensor $\mathcal{X}$.

Definition 4. Discretization: The discretized variant $\mathcal{F}^{\mathcal{G}}$ of $f(\mathbf{x})$, $\mathbf{x} \in \Omega$ is

$$\mathcal{F}^{\mathcal{G}} = f(*\mathcal{G}) \tag{3}$$

where $\mathcal{G}$ is a grid defined to Ω with density M.

Definition 5. TP model: The following TP function-based structure is referred to as the TP model:

$$\mathbf{y} = f(\mathbf{p}) = \mathcal{S} \boxtimes_{n=1}^{N} \mathbf{w}_n(p_n) \tag{4}$$

where $\mathbf{y} \in \mathbb{R}^{O_1 \times O_2 \times \dots \times O_K}$, $\mathbf{p} \in \mathbb{R}^N$, and core tensor $\mathcal{S} \in \mathbb{R}^{R_1 \times R_2 \times \dots \times R_N \times O_1 \times O_2 \times \dots \times O_K}$ contains the vertexes $\mathbf{s}_{r_1,r_2,\dots,r_N} \in \mathbb{R}^{O_1 \times O_2 \times \dots \times O_K}$ and vectors $\mathbf{w}_n(p_n) \in \mathbb{R}^{R_n}$ contain the weighting functions $w_{n,r_n}(p_n)$ assigned to the vertexes. Several beneficial properties of TP models, as well as various types thereof have been studied in the past. Two important research areas relevant to the topic are as follows.

1) Higher order singular value (HOSVD)-based canonical form of TP models. Here, R_n also expresses the rank of the function on each dimension. The TP model structure is based on the higher order singular values, the vertexes, and the weightings of orthonormed systems. For further details, see [5] and [6].
2) Convex TP models. Here, the $\mathbf{y}$ is within the convex hull defined by the vertexes. Various types (loose and tight) hulls were defined, such as SN, NN, NO, CNO, RNO, INO, ..., see [1]–[3] and [7]. It was also proved that the manipulation of the convex hull has a crucial role in control design theory, see [2], [9]–[11].

Polytopic form in (4) is the higher structured variant of the frequently used formulae (in polytopic modeling) $\mathbf{S}(\mathbf{p}) = \sum_{h=1}^{H} w_h(\mathbf{p})\mathbf{S}_h$, where vertexes $\mathbf{S}_h$ are the elements $\mathbf{S}_{r_1,r_2,\dots,r_N}$ of the core tensor $\mathcal{S}$, where the multidimensional index $r_1, r_2, \dots, r_N$ is replaced with its linear equivalent $h = 1 \dots H = \prod_1^N R_n$. Accordingly, $w_h(\mathbf{p}) = \prod_{n=1}^N w_{n,r_n}(p_n)$.

Method 1. TP model transformation: Assume a given function $\mathcal{Y} = f(\mathbf{p})$, where $\mathcal{Y} \in \Omega^y \subset \mathbb{R}^O$ and $\mathbf{p} \in \Omega^p \subset \mathbb{R}^N$. The TP model transformation numerically reconstructs the TP model of the given function as

$$f(\mathbf{p}) \approx_\epsilon \mathcal{S} \boxtimes_{n=1}^{N} \mathbf{w}_n(p_n). \tag{5}$$

The TP model transformation finds the exact TP structure, if it exists ($\epsilon = 0$), with the minimal number of vertices and weighting functions. It has further variants such as the pseudo- and multi-TP transformation, as well as other generalized TP model transformations for different purposes [2], [3].

The TP model transformation has three steps.

Step 1: Discretization: Defining $\mathcal{F}^{\mathcal{G}}$ of $f(\mathbf{p})$.

Step 2: Extracting the TP structure: $\mathcal{F}^{\mathcal{G}} = \mathcal{S} \boxtimes_{n=1}^{N} \mathbf{U}_n$ and performing the necessary complexity tradeoff. This is done by executing the HOSVD [60] on $\mathcal{F}^{\mathcal{G}}$. The HOSVD results in matrices $\mathbf{U}_n$ and core tensor $\mathcal{S}$. The matrices $\mathbf{U}_n$ are the singular matrices by dimensions. Further convex hull manipulation can be executed by transforming $\mathbf{U}_n$ to specialized matrices $\mathbf{U}_n^*$ while $\mathcal{S} \boxtimes_{n=1}^{N} \mathbf{U}_n = \mathcal{S}^* \boxtimes_{n=1}^{N} \mathbf{U}_n^*$ as detailed in [1]. This will control the characteristics of the weighting functions determined in the next step. For instance, to have convex combination (i.e., to determine the antecedent membership function in the Ruspini partition), the singular matrices are transformed to CNO type, see [1].

Step 3: Defining weighting functions $\mathbf{w}_n(p_n)$ from $\mathbf{U}_n^*$. There are two ways to reconstruct the weighting functions. One is to simply perform linear interpolation between the elements of the columns in matrix $\mathbf{U}_n$, then each column defines one piecewise linear weighting function. This is referred to as the bilinear TP model transformation. The other way is to recalculate the weighting functions at any given p_n (while all $p_i, i \neq n$ are fixed). This requires the calculation of $f(\mathbf{p})$ for that point. Usually a combination of these two approaches is used. The weighting functions are recalculated over a large number of points "off-line," then a piecewise linear function is used between the points.

III. Problem Outline

Assume a state-space model

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \mathbf{y} \end{bmatrix} = f(\mathbf{x}, \mathbf{u}) \tag{6}$$

where $\begin{bmatrix} \mathbf{x} & \mathbf{u} \end{bmatrix}^T \in \Omega^x \subset \mathbb{R}^I$ contains the state vector $\mathbf{x}$ and input $\mathbf{u}$. Vector $\begin{bmatrix} \dot{\mathbf{x}} & \mathbf{y} \end{bmatrix}^T \in \mathbb{R}^O$ contains output $\mathbf{y}$.

The paper introduces a framework to reconstruct the LTI representation of (6) as

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \mathbf{y} \end{bmatrix} \approx_\epsilon \mathbf{S} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} \tag{7}$$

where $\mathbf{S} \in \mathbb{R}^{O \times I}$ and ϵ are minimized in the least-square sense. If it has no acceptable accuracy then the proposed framework is also capable of reconstructing the LPV representation

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \mathbf{y} \end{bmatrix} \approx_\epsilon \mathbf{S}(\mathbf{p}) \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} \tag{8}$$

of parameter dependent state-space model

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \mathbf{y} \end{bmatrix} = f(\mathbf{x}, \mathbf{u}, \mathbf{p}) \tag{9}$$

where parameter vector $\mathbf{p} \in \Omega^p \subset \mathbb{R}^N$ and ϵ are minimized in the least-square sense. If the approximation error is not acceptable (because of the nonlinearity), the paper proposes a method to reconstruct the qLPV representation as

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \mathbf{y} \end{bmatrix} \approx_\epsilon \mathbf{S}(\mathbf{p}) \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} \tag{10}$$

where vector $\mathbf{p}$ contains elements of the state vector $\mathbf{x}$. Thus, the structure is still linear, however, the model belongs to the family of nonlinear systems.

In the abovementioned cases, the parameter-varying system matrix is given in TP model form as

$$\mathbf{S}(\mathbf{p}) = \mathcal{S} \overset{N}{\underset{n=1}{\boxtimes}} \mathbf{w}_n(p_n) \tag{11}$$

such that tensor $\mathcal{S}$ contains the vertex systems and vectors $\mathbf{w}_n(p_n)$ contain the weighting functions by dimensions of $\mathbf{p}$. Once again, this is equivalent to the T–S fuzzy model, where $\mathcal{S}$ contains the consequent system matrices and $\mathbf{w}_n(p_n)$ contains the antecedent membership functions.

