Fast Tensor Product Model Transformation for Higher Dimensional Problems

Péter Baranyi, Zoltán Petres, László T. Kóczy, and Péter Várlaki

***Abstract*— Tensor Product (TP) model transformation method was proposed recently as an automated gateway between a class of non-linear models and linear matrix inequality based control design. The core of the TP model transformation is the higher order singular value decomposition of a large sized tensor, which requires high computational power that is usually out of a regular computer capacity in case of higher dimensionality. This disadvantage restricts the applicability of the TP model transformation method to linear parameter varying state-space models with smaller number of state values. The aim of this paper is to propose a modification of the TP model transformation. The proposed version needs considerable less computational effort. The paper also presents a 6 dimensional problem to show that the execution of the original TP model transformation fails in higher dimension, but the modified transformation leads to solution.**

***Index Terms*— Non-linear control design, TP model transformation, parallel distributed compensation, linear matrix inequality**

I. Introduction

TP model transformation control design was proposed in recent papers [1], [2]. It is a numerical method capable of transforming given Linear Parameter Varying (LPV) state-space models (where the parameter vector may contain elements of the state vector as well) into a parameter varying convex combination of linear time invariant (LTI) models, namely to *polytopic* models. The convex combination is determined in such a way that a set of linear matrix inequality (LMI) control design theorems can immediately be applied to the resulting polytopic model. Especially those LMIs which have been developed under the Parallel Distributed Compensation framework (PDC) [3]. Based on this TP model transformation and PDC design concept the control solutions of complex and benchmark problems were presented in papers [4]–[10] and at the special session of IEEE Int. Conf. Fuzzy Systems 2004 [11]–[14] and at the IEEE Int. Conf. Intelligent Engineering Systems [15]–[18]. The advantage of the TP model transformation based control design framework is that it can be automatically executed on a regular computer without human interaction. The disadvantage is that the computational load of the TP model transformation explodes easily with the dimensionality of the control problem. The reason of the computational explosion is that the core step of the transformation executes Higher Order Singular Value Decomposition (HOSVD) on a large size multi dimensional tensor resulted by the discretization of the given LPV model. The number of the dimensions of the tensor equals the number of the elements of the parameter vector. If the number of these elements is high (this practically means more than 4 or 5) then a regular computer cannot execute the TP model transformation.

The aim of this paper is to relax the computational explosion and to propose a modification of the TP model transformation. The original TP model transformation computes all the singular values of the HOSVD. However, the resulting TP model needs those singular values only, which determine the weighting functions of the resulting polytopic model. The number of the weighting functions is considerable less than the number of all singular values of the discretized tensor of the LPV model in practical cases. This implies that the computation of all singular values is not necessary. The key idea in the proposed modified TP model transformation is that we restrict the computation of the HOSVD to the required singular values only. In this case, however, we should further transform the result to keep the accuracy of the next steps of the TP model transformation.

The paper also present a numerical example with a dynamic model where the parameter vector has 6 elements. The example shows that in this case the original TP model transformation cannot be executed on a regular computer because of its computational complexity. The example also shows that the modified TP model transformation is, however, executable.

The paper is organized as follows: Section 2 introduces the notation and basic concepts being used later on. Section 3 is devoted to introduce the computational complexity problem of the TP model transformation. Section 4 presents the main novelty of the paper, it proposes the modified TP model transformation. Section 5 shows the effectiveness of the modified TP model transformation via a numerical example with comparison to the original TP model transformation. Section 6 concludes the paper.

II. Nomenclature and basic concepts

This section is devoted to introduce the notations and basic concepts being used in this paper.

- $\{a, b, \dots\}$: scalar values;
- $\{\mathbf{a}, \mathbf{b}, \dots\}$: vectors;
- $\{\mathbf{A}, \mathbf{B}, \dots\}$: matrices;
- $\{\mathcal{A}, \mathcal{B}, \dots\}$: tensors;
- $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$:vector space of real valued $(I_1 \times I_2 \times \cdots \times I_N)$-tensors.
- Subscript defines lower order: for example, an element of matrix $\mathbf{A}$ at row-column number i, j is symbolized as

Affiliation and correspondence: Computer and Automation Research Institute of the Hungarian Academy of Sciences, H–1111 Budapest, Kende utca 13–17, Hungary, phone: +36 (1) 2796111, fax: +36 (1) 4667503, e-mail: baranyi@sztaki.hu

$(\mathbf{A})_{i,j} = a_{i,j}$. Systematically, the ith column vector of $\mathbf{A}$ is denoted as $\mathbf{a}_i$, i.e. $\mathbf{A} = \begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \cdots\end{bmatrix}$.

- $(\cdot)_{i,j,n}, \ldots$: are indices;
- $(\cdot)_{I,J,N}, \ldots$: index upper bound: for example: $i = 1..I$, $j = 1..J$, $n = 1..N$ or $i_n = 1..I_n$.
- $\mathbf{A}^+$: the pseudo inverse of matrix $\mathbf{A}$.
- $\mathbf{A}_{(n)}$: n-mode matrix of tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$;
- $\mathcal{A} \times_n \mathbf{U}$: n-mode matrix-tensor product;
- $\mathcal{A} \otimes_n \mathbf{U}_n$: multiple product as $\mathcal{A} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 .. \times_N \mathbf{U}_N$;

Detailed discussion of tensor notations and operations is given in [19].

A. LPV model

Consider the following linear parameter-varying state-space model:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(\mathbf{p}(t))\mathbf{x}(t) + \mathbf{B}(\mathbf{p}(t))\mathbf{u}(t), \quad (1)$$
$$\mathbf{y}(t) = \mathbf{C}(\mathbf{p}(t))\mathbf{x}(t) + \mathbf{D}(\mathbf{p}(t))\mathbf{u}(t),$$

with input $\mathbf{u}(t)$, output $\mathbf{y}(t)$ and state vector $\mathbf{x}(t)$. The system matrix

$$\mathbf{S}(\mathbf{p}(t)) = \begin{pmatrix}\mathbf{A}(\mathbf{p}(t)) & \mathbf{B}(\mathbf{p}(t))\\ \mathbf{C}(\mathbf{p}(t)) & \mathbf{D}(\mathbf{p}(t))\end{pmatrix} \in \mathbb{R}^{O \times I} \quad (2)$$

is a parameter-varying object, where $\mathbf{p}(t) \in \Omega$ is time varying N-dimensional parameter vector, and is an element of the closed hypercube $\Omega = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_N, b_N] \subset \mathbb{R}^N$. $\mathbf{p}(t)$ can also include the elements of state-vector $\mathbf{x}(t)$, therefore (1) is considered in the class of nonlinear dynamic state-space models.

B. TP model

$\mathbf{S}(\mathbf{p}(t))$ can be approximated for any parameter $\mathbf{p}(t)$ as the convex combination of LTI (Linear Time Invariant) system matrices $\mathbf{S}$. These matrices are also called *vertex systems*:

$$\begin{pmatrix}\dot{\mathbf{x}}(t)\\ \mathbf{y}(t)\end{pmatrix} \underset{\varepsilon}{\approx} \mathcal{S} \overset{N}{\underset{n=1}{\otimes}} \mathbf{w}_n(p_n(t)) \begin{pmatrix}\mathbf{x}(t)\\ \mathbf{u}(t)\end{pmatrix}, \quad (3)$$

where ε symbolizes the approximation error, and row vector $\mathbf{w}_n(p_n) \in \mathbb{R}^{I_n}$ $n = 1, \ldots, N$ contains one variable weighting functions $w_{n,i_n}(p_n)$, $(i_n = 1..I_n)$. Function $w_{n,i_n}(p_n(t)) \in [0,1]$ is the i_n-th weighting function defined on the n-th dimension of Ω, and $p_n(t)$ is the n-th element of vector $\mathbf{p}(t)$. I_n denotes the number of the weighting functions used in the n-th dimension of Ω. Note that the dimensions of Ω are respectively assigned to the elements of the parameter vector $\mathbf{p}(t)$. The $(N+2)$-dimensional coefficient tensor $\mathcal{S} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times O \times I}$ is constructed from LTI vertex systems $\mathbf{S}_{i_1 i_2 \ldots i_N} \in \mathbb{R}^{O \times I}$. For further details we refer to [1], [2], [5], [6]. The convexity is ensured by the conditions:

$$\forall n, p_n(t) : \sum_{j=1}^{I_n} w_{n,j}(p_n(t)) = 1, \quad (4)$$
$$\forall n, i_n, p_n(t) : w_{n,i_n}(p_n(t)) > 0.$$

Having the above TP model (3) the type of the convex hull defined by the LTI systems $\mathbf{S}_{i_1 i_2 \ldots i_N}$ can readily be defined via the type of the one variable weighting functions, see papers [10], [15], [16], [18]. The type of the convex hull effects the feasibility of the further LMI design see paper [6]. Note that $\varepsilon = 0$ cannot be achieved with bounded I_n in general. In this case the task is to minimize I_n at a given ε. However, if $\mathbf{S}(\mathbf{p}(t))$ has the above tensor product structure then we say that the finite element exact decomposition exist. The TP model transformation determines the minimal number of the LTI systems to a given ε or finds the exact finite element TP model if exists, see Ref. [13], [20].

Note that the TP model 3 can always be simply given in the widely adopted formulation of polytopic models:

$$\begin{pmatrix}\dot{\mathbf{x}}(t)\\ \mathbf{y}(t)\end{pmatrix} \underset{\varepsilon}{\approx} \sum_{r=1}^{R} w_r(\mathbf{p}(t))\mathbf{S}_r \begin{pmatrix}\mathbf{x}(t)\\ \mathbf{u}(t)\end{pmatrix}. \quad (5)$$

We arrange $\mathbf{S}_{i_1 i_2 \ldots i_N}$ in the sequence of $r = 1..R = I_1 \times I_2 \times \ldots I_N$ as $\mathbf{S}_r = \mathbf{S}_{i_1 i_2 \ldots i_N}$, and we define the multi-variable weighting functions according to the sequence of r as:

$$w_r(\mathbf{p}(t)) = \prod_n w_{n,i_n}(p_n(t)).$$

C. TP model transformation

The TP model transformation starts with the dynamic model (1) and results in the TP model (3) over Ω [1], [2], [5], [6]. Note that the TP model transformation guarantees that the resulting weighting functions satisfy (4) furthermore, beyond convexity it guaranties tight convex hull that significantly improves the feasibility of further LMI design. It is important to emphasize here that, in order to start with the TP model transformation, one needs an explicit model, such that the system matrix $\mathbf{S}(\mathbf{p}(t))$ can be sampled for all possible values of the parameters $\mathbf{p}(t) \in \Omega$. Whether these equations represent a physical model, or are just the outcome of black-box identification (e.g. neural net model) is irrelevant.

III. Problem statement

The TP model transformation is a numerical method and has three key steps. The first step is the discretization (sampling) of the given $\mathbf{S}(\mathbf{p}(t))$ over a dense hyper rectangular grid Θ in the transformation space Ω. This step yields the discretized systems in tensor $\mathbf{S}^D$ (superscript "D" means "discretized"). In order to loose minimal information during the discretization we apply as dense grid Θ as possible. The second step extracts the LTI vertex systems from the $\mathbf{S}^D$. This step is specialized to find the minimal number of LTI vertex systems as the vertex points of various types of convex hulls of the given system. The types of the convex hulls are determined via further transformation steps provided in the TP model transformation framework [10], [15], [21], [22]. In this step the more dense Θ we have the more proper convex hull can be generated. The third step constructs the TP model based on the LTI vertex systems obtained in the second step. It defines the continuous weighting functions to the LTI vertex systems.