The control design is typically based on the linear matrix form (8) and 10) of the state-space transfer function (6). Since this representation is not unique, determining how the elements of system matrix $\mathbf{S}$ should be organized and structured is crucial. Furthermore, the TP model of $\mathbf{S}(\mathbf{p})$ is also not unique—an infinite number of different vertex and weighting function systems exist to represent the same $\mathbf{S}(\mathbf{p})$. Since the further design steps (e.g., LMI-based design) are based on the vertexes, the appropriate selection of the vertexes is also a crucial point. As a result, both the internal structure of the system matrix and the vertexes of the models strongly influence the further design steps [2], [9]–[11].

The framework proposed in this paper provides various ways to incorporate the TP model transformation in the process of manipulating these important components of the TP model and the design (see also [2]). Using the TP model transformation to begin has the following advantages:

1) it is executable on models given by equations or soft computing-based representations, such as fuzzy rules, neural networks, or other black-box models. The only requirement is that the model must provide an output for each input (at least on a discrete scale);
2) it will find the minimal complexity, namely, the minimal number of components of the TP model (or rules of the T–S fuzzy model). If further complexity reduction is required, it provides one of the best tradeoffs between the number of components (fuzzy rules) and the approximation error;
3) it works like the principal component analysis in that it determines the order of the components (fuzzy rules) according to their importance;
4) it is capable of deriving the weighting functions or antecedent fuzzy sets according to various constraints. For instance, it can be used to define different convex hulls, a capability which has recently been shown to play an important role in control theory as mentioned above;
5) it is capable of transforming the given model to predefined weighting functions (antecedent fuzzy sets), using, i.e., the pseudo-TP model transformation;
6) it is capable of transforming a set of models simultaneously, while deriving common weighting functions (antecedent fuzzy sets) for all models.

IV. Transformation to Linear Structure

The goal is to replace the function

$$\mathbf{y} = f(\mathbf{x}) \tag{12}$$

with a linear mapping

$$\mathbf{y} = \mathbf{S}\mathbf{x} \tag{13}$$

where $\mathbf{x} \in \Omega \subset \mathbb{R}^I$ and $\mathbf{y} \in \mathbb{R}^O$ (hence, $\mathbf{S} \in \mathbb{R}^{O \times I}$).

Method 2. Linearization: The goal is to find a linear mapping $\mathbf{S}$ between a huge number of input–output pairs. Let these pairs be given by the grid tensor $\mathcal{G}$ defined over Ω with density M and by the corresponding outputs in

$$\mathcal{F}^{\mathcal{G}} = f(*\mathcal{G}). \tag{14}$$

Thus, we define $\mathbf{S}$ as

$$\mathcal{F}^{\mathcal{G}} = \mathcal{G} \times_{I+1} \mathbf{S} \tag{15}$$

therefore, using pseudoinverse we have

$$\mathbf{S} = \mathcal{F}^{\mathcal{G}}_{\left[I+1\right]} \left(\mathcal{G}_{\left[I+1\right]} \right)^{+}. \tag{16}$$

At a more detailed level: (14) means

$$\mathbf{f}_{m_1,m_2,\ldots,m_I} = f(\mathbf{g}_{m_1,m_2,\ldots,m_I}) \tag{17}$$

and (15) means

$$\mathbf{f}_{m_1,m_2,\ldots,m_I} = \mathbf{S}\mathbf{g}_{m_1,m_2,\ldots,m_I} \tag{18}$$

for all vectors $\mathbf{f}_{m_1,m_2,\ldots,m_I}$ and $\mathbf{g}_{m_1,m_2,\ldots,m_I}$ stored in tensors $\mathcal{F}^{\mathcal{G}}$ and $\mathcal{G}$, respectively.

Since these vectors are in the $(I+1)$th dimension, we can layout these tensors in the $(I+1)$th dimension and rewrite (15) in the following form:

$$\mathcal{F}^{\mathcal{G}}_{\left[I+1\right]} = \mathbf{S}\left(\mathcal{G}\right)_{\left[I+1\right]}. \tag{19}$$

Thus we have arrived at (16).

As a result, we have

$$f(\mathbf{x}) \approx_\epsilon \mathbf{S}\mathbf{x}. \tag{20}$$

The error can be evaluated, for instance, over another dense grid $\mathcal{H}$ defined over Ω as $\epsilon = \mathcal{F}^H - \mathcal{H} \times_{I+1} \mathbf{S}$ where $\mathcal{F}^H = f(*\mathcal{H})$.

If the error is not acceptable, one alternative approach is to try the quasi-linear structure detailed later.

V. Transformation to Parameter-Varying Linear Structure

The goal is to replace the parameter-varying function

$$\mathbf{y} = f(\mathbf{x}, \mathbf{p}) \tag{21}$$

$\mathbf{y} \in \mathbb{R}^O$, $\mathbf{x} \in \Omega^x \subset \mathbb{R}^I$, $\mathbf{p} \in \Omega^p \subset \mathbb{R}^N$, with

$$\mathbf{y} = \mathbf{S}(\mathbf{p})\mathbf{x} \tag{22}$$

where $\mathbf{S}(\mathbf{p})$ is given in TP model form

$$\mathbf{S}(\mathbf{p}) = \mathcal{S} \overset{N}{\underset{n=1}{\boxtimes}} \mathbf{w}_n(p_n). \tag{23}$$

The ability to incorporate all of the advantageous properties of the TP model transformation is highly desirable here.

Method 3: Parameter-varying linear structure

Step 1: Define discretization grid $\mathcal{G}^p$ to Ω^p with density M^p.

Step 2: Define a linearization grid $\mathcal{G}^x$ fit to Ω^x with density M^x.

Step 3: Linearize $f(\mathbf{x}, \mathbf{p})$ for each $\mathbf{p}_{m_1,m_2,\ldots,m_N}$ of $\mathcal{G}^p$ by Method 2. Namely, define a linear mapping between a sufficiently large number of input–output pairs given by grid $\mathcal{G}^x$ and

$$\mathcal{F}^{\mathcal{G}^x}_{m_1,m_2,\ldots,m_N} = f(*\mathcal{G}^x, \mathbf{p}_{m_1,m_2,\ldots,m_N}) \tag{24}$$

in the form of

$$\mathcal{F}^{\mathcal{G}^x}_{m_1,m_2,\ldots,m_N} = \mathcal{G}^x \times_{I+1} \mathbf{S}_{m_1,m_2,\ldots,m_N}. \tag{25}$$

As a result, we have

$$\mathbf{S}_{m_1,m_2,\ldots,m_N} = \left(\mathcal{F}^{\mathcal{G}^x}_{m_1,m_2,\ldots,m_N}\right)_{\left[I+1\right]} \Psi \tag{26}$$

where

$$\Psi = \left(\mathcal{G}^x_{\left[I+1\right]}\right)^+. \tag{27}$$

Step 4: Since the tensor $\mathcal{S}^{\mathcal{G}^p} \in \mathbb{R}^{M^p_1 \times M^p_2 \times \cdots \times M^p_N \times O \times I}$ constructed from $\mathbf{S}_{m_1,m_2,\ldots,m_N}$ is the discretized variant of $\mathbf{S}(\mathbf{p})$ over $\mathcal{G}^p$, we can continue with the second step of the TP model transformation (incorporating all the beneficial properties of the TP model transformation), which results in

$$\mathcal{S}^{\mathcal{G}^p} = \mathcal{S} \overset{N}{\underset{n=1}{\boxtimes}} \mathbf{U}_n \tag{28}$$

and then, the third step results in

$$\mathbf{S}(\mathbf{p}) = \mathcal{S} \overset{N}{\underset{n=1}{\boxtimes}} \mathbf{w}_n(p_n) \tag{29}$$

thus, to summarize

$$\mathbf{y} = \left(\mathcal{S} \overset{N}{\underset{n=1}{\boxtimes}} \mathbf{w}_n(p_n)\right) \mathbf{x}. \tag{30}$$

Remark 1. A bilinear TP model can be derived directly from $\mathbf{U}_n$, however, in order to reconstruct the weighting functions over any given p_n, $\mathbf{S}$ must be linearized again for the appropriate $\mathbf{p}$.

VI. Variate the Parameter Dependency

In the previous section, the parameter dependence was defined by $\mathbf{y} = f(\mathbf{x}, \mathbf{p})$. The goal here is to modify this dependence and replace vector $\mathbf{p} \in \Omega^p$ with $\mathbf{v} \in \Omega^v$. In many cases, it is desirable that a linear transformation $\mathbf{T}$ of the parameter space be used

$$\mathbf{p} = \mathbf{T}\mathbf{v} \tag{31}$$

thus, the goal is to find the equivalent

$$\mathbf{y} = f(\mathbf{x}, \mathbf{v}) \tag{32}$$

in parameter-varying linear form. We may examine an even more general case and replace vector $\mathbf{p}$ with another parameter vector $\mathbf{v} \in \mathbb{R}^K$, such as

$$\mathbf{p} = T(\mathbf{v}) \tag{33}$$

that leads to

$$\mathbf{y} = f(\mathbf{x}, \mathbf{v}) \tag{34}$$

and execute Method 3 to reveal the parameter-varying linear structure. In this case the discretization grid $\mathcal{G}^v$ is defined within Ω^v and Method 3 will numerically reconstruct

$$\mathbf{y} = \left(\mathcal{S} \overset{K}{\underset{k=1}{\boxtimes}} \mathbf{w}_k(v_k)\right) \mathbf{x}. \tag{35}$$

For instance, if we know that element p_1 always acts in $\mathbf{y} = f(\mathbf{x}, \mathbf{p})$ as p_1^2, then, we may choose to use $v_1 = p_1^2$ instead. Consider the following example:

$$y = x_1 p^2 + x_2 \tag{36}$$

that can be transformed to the parameter-dependent linear form as

$$y = \begin{bmatrix} p^2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{S}(p)\mathbf{x}. \tag{37}$$

However, we may define a new parameter $v = p^2$ as

$$y = x_1 v + x_2 = \begin{bmatrix} v & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{S}(v)\mathbf{x}. \tag{38}$$

This idea can be extended to entire sets of parameters. For instance, consider the following function:

$$y = x_1 \left(p_1^2 + p_2\right) + x_2 p_2 \tag{39}$$

one may choose to use the new parameters $v_1 = p_1^2 + p_2$ and $v_2 = p_2$ and obtain

$$y = x_1 v_1 + x_2 v_2 = \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{S}(\mathbf{v})\mathbf{x}. \tag{40}$$

VII. Transformation to Quasi-Linear Structure

If the above-discussed linearization leads to a larger than acceptable error ϵ, an alternative approach would be to search for a quasi-linear structure.

The goal, then, is to replace the function

$$\mathbf{y} = f(\mathbf{x}) \tag{41}$$

with a linear mapping

$$\mathbf{y} = \mathbf{S}(\mathbf{p})\mathbf{x} \tag{42}$$

where $\mathbf{x} \in \Omega \subset \mathbb{R}^I$ and $\mathbf{y} \in \mathbb{R}^O$ (hence $\mathbf{S} \in \mathbb{R}^{O \times I}$). Here, matrix $\mathbf{S}(\mathbf{p})$ is a function of vector $\mathbf{p}$, where $\mathbf{p} \in \Omega^p \subset \mathbb{R}^N$ is constructed from elements of vector $\mathbf{x}$ as $\mathbf{p} \sqsubseteq \mathbf{x}$, hence, $\Omega^p \sqsubseteq \Omega^x$.

This is still a linear structure, however, it represents a nonlinear mapping between $\mathbf{x}$ and $\mathbf{y}$—hence the name quasi-linear structure.

The following method approximates $\mathbf{S}(\mathbf{p})$ in a polytopic form as

$$\mathbf{S}(\mathbf{p}) = \mathcal{S} \overset{N}{\underset{n=1}{\boxtimes}} \mathbf{w}_n(p_n). \tag{43}$$

It is a crucial point to define those variables in (41) which will be considered as parameters as well. This can usually be done in more than a single way. For instance, consider the following example:

$$y = x_1 x_2 + x_1 \tag{44}$$

which can be given as

$$y = \begin{bmatrix} 1 & p_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{S}(x_1)\mathbf{x} \tag{45}$$

where $p_1 = x_1$. However, another solution (leading to a completely different structure) would be to write

$$y = \begin{bmatrix} p_2 + 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{S}(x_2)\mathbf{x} \tag{46}$$

where $p_2 = x_2$. This example shows that one has to select the desired parameters before starting the numerical reconstruction.

Based on this, first we need to find the components $f_i(\mathbf{x}_i^p)$ of the function as

$$f(\mathbf{x}) = \sum_{i=1}^{I} f_i(\mathbf{x}_i^p) x_i \tag{47}$$

where vector $\forall i : \mathbf{x}_i^p \sqsubseteq \mathbf{x}$ (p denotes that these values are in parameter vector as well). The product with one of the elements x_i of $\mathbf{x}$ is necessary since each element of matrix $\mathbf{S}$ will be multiplied with one element of the vector $\mathbf{x}$. Once the components are defined, the elements of vectors $\mathbf{x}_i^p$ define the parameter p_n, $n = 1, \ldots, N$ of parameter vector $\mathbf{p}$. In the above-mentioned example: $f(\mathbf{x}) = f(x_1)x_2 + x_1$ or $f(\mathbf{x}) = f(x_2)x_1 + x_1 = (f(x_2) + 1)x_1$. Thus, the candidate for the parameter vector is $p_1 = x_1$ or $p_2 = x_2$.

Assume that we have the parameter vector $\mathbf{p} \sqsubseteq \mathbf{x}$ and the components $f_i(\mathbf{p}_i^p)$, $\mathbf{p}_i^p \sqsubseteq \mathbf{p}$ of

$$f(\mathbf{x}, \mathbf{p}) = \sum_{i=1}^{I} f_i(\mathbf{p}_i^p) x_i. \tag{48}$$

Then, we execute the following method to numerically reconstruct

$$\mathbf{S}(\mathbf{p}) = \mathcal{S} \overset{N}{\underset{n=1}{\boxtimes}} \mathbf{w}_n(p_n). \tag{49}$$

For the sake of simplicity, we extend all components $f_i(\mathbf{p}_i^p)$ with zero blocks of those parameters which are not in $\mathbf{p}_i^p$ to obtain

$$f_i(\mathbf{p}) = f_i(\mathbf{p}_i^p) + \sum_j 0 p_j \tag{50}$$

where $\forall j : p_j$ is not included in $\mathbf{p}_r^p$, but is included in $\mathbf{p}$. Thus, we have

$$f(\mathbf{x}, \mathbf{p}) = \sum_{i=1}^{I} f_i(\mathbf{p}) x_i \tag{51}$$

where $\mathbf{p} \sqsubseteq \mathbf{x}$.

Method 4: Reconstruct the quasi-linear structure

Step 1: Define a discretization grid $\mathcal{G}^p$ to $\Omega^p \sqsubseteq \Omega^x$ with density M^p.

Step 2: Define a linearization grid $\mathcal{G}^x_{m_1,m_2,\ldots,m_N}$ to each element of $\mathcal{G}^p$, where the linearization space $\Omega^x_{m_1,m_2,\ldots,m_N}$ is around $\mathbf{g}^p_{m_1,m_2,\ldots,m_N}$ of $\mathcal{G}^p$. Obviously, the interval ω_i^x of those x_i which are not selected to be a parameter can be set arbitrarily within Ω^x.