In conclusion, the more dense grid Θ is applied the more proper convex hull can be generated. However, the second step executes HOSVD on the tensor $\mathbf{S}^D$. The complexity (number of elements) of $\mathbf{S}^D$ equals the complexity (number of sampling points) of the discretization grid Θ. The computational load of HOSVD explodes with the complexity of $\mathbf{S}^D$. Generating the proper convex hull practically needs at least about 100 grid points by dimensions, see papers [4]–[9], [11], [15]–[18]. This leads to the fact that in the case of higher dimension (in case of 4 dimensions the number of sampling points becomes about 100^4) the HOSVD cannot be executed on $\mathbf{S}^D$, because its computational load becomes out of a regular computer capacity. This fact enforces us to use as sparse grid Θ as possible.

According to the above, we have two contradictory constraints to define the density of Θ. The first constrain enforces us to use dense Θ to achieve a proper decomposition of $\mathbf{S}(\mathbf{p}(t))$ with fine convex hulls. The second constrain comes from the easily exploding computational load of HOSVD.

This paper proposes a modification of the TP model transformation. This modification is based on the fact that in the resulting TP model we expect only a few weighting functions by dimensions based on the following two reasons. The first reason is that we cannot apply more than a few weighting functions (about up to 4-5) by dimensions, because of the computational load of the further LMI based controller design also explodes easily with the number of the resulting LTI systems, and therefore we may not able to solve them. The second reason is that a number of examples shows that [4]–[9], [11], [16]–[18] a few weighting functions are enough by dimensions to have exact or close to exact transformation. This is especially typical, when we start form a mathematical model. In conclusion, when we execute the HOSVD we do not need to calculate all singular values (according to the above example we do not need to calculate 100 singular values by dimensions), but we must calculate the first few ones related to the resulting weighting functions. The next section proposes the modification of the TP model transformation based on this idea.

IV. Modification of the TP model transformation

The key point of the modification is that we apply two discretisation grid Θ and Γ. Θ is a sparse grid where upon we discretize the $\mathbf{S}(\mathbf{p}(t))$ and generate $\mathbf{S}^{D,\Theta}$. Superscript "Θ" indicates that $\mathbf{S}^D$ is the discretization over Θ. Then we execute HOSVD on the "small-sized" $\mathbf{S}^{D,\Theta}$ to yield the minimal number of LTI systems and do the approximation trade-off if necessary. So, Θ is defined according to the maximum computational capacity available for HOSVD. After this we introduce a new step capable of expanding the resulting decomposition to the dense grid Γ without generating $\mathbf{S}^{D,\Gamma}$ (that would be an extremely huge tensor) and executing HOSVD on $\mathbf{S}^{D,\Gamma}$. Once we have the decomposition over Γ then we can generate "fine" convex hulls as in the original TP model transformation. The last step generates the continuous weighting functions in the same way as in the original TP model transformation.

A. *The algorithm of the modified TP model transformation*

Step 1) Discretization over a sparse grid Θ

- Define the transformation space Ω as: $\mathbf{p}(t) \in \Omega : [a_1, b_1] \times [a_2, b_2] \times .. \times [a_N, b_N]$.
- Define a hyper rectangular N-dimensional sparse grid Θ by equidistantly located grid-lines: $\theta_{n,h_n} = a_n + \frac{b_n - a_n}{H_n - 1}(h_n - 1)$, $h_n = 1..H_n$. The numbers of the grid lines in the dimensions are H_n. Notice here that the H_n must be larger or equal to the allowed maximum number of weighting functions in each dimensions of the TP model to be generated. The reason of this is that the Second step can not generate more singular values than the size of the discretised system. The number of weighting functions are equal to the number of the remaining singular values in each dimension.
- Sample the given function $\mathbf{S}(\mathbf{p}(t))$ over the grid-points of Θ:
$$\mathbf{S}^{D,\Theta}_{h_1,h_2,..,h_N} = \mathbf{S}(\mathbf{p}_{h_1,h_2,..,h_N}) \in \mathbb{R}^{O \times I},$$
where $\mathbf{p}_{h_1,h_2,..,h_N} = \begin{pmatrix} \theta_{1,h_1} & \theta_{2,h_2} & .. & \theta_{N,h_N} \end{pmatrix}$.
- Store the discretized matrices $\mathbf{S}^{D,\Theta}_{h_1,h_2,..,h_N}$ into the tensor $\mathcal{S}^{D,\Theta} \in \mathbb{R}^{H_1 \times H_2 \times .. \times H_N \times O \times I}$

Step 2) Approximation trade-off via HOSVD

In this step we execute HOSVD on the first N dimensions of tensor $\mathcal{S}^{D,\Theta}$. During performing the HOSVD we discard all zero or small singular values σ_k and their corresponding singular vectors in all dimensions like in the original TP model transformation. Assume that we keep I_n singular values in dimensions $n = 1..N$. As a result we have

$$\mathcal{S}^{D,\Theta} \underset{\delta}{\approx} \mathcal{S} \underset{n}{\otimes} \mathbf{U}^{\Theta}_n. \tag{6}$$

The error δ is bounded as:

$$\delta = \left(\| \mathcal{S}^D - \mathcal{S} \underset{n}{\otimes} \mathbf{U}^{\Theta}_n \|_{L_2} \right)^2 \leq \sum_k \sigma_k^2. \tag{7}$$

The resulting tensor $\mathcal{S}$, with the size of $(I_1 \times I_2 \times .. \times I_N \times O \times I)$, where $\forall n : I_n \leq H_n$, contains the first variant of the LTI vertex systems, and is immediately substitutable into (3). The size of $\mathbf{U}_n$ is $H_n \times I_n$.

Step 3) Increasing the grid density to Γ

This step expands the result of the second step to the dense discretization grid Γ. Define a hyper rectangular dense grid Γ by equidistantly located grid-lines: $\gamma_{n,m_n} = a_n + \frac{b_n - a_n}{M_n - 1}(m_n - 1)$, $m_n = 1..M_n$. The numbers of the grid lines in the dimensions are M_n. Define M_n in such a way that all grid points of Θ are in Γ, namely $\Theta \in \Gamma$.