Step 3: Execute linearization according to Method 2 for each $\mathbf{g}^p_{m_1,m_2,\ldots,m_N} \in \mathcal{G}^p$ as

$$\mathcal{F}^{\mathcal{G}^x_{m_1,m_2,\ldots,m_N}} = f(*\mathcal{G}^x_{m_1,m_2,\ldots,m_N}, \mathbf{g}^p_{m_1,m_2,\ldots,m_N}) \tag{52}$$

and

$$\mathbf{S}_{m_1,m_2,\ldots,m_N} = \left(\mathcal{F}^{\mathcal{G}^x_{m_1,m_2,\ldots,m_N}}\right)_{\left[\, I+1 \,\right]} \Psi \tag{53}$$

where

$$\Psi = \left(\left(\mathcal{G}^x_{m_1,m_2,\ldots,m_N}\right)_{\left[\, I+1 \,\right]} \right)^{+}. \tag{54}$$

Step 4: Since tensor $\mathcal{S}^{\mathcal{G}^p} \in \mathbb{R}^{M_1^p \times M_2^p \times \cdots \times M_N^p \times O \times I}$ constructed from $\mathbf{S}_{m_1,m_2,\ldots,m_N}$ is the discretized variant of $\mathbf{S}(\mathbf{p})$ over $\mathcal{G}^p$, we can continue with the second step of the TP model transformation (incorporating all the beneficial properties of the TP model transformation), that results in

$$\mathcal{S}^{\mathcal{G}^p} = \mathcal{S} \overset{N}{\underset{n=1}{\boxtimes}} \mathbf{U}_n \tag{55}$$

and finally, we have

$$\mathbf{S}(\mathbf{p}) = \mathcal{S} \overset{N}{\underset{n=1}{\boxtimes}} \mathbf{w}_n(p_n) \tag{56}$$

thus

$$\mathbf{y} = \left(\mathcal{S} \overset{N}{\underset{n=1}{\boxtimes}} \mathbf{w}_n(p_n) \right) \mathbf{x} \tag{57}$$

where p_n are the elements of vector $\mathbf{x}$.

Remark 2: A bilinear TP model can be derived directly from $\mathbf{U}_n$, however, in order to reconstruct the weighting functions over any given p_n, $\mathbf{S}$ must be linearized for the appropriate $\mathbf{p}$.

VIII. Examples

This section demonstrates how the above-discussed theories may be applied to the model of the inverted pendulum. An implementation of the relevant procedures is provided in the MATLAB language using the TPtool toolbox (which can be downloaded through the Wikipedia page on TP model-based control). The variables in the MATLAB code use the same notation as was used in the previous sections.

A. Model of the Inverted Pendulum

Assume that we have an inverted pendulum, such that the mass of the pendulum is m_p, the mass of the cart is M_c, the length of the pendulum is l, the motor force is u, and the angle of the pendulum is θ. The inverted pendulum has the following state-space model representation:

$$\dot{\mathbf{x}} = f(\mathbf{x}, u) = \begin{bmatrix} x_2 \\ f_1(\mathbf{x}, u) \\ x_4 \\ f_2(\mathbf{x}, u) \end{bmatrix} \tag{58}$$

where $x_1 = \theta$, $x_2 = \dot{\theta}$, $x_3 = s$, $x_4 = \dot{s}$, and

$$f_1 = \frac{a_1 + a_2}{l(M_c + m_p \sin^2(x_1))} \tag{59}$$

$$f_2 = \frac{a_3 + a_4}{M_c + m_p \sin^2(x_1)} \tag{60}$$

where

$$a_1 = (M_c + m_p) g \sin(x_1) + \cos(x_1) u \tag{61}$$

$$a_2 = b \cos(x_1) x_4 - m_p l \sin(x_1) \cos(x_1) x_2^2 \tag{62}$$

$$a_3 = -m_p g \sin(x_1) \cos(x_1) - b x_4 + u \tag{63}$$

$$a_4 = m_p l \sin(x_1) x_2^2. \tag{64}$$

Let $M_c = 3$ kg, $m_p = 0.2$ kg, $l = 0.31$ m, $b = 0.1$ N/ms^{-1}. The MATLAB code of the models is:

```
function DX=model(X)
th=X(1); dth=X(2); x=X(3); dx=X(4);
u=X(5); si=sin(th); co=cos(th);
Mc=3;mp=0.2;l=0.31;b=0.1;g=9.88;
s1l=l*(Mc+mp*si*si); s2l=Mc+mp*si*si;
T1=(Mc+mp)*g*si/s1l; T2=b*co*dx/s1l;
T3=-mp*l*si*co*dth*dth/s1l;
T4=-co*u/s1l;
P1=-mp*g*si*co/s2l; P2=-b*dx/s2l;
P3=mp*l*si*dth*dth/s2l; P4=u/s2l;
DX=[dth T1+T2+T3+T4 dx P1+P2+P3+P4]';
```

B. Example 1: Reconstruct the Linear Structure

Assume that the equations of the above model are unknown (the code is unknown), only the input $\mathbf{x}, u$ and output $\dot{\mathbf{x}}$ pairs are available. In order to replace $\dot{\mathbf{x}} = f(\mathbf{x})$ with $\dot{\mathbf{x}} = \mathbf{S}\mathbf{x}$, we execute Method 2. First, let us define a grid as

```
function g=Getgrid(m,M,Omega,O)
a=length(m); for i=1:a a=(m(i)-
1)*2*Omega(i)/(M-1);
g(i)=O(i)-Omega(i)+a; end
```

Remark 3: O defines the center of the grid. Omega defines the intervals around these center points (thus, the value in Omega is at the half of the interval) for each dimension.

Then, the linearization is

```
function S=linear(Omega,M,O)
for m1=1:M for m2=1:M for m3=1:M
for m4=1:M for m5=1:M
    m=[m1 m2 m3 m4 m5];
    g=(Getgrid(m,M,Omega,O))';
    G(m1,m2,m3,m4,m5,:)=g;
    DXT(m1,m2,m3,m4,m5,:,:)=model(g);
end end end end end
Xu=ndim_unfold(G,6);
DXu=ndim_unfold(DXT,6);
S=DXu*pinv(Xu)
```

The overall algorithm can then be written as follows:

```
clear;
Omega=[1.6 1.6 1 1 1]; M=5;
O=[0 0 0 0 0];% center of Omega
S=linear(Omega,M,O);
```

The resulting $\mathbf{S}$ is the best approximation in least-square sense. We may increase the grid density M as high as possible. The linearization error is quite large over Ω. For control design purposes, let us linearize the model for θ and $\dot{\theta} \approx 0$ (the typical stabilisation point). Let us execute the above algorithm with `Omega=[1e-10 1e-10 1 1 1];` The resulting $\mathbf{S}$ is

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 33.75 & 0 & 0 & 0.11 & -1.08 \\ 0 & 0 & 0 & 1 & 0 \\ -0.65 & 0 & 0 & -0.03 & 0.33 \end{bmatrix}. \tag{65}$$

Indeed, we arrive at the same matrix when we substitute $\sin(x_1) \approx 0$ and $\cos(x_1) \approx 1$ into (58). The conclusion is that, using Method 2 we can numerically reconstruct the linearization of the model at a global scale or over any desired point, without analytical derivation and even in cases where the closed formulae of the model are not given.

C. Example 2: Extended TP Model Transformation

If the simple replacement of the model with $\dot{\mathbf{x}} = \mathbf{S}\mathbf{x}$ is not enough, because of strong nonlinearities, we may execute Method 3 to find the qLPV representation. Method 3 generates the TP model (or a T–S fuzzy model)-based qLPV representation. This example demonstrates how the proposed extension to the TP model transformation can be used in cases where the state vector $\mathbf{x}$, input $\mathbf{u}$, and $\dot{\mathbf{x}}$, and hence, the size of the system matrix, are known as in Example 1; however, the components of the qLPV structure, such as the elements of the parameter-dependent system matrix $\mathbf{S}(\mathbf{p})$ and parameter vector $\mathbf{p}$ are not known, so that the matrix $\mathbf{S}(\mathbf{p})$ cannot be sampled directly.

Assume again that we have $\dot{\mathbf{x}} = f(\mathbf{x}, u)$, and the goal is to extract

$$\dot{\mathbf{x}} = \mathbf{S}(\mathbf{p}) \begin{bmatrix} \mathbf{x} \\ u \end{bmatrix}. \tag{66}$$

Let us extend the algorithm so that we can perform linearization at each gridpoint over a linearization grid. In order to achieve more stable computation, we use the pseudoinverse at the gridpoint O. Thus, a small modification in the MATLAB function LINEAR is necessary: `G(m1,m2,m3,m4,m5,:)=O;` Then, the algorithm is:

```
D=3; r=1e-10; M=10; LOmega=[r r r r r];
```

```
Omega=[1.6 1.6 1 1 1]; O=[0 0 0 0 0];
for m1=1:M for m2=1:M for m3=1:M
for m4=1:M for m5=1:M
   m=[m1 m2 m3 m4 m5];
   g=(Getgrid(m,M,Omega,O))';
   S=linear(LOmega,D,g);
   ST(m1,m2,m3,m4,m5,:,:)=S;
end end end end end
```

Remark 4: LOmega and r define the linearization subspace for each grid. D is the sampling density for the linearization.

Next, we extract the convex TP model form with SNNN-type weighting functions and vertexes

```
[S,U,sv]=hosvd(ST,[1 1 1 1 1 0 0],1e-11);
for i=1:5 Uc{i}=genhull(U{i},'snnn');
figure(i); plot(Uc{i},'LineWidth',3);
Ucp{i}=pinv(Uc{i});
end
Sc=tprods(S,Ucp);
```

The SNNN weighting functions are determined in cell U and the related vertexes are stored in Sc. Thus, finally we obtain

$$f(\mathbf{x},u)=f(\mathbf{x},u,\mathbf{p})\begin{bmatrix}\mathbf{x}\\u\end{bmatrix}=\mathcal{S}\underset{n}{\boxtimes}\mathbf{W}_n(p_n)\begin{bmatrix}\mathbf{x}\\u\end{bmatrix} \tag{67}$$

where $p_1=x_1$, $p_2=x_2$, $p_3=x_3$, $p_4=x_4$, and $p_5=u$.

D. Example 3: Varying the Components of the Vertexes

This example shows how we can easily modify the structure of the pendulum model and generate various alternative T–S fuzzy models in a few minutes without any analytical derivation of closed formulae. Exploring such variations will allow us to find the best option for further design steps. The state-space model of (58) is

$$\dot{\mathbf{x}}=\begin{bmatrix}0 & 1 & 0 & 0 & 0\\ 0 & t_1/x_2+t_2 & 0 & bt_3 & t_3\\ 0 & 0 & 0 & 1 & 0\\ 0 & v_1/x_2+v_2 & 0 & -b/h_2 & 1/h_2\end{bmatrix}\begin{bmatrix}\dot{\mathbf{x}}\\u\end{bmatrix} \tag{68}$$

where

$$h_1=l(M_c+m_p\sin^2(x_1)) \tag{69}$$

$$h_2=M_c+m_p\sin^2(x_1) \tag{70}$$

$$t_1=(M_c+m_p)b\sin(x_1)/h_1 \tag{71}$$

$$t_2=m_p l\sin(x_1)\cos(x_1)x_2/h_1 \tag{72}$$

$$t_3=\cos(x_1)/h_1 \tag{73}$$

$$v_1=-m_p g\sin(x_1)\cos(x_1)/h_2 \tag{74}$$

$$v_2=m_p l\sin(x_1)x_2/h_2. \tag{75}$$

Its MATLAB code is

```
function S=model1S(P)
dth=P(2); si=sin(P(1)); co=cos(P(1));
Mc=3;mp=0.2;l=0.31;b=0.1;g=9.88;
s1l=l*(Mc+mp*si*si); s2l=Mc+mp*si*si;
T1=(Mc+mp)*g*si/s1l; T2=b*co/s1l;
T3=-mp*l*si*co/s1l; T4=-co/s1l;
P1=-mp*g*si*co/s2l; P2=-b/s2l;
P3=mp*l*si*dth*dth/s2l; P4=1/s2l;
S=[0 1 0 0 0; 0 T1/dth+T3 0 T2 T4;
0 0 0 1 0; 0 P1/dth+P3 0 p2 p4];
```

The nonlinearity is caused by x_1 and x_2. We can execute the TP model transformation (in order to avoid division by zero, the number of grids are set to an even number) as follows:

```
M=50; Omega=[1.6 1.6]; O=[0 0];
for m1=1:M for m2=1:M
   m=[m1 m2];
   g=(Getgrid(m,M,Omega,O))';
   ST(m1,m2,:,:)=model1S(g);
end end
[S,U,sv]=hosvd(ST,[1 1 0 0],1e-11);
for i=1:2 Uc{i}=genhull(U{i},'snnn');
figure(i); plot(Uc{i},'LineWidth',3);
Ucp{i}=pinv(Uc{3});
end
Sc=tprods(S,Ucp);
```

In the end, we arrive at the weighting functions depicted in Fig. 1. Thus, the T–S fuzzy model of the pendulum is

$$\dot{\mathbf{x}}=\mathbf{S}(p_1,p_2)\begin{bmatrix}\mathbf{x}\\u\end{bmatrix}=\mathcal{S}\overset{2}{\underset{n}{\boxtimes}}\mathbf{W}_n(p_n)\begin{bmatrix}\mathbf{x}\\u\end{bmatrix} \tag{76}$$

where $p_1=x_1$ and $p_2=x_2$. The number of rules are $3\times 5=15$. Various alternative parameter-dependent system matrices

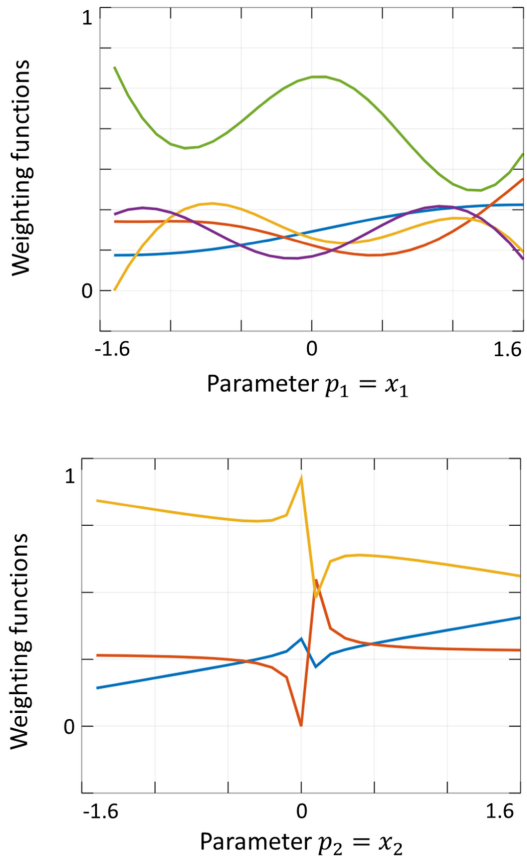

Fig. 1. Weighting functions of p_1 and p_2, see Example 3.

can be derived, for instance

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ t1/x_1 & t_2 & 0 & bt_3 & t_3 \\ 0 & 0 & 0 & 1 & 0 \\ v_1/x_1 & v_2 & 0 & -b/h_2 & 1/h_2 \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}} \\ u \end{bmatrix}. \quad (77)$$

By applying the TP model transformation, all the different T–S fuzzy models can easily be derived from the differently structured system matrices. The type of the convex hull defined by the vertices, namely the type of the weighting functions, can also be manipulated easily, as mentioned in Section II.