The goal is to generate the HOSVD of $\mathcal{S}^{D,\Gamma}$ as:

$$\mathcal{S}^{D,\Gamma} \underset{\varepsilon}{\approx} \mathcal{S} \underset{n}{\otimes} \mathbf{U}^{\Gamma}_n, \tag{8}$$

where the size of $\mathbf{U}^{\Gamma}_n$ is $M_n \times I_n$ and the rows of the $\mathbf{U}^{\Gamma}_n$ are assigned to the grid-lines of Γ. We have $\varepsilon = 0$ if the rank of $\mathcal{S}^{D,\Gamma}$ is equal to maximum rank constrain I_n on dimensions $n = 1..N$. If it has higher rank then we obtain $\varepsilon > 0$. However, we do not have $\mathcal{S}^{D,\Gamma}$ since its size and especially the computational load of HOSVD would be out of the regular computer capacity. Therefore, in order to avoid the execution of HOSVD on $\mathcal{S}^{D,\Gamma}$ we compute each rows of $\mathbf{U}^{\Gamma}_n$ one by one from $\mathcal{S}$ and $\mathbf{S}(\mathbf{p}(t))$. We determine the m_d-th row (assigned to the grid line γ_{d,m_d}) in matrix $\mathbf{U}^{\Gamma}_d$ in such a way that: let p_k be fixed to one θ_{k,h_k} of the the grid-lines in each dimensions $k=1..N, k\neq d$ of discretization grid Θ except the d-th dimension as:

$$p_k=\theta_{k,h_k}\quad k=1..N,\quad k\neq d,$$

where $h_k\in\{1,..,H_k\}$ is arbitrarily selected. We may simply select $h_k=1$ in practical cases if there is no special reason. Then we create vector $\mathbf{p}=\begin{pmatrix}\theta_{1,h_1} & \theta_{2,h_2} & \dots & \gamma_{d,m_d} & \dots & \theta_{N,h_N}\end{pmatrix}$. The new row $\mathbf{u}^{\Gamma}_{d,m_d}$ of matrix $\mathbf{U}^{\Gamma}_d$ is computed as:

$$\mathbf{u}^{\Gamma}_{d,m_d}=(\mathbf{S}(\mathbf{p}))_{(1)}\left(\left(\mathcal{S}\underset{k}{\otimes}\mathbf{u}^{\Theta}_{k,h_k}\right)_{(n)}\right)^{+},$$

where superscript "$(\cdot)^+$" denotes pseudo inverse, and $\mathbf{u}^{\Theta}_{k,h_k}$ is the h_k-th row vector of $\mathbf{U}^{\Theta}_k$. The first mode matrix $(\mathbf{S}(\mathbf{p}))_{(1)}$ of matrix $\mathbf{S}(\mathbf{p})$ is understood such that matrix $\mathbf{S}(\mathbf{p})\in\mathbb{R}^{O\times I}$ is considered as a three dimensional tensor $\mathcal{S}(\mathbf{p})\in\mathbb{R}^{1\times O\times I}$, where the length of the first dimension is one. $\mathbf{S}(\mathbf{p})_{(1)}$ practically means that the matrix $\mathbf{S}(\mathbf{p})$ is stored into one row vector by placing the rows of $\mathbf{S}(\mathbf{p})$ next to each other, respectively, see [19].

As result we have now a decomposition

$$\mathcal{S}^{D,\Gamma}\underset{\varepsilon}{\approx}\mathcal{S}\underset{n}{\otimes}\mathbf{U}^{\Gamma}_n,$$

in which the matrices $\mathbf{U}_n$ may not be the same as could be resulted by HOSVD in (8), however we kept the I_n rank constrain and yield the best approximation of the new rows in $\mathbf{U}_n$ in the sense of L_2 norm. This is guaranteed by the use of pseudo inverse. Note that if $\delta=0$ in (6) and $\mathcal{S}^{D,\Gamma}$ has the same rank as $\mathcal{S}^{D,\Theta}$ in each dimension then $\varepsilon=0$, for instance see the example in the next section.

Step 4) Generating different convex hulls

This step executes SN (Sum Normalization), NN (Non Negativeness), and NO (Normality) or INO-RNO (Inverse-NO and Relaxed-NO) transformations [6], [10], [15], [21], [22] according to the desired type of the convex hull. The previous step results in $\mathbf{U}^{\Gamma}_n$. The goal is here to transform matrices $\mathbf{U}^{\Gamma}_n$ to $\bar{\mathbf{U}}^{\Gamma}_n$, where the bar over matrix $\mathbf{U}$ means that the matrix is SN, NN NO or other type according to the executed transformation. Note that the Yam's type SN transformation [21], [22] needs the discarded singular values computed by the HOSVD. Since we do not have the singular values of $\mathcal{S}^{D,\Gamma}$ we use transformation techniques proposed in papers [6] and [15]. These techniques do not use any information about the result of HOSVD, but matrices $\mathbf{U}^{\Gamma}$ only. As a matter of fact when we transform matrices $\mathbf{U}^{\Gamma}_n$ to $\bar{\mathbf{U}}^{\Gamma}_n$ then we must modify $\mathcal{S}$ to $\bar{\mathcal{S}}$ accordingly, in order to keep:

$$\mathcal{S}\underset{n}{\otimes}\mathbf{U}^{\Gamma}_n=\bar{\mathcal{S}}\underset{n}{\otimes}\bar{\mathbf{U}}^{\Gamma}_n,$$

$\bar{\mathcal{S}}$ could be readily computed form a $\mathcal{S}^{D,\Gamma}$ as:

$$\bar{\mathcal{S}}=\mathcal{S}^{D,\Gamma}\underset{n}{\otimes}\left(\bar{\mathbf{U}}^{\Gamma}_n\right)^{+},$$

however, as mentioned we do not have $\mathcal{S}^{D,\Gamma}$, but we have $\mathcal{S}^{D,\Theta}$. Therefore, let us select those rows of $\bar{\mathbf{U}}^{\Gamma}_n$ which are also assigned to the sparse grid Θ and then store into matrix $\bar{\mathbf{U}}^{\Theta}_n$ (note that if $\bar{\mathbf{U}}^{\Gamma}_n$ is SN, NN, NO, RNO or INO-RNO type then $\bar{\mathbf{U}}^{\Theta}_n$ will also be SN, NN, NO, RNO or INO-RNO type, respectively). Thus, the tensor $\bar{\mathcal{S}}$ can be computed by:

$$\bar{\mathcal{S}}=\mathcal{S}^{D,\Theta}\underset{n}{\otimes}\left(\bar{\mathbf{U}}^{\Theta}_n\right)^{+}.$$

Finally we have

$$\mathcal{S}^{D,\Theta}\underset{\delta}{\approx}\bar{\mathcal{S}}\underset{n}{\otimes}\bar{\mathbf{U}}^{\Theta}_n=\mathcal{S}\underset{n}{\otimes}\mathbf{U}^{\Theta}_n.$$

then

$$\mathcal{S}^{D,\Gamma}\underset{\varepsilon}{\approx}\bar{\mathcal{S}}\underset{n}{\otimes}\bar{\mathbf{U}}^{\Gamma}_n=\mathcal{S}\underset{n}{\otimes}\mathbf{U}^{\Gamma}_n.$$

Step 5) Generating the continuous weighting functions

This step is equivalent with the last step of the original TP model transformation. It results in the continuous weighting functions such as:

$$\mathbf{S}(\mathbf{p}(t))\approx\bar{\mathcal{S}}\overset{N}{\underset{n=1}{\otimes}}\mathbf{w}_n(p_n(t)),\tag{9}$$

Step 6) Checking the accuracy of the resulting TP model

Finally we numerically check the accuracy of the resulting TP model over a huge number of points in Ω. We also check the SN, NN, NO, RNO and INO-RNO type of the weighting functions via numerical computation. In the case of exact transformation the best error we can achieve is about in the range of 10^{-13} that is caused by the numerical computation of the HOSVD.

B. Evaluation of the calculation complexity reduction

In this subsection we compare the computational complexity of the original and the modified TP model transformation. Since the computation problem is caused by the computational explosion of the HOSVD, we focus attention on the number of elements in $\mathcal{S}^D$ in both cases.

The original TP model transformation executes HOSVD on $\mathcal{S}^D$ with the size of $M_1\times M_2\times M_N\times O\times I$. The number of elements of $\mathcal{S}^D$ is $OI\prod_{n=1}^{N}M_n$. In practical cases M_n is about 100, see papers [4]–[9], [11]. The modified TP model transformation executes HOSVD on $\mathcal{S}^D$ with the size of $H_1\times H_2\times H_N\times O\times I$, where H_n is about $2-5$.

In higher dimension even the discretization step of the original TP model transformation fails, since the size of $\mathcal{S}^D$ requires memory array that is out of a regular computer.

The next section gives a 6 dimensional example when the original TP model transformation cannot be applied because of the extremely huge computational requirement, however the modified TP model transformation can easily be executed.

V. A 6 DIMENSIONAL NUMERICAL EXAMPLE

A. Numerical example

Consider the following dynamic model:

$$\dot{\mathbf{x}}=\mathbf{S}(\mathbf{p})\mathbf{x},$$

where

$$\mathbf{S}(\mathbf{p}) = \begin{bmatrix} s_{1,1} & 2 & 2 & 0 & s_{1,5} & 0 \\ 2 & 3+x_2^3 & 3 & 1 & 0 & 2 \\ 2 & \frac{x_4 * \sin(x_3)}{\cos(x_3)^2} & 0 & S_{3,4} & 0 & 3 \\ x_4^2 & 3 & 0 & \frac{1+x_3^3}{1-x_3^2} & 0 & 0 \\ 1 & 2 & 0.5 & 4 & 5 & 0 \\ 0 & 2 & 0 & 4 & 0 & s_{6,6} \end{bmatrix} \qquad (10)$$

$$s_{1,1} = x_1 + x_1^2 + x_1^3 + x_1^4 \qquad (11)$$

$$s_{1,5} = 3 + x_5^2 + x_5^4 \qquad (12)$$

$$s_{3,4} = 3\cos(x_2)\sin(x_2)^2 \qquad (13)$$

$$s_{6,6} = \frac{x_6^3 + x_6^2}{1-\sin(x_6)} + x_6^4\cos(x_6) + 4 \qquad (14)$$

Observe that the above system matrix $\mathbf{S}(\mathbf{p})$ non-linearly depends on all elements of the state vector $\mathbf{x} \in \mathbb{R}^6$. This means that $N = 6$ and $\mathbf{p} = \mathbf{x}$. Let the transformation space be $\mathbf{\Omega} : [a_1, b_1] \times [a_2, b_2] \times .. \times [a_N, b_N]$, where $\forall n : a_n = -0.5$ and $\forall n : b_n = -a_n$. First we apply the original TP model transformation and we define 129 grid lines on each dimension for discretization. The discretization result in $\mathcal{S}^D$ whose number of elements is $36 \cdot 129^6$. A regular software (for instance MATLAB) and a computer cannot store such a large tensor $\mathcal{S}^D$, and even in case we have a huge memory storage we cannot execute HOSVD on this tensor. In conclusion the original TP model transformation cannot be executed in the present case.