E. Example 4: Modifying the Parameter Space

This example shows that the parameter space of the model can also readily be changed. Let us have $p_1 = \sin(x_1)$, $p_2 = \cos(x_2)$, $p_3 = x_2$, and let us change the first line of function MODELS accordingly, as follows:

```
function S=modelS(P)
dth=P(3); si=P(1); co=P(2);
```

We then execute the TP model transformation to derive CNO-type weighting functions as follows:

```
M=50; Omega=[1.6 1.6 0.6]; O=[0 0 0];
for m1=1:M for m2=1:M for m3=1:M
   m=[m1 m2 m3];
   g=(Getgrid(m,M,Omega,O))';
   ST(m1,m2,:,:)=modelS(g);
end end
[S,U,sv]=hosvd(ST,[1 1 1 0 0],1e-11);
for i=1:3 Uc{i}=genhull(U{i},'cno');
figure(i); plot(Uc{i},'LineWidth',3);
Ucp{i}=pinv(Uc{3});
end
Sc=tprods(S,Ucp);
```

Finally, we obtain the weighting function depicted in Fig. 2. Thus, we can express the T–S fuzzy model as follows:

$$\dot{\mathbf{x}} = \mathbf{S}(p_1, p_2, p_3) \begin{bmatrix} \mathbf{x} \\ u \end{bmatrix} = \mathcal{S} \overset{3}{\underset{n}{\boxtimes}} \mathbf{W}_n(p_n) \begin{bmatrix} \mathbf{x} \\ u \end{bmatrix}. \quad (78)$$

Indeed, the dimensionality of the model has increased, but at the same time the number of antecedents in the different dimensions are reduced, and only 18 rules remain. Therefore, it is much easier to manipulate the weighting functions for further control design purposes, i.e., to obtain a CNO-type derivation. We can go further and define the following setting $p_1 = \sin(x_1)$, $p_2 = \cos(x_1)$, $p_3 = x_2$, and $p_4 = 1/x_2$. The resulting weighting functions are depicted in Fig. 3. The number of rules is 24, however, the most simple qLPV model is obtained. Again, we may define a number of variations easily using the presented framework and see which one leads to the best control performance.

IX. Comparison to Other Solutions

If we compare the proposed TP model transformation-based framework to its previous version, the most important difference to highlight is that the proposed version is applicable to a considerably larger class of models. Namely, the previous version

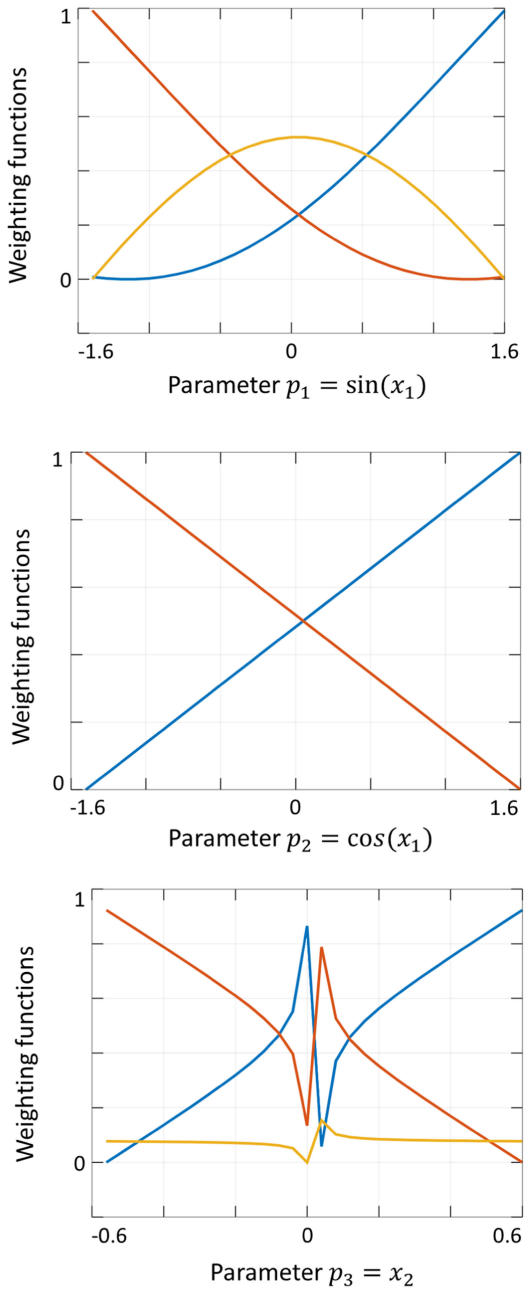

Fig. 2. Weighting functions of p_1, p_2, and p_3, see Example 4.

was applicable to

$$\dot{\mathbf{x}} = \mathbf{S}(\mathbf{x}, \mathbf{p}, u) \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} \quad (79)$$

where the linear matrix structure is already determined, but the current, extended version is applicable to

$$\dot{\mathbf{x}} = f(\mathbf{x}, u, \mathbf{p}) \quad (80)$$

or even to

$$\mathbf{x} = f(\mathbf{x}, u) \quad (81)$$

where the elements of $\mathbf{x}$, $\dot{\mathbf{x}}$, $\mathbf{u}$, and the dimensionality of $\mathbf{S}(\mathbf{p})$ are available; however, the inner structure of the system matrix $\mathbf{S}(\mathbf{p})$ is not known and the parameter vector is not revealed. This

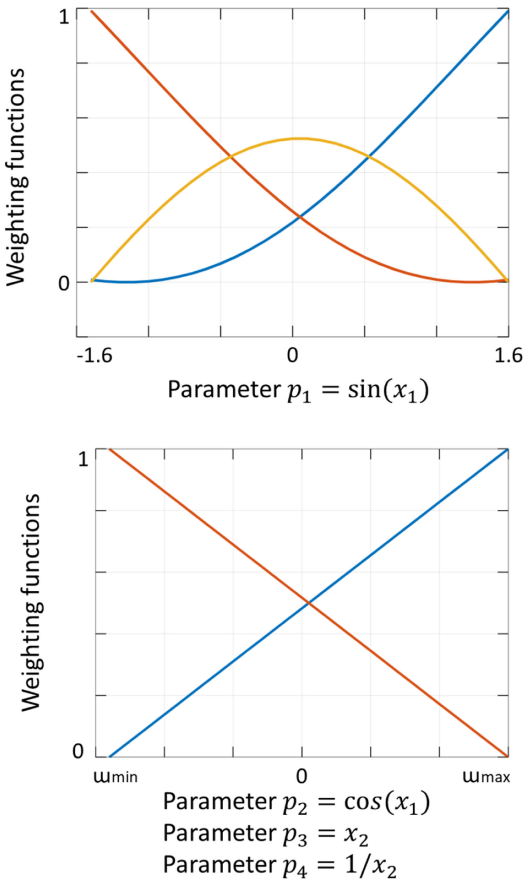

Fig. 3. Weighting functions of p_1, p_2, p_3, and p_4, see Example 4.

means that the proposed version of the TP model transformation offers a very powerful way to extract various alternative T–S fuzzy models of a given dynamic model.

One of the most frequently used approaches to convert a given dynamic model to a T–S fuzzy model is the sector nonlinearity method. Many of the publications dealing with T–S fuzzy control use the sector nonlinearity method to convert the dynamic equations to the T–S fuzzy model. If we compare the proposed method to the sector nonlinearity method, the most important difference is that the TP model transformation provides various features to manipulate all the parameters of the T–S fuzzy model, i.e., the shape of the antecedents, the number of fuzzy rules, and the convex hull defined by the vertexes. It provides an exact HOSVD-based canonical form and complexity tradeoff. The sector nonlinearity based solution is not equipped with such important features. Since the manipulation of the parameters of the T–S fuzzy model is a crucial step, as mentioned in the introduction, the manipulation power of the TP model transformation plays a crucial role in control design. A further important point is that the TP model transformation has a well developed, tractable numerical implementation based on the numerical implementations of HOSVD. Thus, in contrast to the sector nonlinearity method, it is executable automatically with minimal human interaction and has all the same benefits as the HOSVD does for tensors.

X. Conclusion

This paper was based on the observation that an identified state-space model can have an infinite number of T–S fuzzy or TP model representations. Such representations can differ in the structure of the model, and in the construction of the parameter vector—especially if elements of the state vector are involved—and the location of the vertexes in the polytopic form. The selection of the representation strongly influences the control design. Therefore, we were required to derive a huge set of variations to see which one of them leads to the best solution. This paper proposed an effective tool to readily define such variations, and placed the tool into the context of the TP model transformation-based control design framework, which allows for the principled use of the tool as part of a broader toolset for controller design.

Acknowledgment

The author declares that there is no conflict of interest regarding the publication of this paper.

References

[1] P. Baranyi, Y. Yam, and P. Várlaki, *Tensor Product Model Transformation in Polytpic Model Based Control* (Automation and Control Engineering). Boca Raton, FL, USA: CRC Press, 2017.
[2] P. Baranyi, *TP-Model Transformation Based Control Design Frameworks* (Control Engineering). Cham, Switzerland: Springer, 2016.
[3] P. Baranyi, "The generalized TP model transformation for T–S fuzzy model manipulation and generalized stability verification," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 4, pp. 934–948, Aug. 2014.
[4] P. Baranyi, "Extension of the multi-TP model transformation to functions with different numbers of variables," *Complexity*, vol. 2018, 2018, Art. no. 8546976, doi: 10.1155/2018/8546976.
[5] P. Baranyi, L. Szeidl, P. Várlaki, and Y. Yam, "Definition of the HOSVD based canonical form of polytopic dynamic models," in *Proc. IEEE Int. Conf. Mechatronics*, 2006, pp. 660–665.
[6] L. Szeidl and P. Várlaki, "HOSVD based canonical form for polytopic models of dynamic systems," *J. Adv. Comput. Intell. Intell. Inform.*, vol. 13, no. 1, pp. 52–60, 2009.
[7] P. Várkonyi, D. Tikk, P. Korondi, and P. Baranyi, "A new algorithm for RNO-INO type tensor product model representation," in *Proc. 9th IEEE Int. Conf. Intell. Eng. Syst.*, 2005, pp. 263–266.
[8] D. Tikk, P. Baranyi, and R. Patton, "Approximation properties of TP model forms and its consequences to TPDC design framework," *Asian J. Control*, vol. 9, no. 3, pp. 221–231, Oct. 2008.
[9] A. Szollosi and P. Baranyi, "Influence of the tensor product model representation of qLPV models on the feasibility of linear matrix inequality," *Asian J. Control*, vol. 18, no. 4, pp. 1328–1342, Dec. 2015.
[10] A. Szollosi and P. Baranyi, "Improved control performance of the 3-DoF aeroelastic wing section: A TP model based 2D parametric control performance optimization," *Asian J. Control*, vol. 19, no. 2, pp. 450–466, Dec. 2016.
[11] A. Szollosi and P. Baranyi, "Influence of the tensor product model representation of qLPV models on the feasibility of linear matrix inequality based stability analysis," *Asian J. Control*, vol. 10, no. 1, pp. 531–547, Jul. 2017.
[12] J. Cui, K. Zhang, and T. Ma, "An efficient algorithm for the tensor product model transformation," *Int. J. Control, Autom. Syst.*, vol. 14, pp. 1205–1212, Oct. 2016.
[13] S. Nagy, Z. Petres, P. Baranyi, and H. Hashimoto, "Computational relaxed TP model transformation: Restricting the computation to subspaces of the dynamic model," *Asian J. Control*, vol. 11, no. 5, pp. 461–475, Sep. 2009.
[14] X. Liu, Y. Yu, Z. Li, H. H. C. Lu, and T. Fernando, "An efficient algorithm for optimally reshaping the TP model transformation," *IEEE Trans. Circuits Syst. II, Express Briefs*, vol. 64, no. 10, pp. 1187–1191, Oct. 2017.
[15] J. Kuti, P. Galambos, and P. Baranyi, "Minimal volume simplex (MVS) polytopic model generation and manipulation methodology for TP model transformation," *Asian J. Control*, vol. 19, no. 1, pp. 289–301, Jan. 2017.