Let us apply the modified TP model transformation. We use the same $\mathbf{\Omega}$ and we require the same grid density with 129 grid-lines on each dimension. First we define the sparse discretization grid. Here we have to consider two constrains. The first one is that the HOSVD (in MATLAB) can be executed when the number of grid-lines by dimensions are about 3 in the present 6 dimensional case. The second restriction is that the computational load of LMI solver (to be executed in the case of controller design) explodes easily. Therefore, we restrict the computation of the TP model transformation to yield 3 weighting functions by dimensions, that means maximum $3^6 = 729$ LTI systems. Thus, we define discretization grid Θ as $\forall n : H_n = 3$. The discretization results in $\mathcal{S}^{D,\Theta}$ whose number of elements is only $36 \cdot 3^6$. When we execute HOSVD then we see the rank of $\mathcal{S}^{D,\Theta}$ by dimensions respectively are 2, 3, 3, 3, 2, 2. Hence, the sizes of the resulting $\mathbf{U}_n$, $n = 1..N$ are: 3×2, 3×3, 3×3, 3×3, 3×2 and 3×2. The size of $\mathcal{S}$ is $2 \times 3 \times 3 \times 3 \times 2 \times 2 \times 6 \times 6$. Figure 1 depicts the values of the columns of $\mathbf{U}_n$ (assigned to the discretization grid-lines) as the points of the discretized weighting functions. For visualization we connected the points by straight lines.

For Step 3 we define the dense grid Γ with 129 grid-lines on each dimension. The resulting weighting functions, disretized over 129 points, are depicted on Figure 2.

If we execute Step 5 directly here then we can generate the continuous weighting functions and check the error here numerically between the original and the resulting TP model over huge number of points. We find that the error is in the range of 10^{-13} that comes from the numerical computation of HOSVD. In conclusion we can say that the resulting TP model is exact and it includes minimum 216 LTI systems. As a next step we execute Step 4 and generate SN, NN and NO type weighting functions like in the case of original TP model transformation. By Step 5 we can generate the continuous weighting functions see Figure 3

We can conclude form the above example that the original TP model transformation cannot be applied in case of 6 dimensional problem because of its heavy computational load. The modified TP model transformation, however, can be executed on a regular PC and in the present example it yields an exact TP model.

B. Further comparison to the original TP model transformation

In order to compare the original and the modified TP model transformation we executed the modified TP model transformation on the dynamic models examined in papers [4]–[9], [11], [16]–[18] where the original TP model transformation was applied. We concluded that in all these cases both transformations resulted in numerically equivalent solutions. This came from the fact that in all these cases the exact TP model representation exists.

VI. Conclusion

This paper proposes a modified TP model transformation that requires considerable less computational effort than the original TP model transformation. The paper shows via a numerical example that the modified transformation is executable in higher dimensional cases while the original TP model transformation fails. The key idea of the computational complexity reduction is that the modified version uses a sparse grid (that fits the maximum available computational capacity for HOSVD) for discretization and the complexity trade-off, and expands the result to a significantly more dense discretization, in order to determining various types of convex hulls. Having this relaxed computational requirement we can significantly extend the applicability of the TP model transformation based control design framework for models with higher dimensionality.

References

[1] P. Baranyi, D. Tikk, Y. Yam, and R. J. Patton, "From differential equations to PDC controller design via numerical transformation," *Computers in Industry, Elsevier Science*, vol. 51, pp. 281–297, 2003.

[2] P. Baranyi, "TP model transformation as a way to LMI based controller design," *IEEE Transaction on Industrial Electronics*, vol. 51, no. 2, pp. 387–400, 2004.

[3] K. Tanaka and H. O. Wang, *Fuzzy Control Systems Design and Analysis - A Linear Matrix Inequality Approach.* Hohn Wiley and Sons, Inc., 2001.

[4] P. Baranyi, P. Korondi, R.J.Patton, and H. Hashimoto, "Global asymptotic stabilisation of the prototypical aeroelastic wing section via TP model transfromation," *Asian Journal of Control*, vol. 7, no. 2, pp. 99–111, 2004.

[5] P. Baranyi, "Tensor product model based control of 2-D aeroelastic system," *Journal of Guidance, Control, and Dynamics (in Press).*

[6] ——, "Output feedback control of 2-D aeroelastic system via TP model transformation," *Journal of Guidance, Control, and Dynamics (in Press).*

[7] P. Baranyi, Z. Petres, P. Varlaki, and P. Michelberger, "Observer and control law design to the TORA system via TPDC framewrok," *WSEAS Transactions on Systems*, vol. 1, pp. 156–163, 2005.

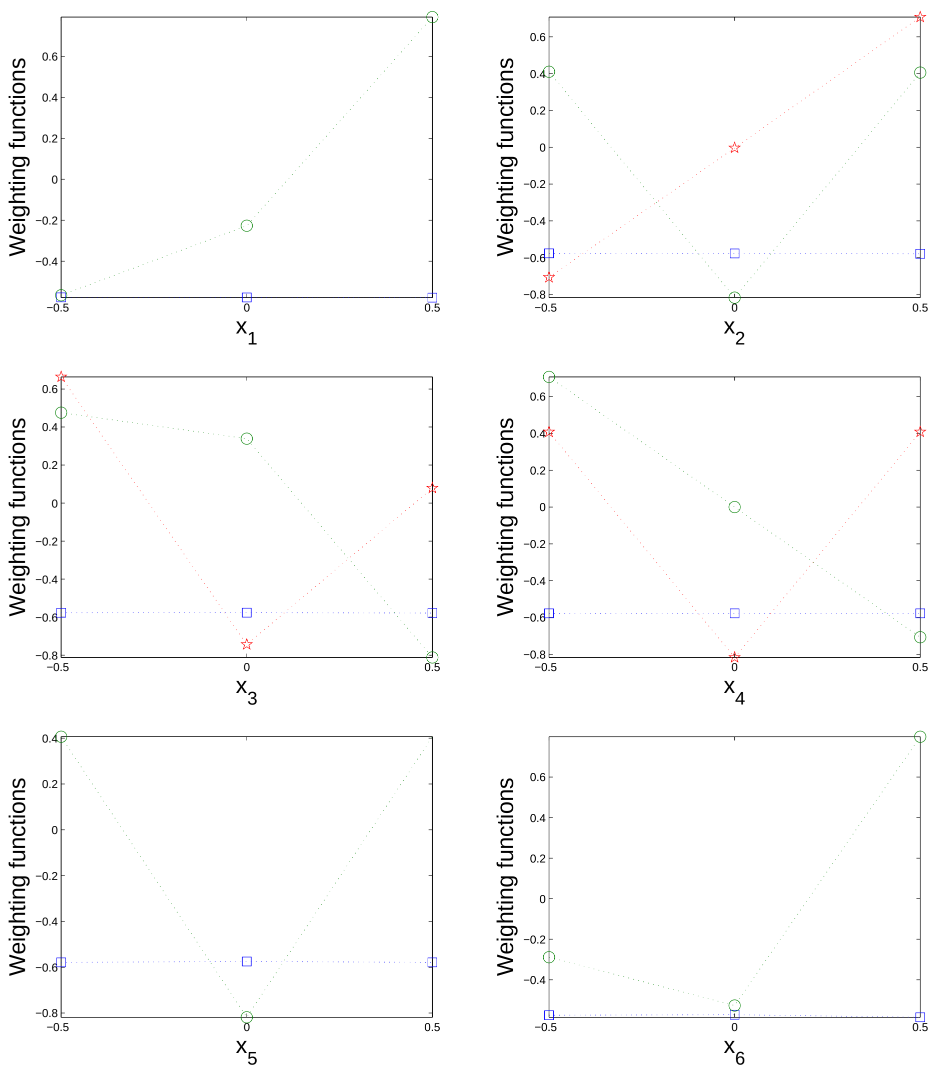

Fig. 1. Discretised weighting functions over Θ.

[8] P. Baranyi, A. Varkonyi-Koczy, P. Varlaki, P. Michelberger, and R. Patton, *Singular Value Based Model Approximation*, section in n. mastorakis (ed.) problems in applied mathematics and computational intelligence, mathematics and computers in sci. and eng. ed. World Scientific and Engineering Society Press, Danvers, 2001.

[9] G. Hancke and A. Szeghegyi, "Application study of the TP model transformation in the control of an inverted pendulum," in *International Conference on Computational Cybernetics (ICCC)*, Siófok, Hungary, 2003.

[10] J. Bokor, P. Baranyi, P. Michelbereger, and P. Varlaki, "Tp model transformation in non-linear system control," in *3rd IEEE International Conference on Computational Cybernetics (ICCC)*, Mauritius, Greece, 13-16 April 2005, pp. 111–119, (plenary lecture) ISBN: 0-7803-9474-7.

[11] Z. Petres, B. Resko, and P. Baranyi, "Reference signal control design of the TORA system: a TP model transformation approach," in *IEEE Int. Conf. Fuzzy Systems*, Budapest, Hungary, 2004, p. Proc. on CD.

[12] Y.Yam and P.Baranyi, "On application of grid point sampling and SVD consolidation approach," in *IEEE Int. Conf. Fuzzy Systems*, Budapest, Hungary, 2004, p. Proc. on CD.

[13] D. Tikk and P. Baranyi, "On the approximation properties of TP model forms," in *IEEE Int. Conf. Fuzzy Systems*, Budapest, Hungary, 2004, p. Proc. on CD.

[14] P. Baranyi, "Global asymptotic stabilization of the aeroelastic wing section: a TP model transformation based approach," in *IEEE Int. Conf. Fuzzy Systems*, Budapest, Hungary, 2004, p. Proc. on CD.

[15] P. Várkonyi, D. Tikk, P. Korondi, and P. Baranyi, "A new algorithm for RNO-INO type tensor product model representation," in *IEEE 9th International Conference on Intelligent Engineering Systems*, Athens, Greece, 16-19 September 2005, pp. 263–266, iSBN: 0-7803-9474-7.

[16] Z. Petres, P. L. Várkonyi, P. Baranyi, and P. Korondi, "Different affine decomposition of the TORA system by TP model transformation," in *IEEE 9th International Conference on Intelligent Engineering Systems*, Athens, Greece, 16-19 September 2005, pp. 93–98.

[17] B. Reskó, A. Róka, Z. Petres, and P. Baranyi, "Visual cortex inspired intelligent contouring," in *IEEE 9th International Conference on Intelligent Engineering Systems*, Athens, Greece, 16-19 September 2005, pp. 47–52.

[18] P. Baranyi, P. L. Várkonyi, and P. Korondi, "Different affine decomposition of the model of the prototypical aeroelastic wing section by TP model transformation," in *IEEE 9th International Conference on Intelligent Engineering Systems*, Athens, Greece, 16-19 September 2005, pp. 105–110.

[19] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, "A multi linear singular value decomposition," SIAM *Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.

[20] D. Tikk, P. Baranyi, R. Patton, and J. Tar, "Approximation capability of tp model forms," *Australian Journal of Intelligent Information Processing Systems (accepted for publication)*, 2004.

[21] Y. Yam, P. Baranyi, and C. T. Yang, "Reduction of fuzzy rule base via singular value decomposition," *IEEE Transaction on Fuzzy Systems*, vol. 7, no. 2, pp. 120–132, 1999.

[22] Y. Yam, C. T. Yang, and P. Baranyi, *Singular Value-Based Fuzzy Reduction with Relaxed Normalization Condition*, interpretability issues in fuzzy modeling, J. Casillas, O. Cordón, F.Herrera, L.Magdalena