[16] X. Liu, X. Xin, Z. Li, and Z. Chen, "Near optimal control based on the tensor-product technique," *IEEE Trans. Circuits Syst. II, Express Briefs*, vol. 64, no. 5, pp. 560–564, May 2017.
[17] X. Liu, Y. Yu, Z. Li, and H. H. C. Lu, "Polytopic H_∞ filter design and relaxation for nonlinear systems via tensor product technique," *Signal Process.*, vol. 127, pp. 191–205, Oct. 2016.
[18] G. Zhao, D. Wang, and Z. Song, "A novel tensor product model transformation-based adaptive variable universe of discourse controller," *J. Franklin Inst.*, vol. 353, no. 17, pp. 4471–4499, Nov. 2016.
[19] T. Jiang and D. Lin, "Tensor product model-based gain scheduling of a missile autopilot," *Trans. Jpn. Soc. Aeronaut. Space Sci.*, vol. 59, no. 3, pp. 142–149, 2016.
[20] A. Hajiloo and W. F. Xie, "The stochastic robust model predictive control of shimmy vibration in aircraft landing gears," *Asian J. Control*, vol. 17, no. 2, pp. 476–485, Mar. 2015.
[21] V. C. S. Campos, F. O. Souza, L. A. B. Torres, and R. M. Palhares, "New stability conditions based on piecewise fuzzy Lyapunov functions and tensor product transformations," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 4, pp. 784–760, Aug. 2013.
[22] S. Kuntanapreeda, "Tensor product model transformation based control and synchronization of a class of fractional-order chaotic systems," *Asian J. Control*, vol. 17, no. 2, pp. 371–373, Mar. 2015.
[23] R.-E. Precup, E. M. Petriu, M.-B. Radac, S. Preitl, L.-O. Fedorovici, and C.-A. Dragos, "Cascade control system-based cost effective combination of tensor product model transformation and fuzzy control," *Asian J. Control*, vol. 17, no. 2, pp. 381–391, 2015.
[24] G. Zhao, H. Li, and Z. Song, "Tensor product model transformation based decoupled terminal sliding mode control," *Int. J. Syst. Sci.*, vol. 47, no. 8, pp. 1791–1803, Aug. 2014.
[25] P. Korondi, "Tensor product model transformation-based sliding surface design," *Acta Polytechnica Hungarica*, vol. 3, no. 4, pp. 23–35, 2006.
[26] T. Wang, W. Xie, G. Liu, and Y. Zhao, "Quasi-min-max model predictive control for image-based visual servoing with tensor product model transformation," *Asian J. Control*, vol. 17, no. 2, pp. 402–416, 2015.
[27] J. Kuti, P. Galambos, and Á. Miklós, "Output feedback control of a dual-excenter vibration actuator via qLPV model and TP model transformation," *Asian J. Control*, vol. 17, no. 2, pp. 432–442, 2015.
[28] J. Pan and L. Lu, "TP model transformation via sequentially truncated higher-order singular value decomposition," *Asian J. Control*, vol. 17, no. 2, pp. 467–475, 2015.
[29] J. Matuško, Š. Ileš, F. Kolonić, and V. Lešić, "Control of 3D tower crane based on tensor product model transformation with neural friction compensation," *Asian J. Control*, vol. 17, no. 2, pp. 443–458, 2015.
[30] S. Campos, V. Costa, L. A. B. Tôrres, and R. M. Palhares, "Revisiting the TP model transformation: Interpolation and rule reduction," *Asian J. Control*, vol. 17, no. 2, pp. 392–401, 2015.
[31] G. Zhao, K. Sun, and H. Li, "Tensor product model transformation based adaptive integral-sliding mode controller: Equivalent control method," *Sci. World J.*, vol. 2013, 2013, Art. no. 726963.
[32] S. Chumalee and J. F. Whidborne, "Gain-scheduled H_∞ control for tensor product type polytopic plants," *Asian J. Control*, vol. 17, no. 2, pp. 417–431, 2015.
[33] B. Takarics and Y. Yam, "Robust grid point-based control design for LPV systems via unified TP transformation," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2015, pp. 2626–2631.
[34] Z. He, M. Yin, and Y. Lu, "Tensor product model-based control of morphing aircraft in transition process," *Proc. Inst. Mech. Eng., Part G, J. Aerosp. Eng.*, vol. 230, no. 2, pp. 378–391, 2016.
[35] J. Chen, R. Li, and C. Cao, "Convex polytopic modeling for flexible joints industrial robot using TP-model transformation," in *Proc. IEEE Int. Conf. Inf. Autom.*, 2014, pp. 1046–1050.
[36] R.-E. Precup, C.-A. Dragos, S. Preitl, M.-B. Radac, and E. M. Petriu, "Novel tensor product models for automatic transmission system control," *IEEE Syst. J.*, vol. 6, no. 3, pp. 488–498, Sep. 2012.
[37] K. K. Wu and Y. Yam, "Control stability of TP model transformation design via probabilistic error bound of plant model," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2013, pp. 1259–1264.
[38] R.-E. Precup, L.-T. Dioanca, E. M. Petriu, M.-B. Rădac, S. Preitl, and C.-A. Dragoş, "Tensor product-based real-time control of the liquid levels in a three tank system," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatron.*, 2010, pp. 768–773.
[39] F. Kolonić, A. Poljugan, and I. Petrović, "Tensor product model transformation-based controller design for gantry crane control system—An application approach," *Acta Polytechnica Hungarica*, vol. 3, no. 4, pp. 95–112, 2006.
[40] P. Grof and Y. Yam, "Furuta pendulum—A tensor product model-based design approach case study," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2015, pp. 2620–2625.
[41] C. Ariño and A. Sala, "Relaxed LMI conditions for closed-loop fuzzy systems with tensor-product structure," *Eng. Appl. Artif. Intell.*, vol. 20, no. 8, pp. 1036–1046, 2007.
[42] F. Yang, Z. Chen, X. Liu, and B. Liu, "A new constant gain Kalman filter based on TP model transformation," in *Proc. Chin. Intell. Autom. Conf.*, 2013, pp. 305–312.
[43] Š. Ileš, J. Matuško, and F. Kolonić, "Tensor product transformation based speed control of permanent magnet synchronous motor drives," in *Proc 17th Int. Conf. Elect. Drives Power Electron. (5th Joint Slovak-Croatian Conf.)*, 2011, pp. 323–328.
[44] Š. Ileš, J. Matuško, and F. Kolonić, "TP transformation based control of rotary pendulum," in *Proc. 34th IEEE Int. Conv.*, 2011, pp. 833–839.
[45] R. Precup, C. Dragoş, S. Preitl, M. Rădac, and E. Petriu, "Tensor product models for automotive applications," in *Proc. 14th Int. Conf. Syst. Theory Control*, 2010, pp. 405–410.
[46] P. Korondi, "Sector sliding mode design based on tensor product model transformation," in *Proc. 11th IEEE Int. Conf. Intell. Eng. Syst.*, 2007, pp. 253–258.
[47] Y. Kunii, B. Solvang, G. Sziebig, and P. Korondi, "Tensor product transformation based friction model," in *Proc. 11th IEEE Int. Conf. Intell. Eng. Syst.*, 2007, pp. 259–264.
[48] P. Korondi, "Sliding sector design based on tensor product model transformation," *Trans. Autom. Control Comput. Sci.*, vol. 51, no. 1, pp. 101–108, 2006.
[49] P. Korondi, P. Bartal, and F. Kolonić, "Friction model based on tensor product transformation," in *Proc. 7th Int. Symp. Hung. Res. Comput. Intell.*, 2006, pp. 83–94.
[50] Š. Ileš, F. Kolonić, and J. Matuško, "Linear matrix inequalities based H_∞ control of gantry crane using tensor product transformation," in *Proc. 18th Int. Conf. Process Control*, 2011, pp. 92–99.
[51] A. Rövid, L. Szeidl, and P. Várlaki, "On tensor-product model based representation of neural networks," in *Proc. 15th IEEE Int. Conf. Intell. Eng. Syst.*, 2011, pp. 69–72.
[52] A. M. F. Pereire, L. M. S. Vianna, N. A. Keles, and V. C. S. Campos, "Tensor product model transformation simplification of Takagi–Sugeno control and estimation laws—An application to a thermoelectric controlled chamber," *Acta Polytechnica Hungarica*, vol. 15, no. 3, pp. 13–29, 2018.
[53] V. C. S. Campos, L. M. S. Vianna, and M. F. Braga, "A tensor product model transformation approach to the discretization of uncertain linear systems," *Acta Polytechnica Hungarica*, vol. 15, no. 3, pp. 31–43, 2018.
[54] Y. Kan, Z. He, and J. Zhao, "Tensor product model-based control design with relaxed stability conditions for perching maneuvers," *Acta Polytechnica Hungarica*, vol. 15, no. 3, pp. 45–61, 2018.
[55] H. Gong *et al.*, "Tensor product model-based control for spacecraft with fuel slosh dynamics," *Acta Polytechnica Hungarica*, vol. 15, no. 3, pp. 63–80, 2018.
[56] H. Du, J. Yan, and Y. Fan, "A state and input constrained control method for air-breathing hypersonic vehicles," *Acta Polytechnica Hungarica*, vol. 15, no. 3, pp. 81–99, 2018.
[57] L. Kovács and G. Eigner, "Tensor product model transformation-based parallel distributed control of tumor growth," *Acta Polytechnica Hungarica*, vol. 15, no. 3, pp. 101–123, 2018.
[58] B. Takarics and P. Baranyi, "Tensor-product-model-based control of a three degrees-of-freedom aeroelastic model," *J. Guid., Control, Dyn.*, vol. 36, no. 5, pp. 1527–1533, 2013.
[59] B. Takarics, A. Szollosi, and B. Vanek, "Tensor product type polytopic LPV modeling of aeroelastic aircraft," in *IEEE Aerosp. Conf.*, 2018, pp. 1–10.
[60] L. de Lathauwer, B. de Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Anal.*, vol. 21, no. 4, pp. 1253–1278, 2000.

Peter Baranyi received the M.Sc. and Ph.D. degrees in informatics from the Budapest University of Technology and Economics Institution, Budapest, Hungary, in 1995 and 1999, respectively, and the D.Sc. degree in control theory from the Hungarian Academy of Sciences, Budapest, Hungary, in 2006.

He initiated the core theory of the TP model transformation. His key research interests include quasi-linear parameter varying, linear matrix inequality, and tensor product model transformation-based control design.

Dr. Baranyi is currently a Member of the Hungarian Academy of Engineering.