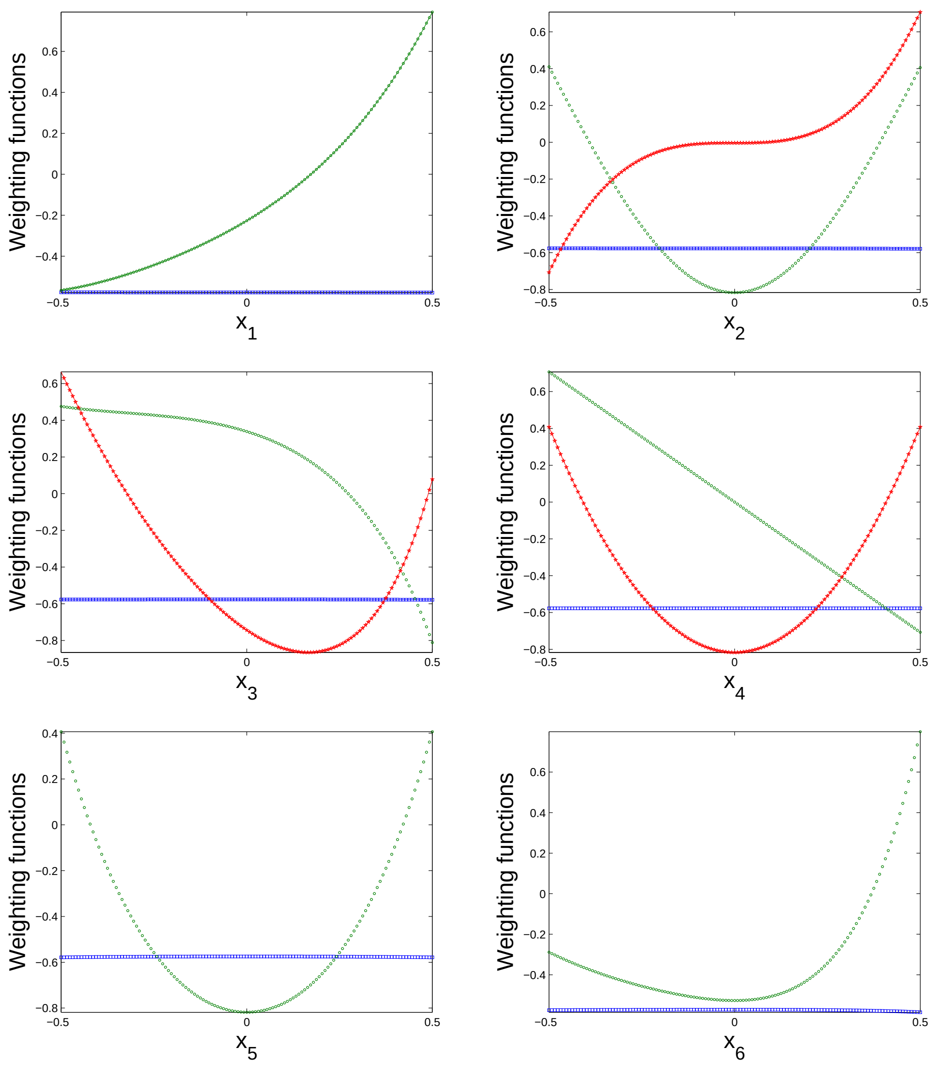

Fig. 2. Discretized weighting functions over Γ.

(Eds.) ed., ser. Studies in Fuzziness and Soft Computing. Springer-Verlag, 2003, vol. 128.

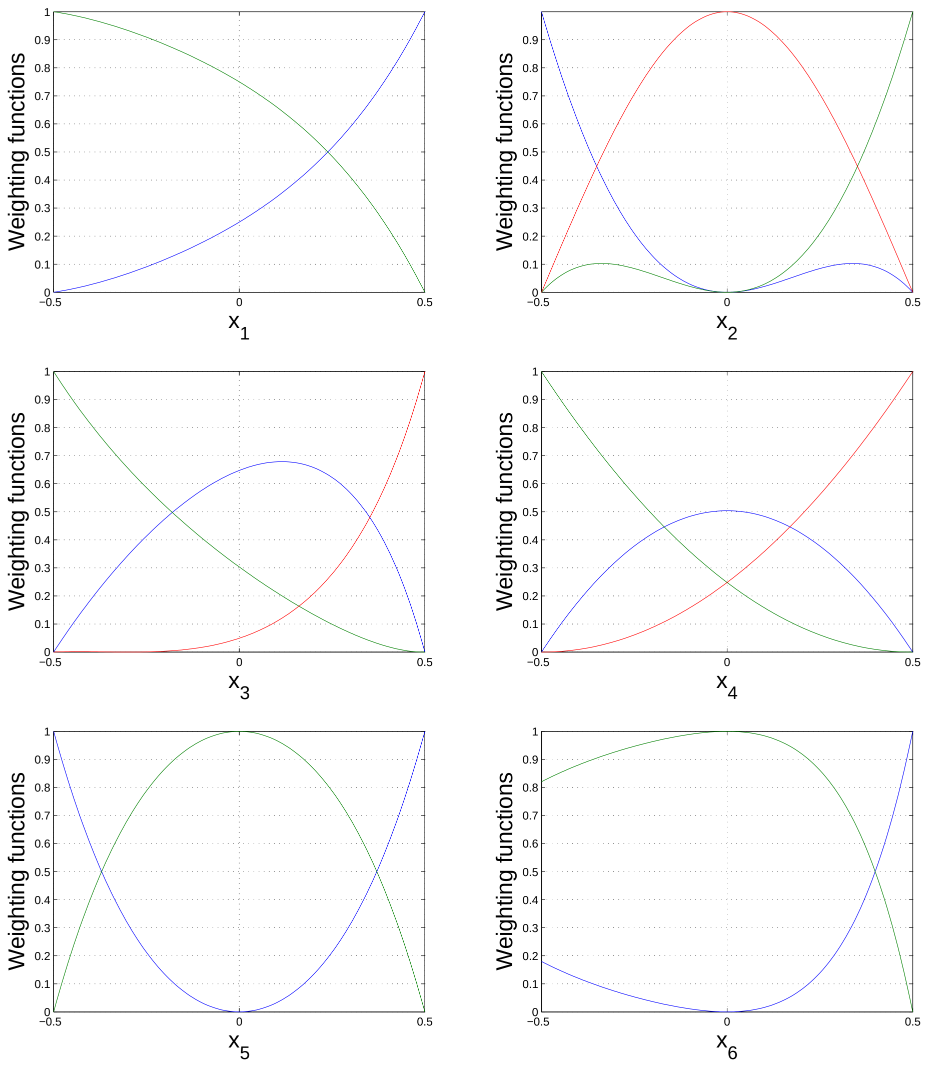

Fig. 3. Discretized weighting functions over Γ.