Research Article

Extension of the Multi-TP Model Transformation to Functions with Different Numbers of Variables

Péter Baranyi

Széchenyi István University, Gyor, Hungary

Correspondence should be addressed to Péter Baranyi; prof.peter.baranyi@gmail.com

Received 18 September 2017; Accepted 29 January 2018; Published 19 March 2018

1. Introduction

The appearance of the Singular Value Decomposition (SVD) was one of the largest breakthroughs in matrix algebra [1]. Its applicability was extended to tensors in the form of the Higher-Order SVD (HOSVD) of functions. It plays the same role with respect to functions as HOSVD does for tensors (and SVD for matrices). The need for certain advantageous features, such as rank/complexity reduction, trade-offs between complexity and accuracy, and a manipulation power representative of the TP form, has motivated novel concepts in TS fuzzy model based modelling and control. The latest extensions of the TP model transformation, called the multi- and generalised TP model transformations, are applicable to a set functions where the dimensionality of the outputs of the functions may differ, but there is a strict limitation on the dimensionality of their inputs, which must be the same. The paper proposes an extended version that is applicable to a set of functions where both the input and output dimensionalities of the functions may differ. This makes it possible to transform complete multicomponent systems to TS fuzzy models along with the above-mentioned advantages.

The tensor product (TP) model transformation defines and numerically reconstructs the Higher-Order Singular Value Decomposition (HOSVD) of functions. It plays the same role with respect to functions as HOSVD does for tensors (and SVD for matrices). The need for certain advantageous features, such as rank/complexity reduction, trade-offs between complexity and accuracy, and a manipulation power representative of the TP form, has motivated novel concepts in TS fuzzy model based modelling and control. The latest extensions of the TP model transformation, called the multi- and generalised TP model transformations, are applicable to a set functions where the dimensionality of the outputs of the functions may differ, but there is a strict limitation on the dimensionality of their inputs, which must be the same. The paper proposes an extended version that is applicable to a set of functions where both the input and output dimensionalities of the functions may differ. This makes it possible to transform complete multicomponent systems to TS fuzzy models along with the above-mentioned advantages.

The approximation power of the TP model transformation applied to TS fuzzy models is investigated in [9].

The above-mentioned extensions and variations of the TP model transformation were primarily applied to fuzzy model complexity reduction [10, 11] and in the widely used TS fuzzy model based PDC (Parallel Distributed Compensation) control theories [12–14]. But also, in general, it has been applied to polytopic model, TP/TS fuzzy model, and LMI (Linear Matrix Inequality [15]) based control theories. The most important features of the TP model transformation are guaranteed by the key transformation step whereby a numerically reconstructed HOSVD structure is determined. Key features of the transformation are as follows:

(i) It is executable on models given by equations or soft computing based representations, such as fuzzy rules or 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, see Section 4, Step 1).

(ii) It will find the minimal complexity, namely, the minimal number of rules of the TS fuzzy model. If further complexity reduction is required, it provides one of the best trade-offs between the number of rules and approximation error.

(iii) It works like a principle component analysis, in that it determines the order of the components/fuzzy rules according to their importance.

(iv) It is capable of deriving the 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.
(v) It is capable of transforming the given model to predefined antecedent fuzzy sets (pseudo-TP model transformation).

(vi) It is capable of transforming a set of models simultaneously, while common antecedent fuzzy sets are derived for all models.

Based on the above, various theories and applications have emerged using the TP model transformation. Further computational improvements were proposed in [16, 17]. It has been proved in [5, 18–20] that LMI-based control design theories are very sensitive for convex hulls defined by consequents (vertices) of TS fuzzy models. Thus, 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 [21–23]. Further useful control approaches and applications were published in the field of control theory [24–41]. Many powerful approaches are published on the field of sliding mode control in [29, 42, 43]. In physiological control the usability of TP model transformation has been demonstrated as well [44–49]. Various further theories and applications are studied in [50–87].

One of the key advantages of the TP model transformation is that it is capable of finding the minimal complexity of all components of the system and guarantees the same antecedent system for all components. This is a very typical requirement in design or stability verification methodologies, that is, the model, controller, and observer need to have the same antecedent system, hence, convex representation. Therefore, the simultaneous manipulation of the components with the multi-TP model transformation or the generalised TP model transformation (that combines all variants of the TP model transformation) yields further possibilities for control performance optimisation [18–20].

Despite the above advantages, a crucial limitation of the generalised TP model transformation is that it can only be applied to a set of systems which have the same number of inputs. For instance, consider four different systems given with different representations, as shown in Figure 1. S1 is a fuzzy logic model; S2 is neural network; S3 is given by an equation; and S4 is a black-box model. All of these models have the same inputs but may have different sized output tensors. The multi-TP model transformation is capable of simultaneously transforming all systems to TP or TS fuzzy model form, such that the same antecedent sets are defined on the inputs. The generalised TP model transformation can also transform to predefined antecedent fuzzy sets.

A further generalisation proposed in this paper can be applied to systems like in the example given in Figure 2. Here each system may be given by different representations (like in the above case) but may also have different numbers of inputs. The transformation can simultaneously convert all of the systems to TS fuzzy model form, such that the antecedent fuzzy sets will either be the same or assume a predefined structure. From all other perspectives, the proposed TP model transformation inherits all of the advantageous features of the previous TP-based approaches.

Recently proposed SOS-type (Sum-of-squares) TS fuzzy LPV models are also widely applied in fuzzy control theories [88, 89]. The further extension of the TP model transformation to such systems is highly welcome in future works.

2. Notation and Concepts

2.1. Notation. The following notations are used in the paper:

(i) Scalar: $a$ is scalar.

(ii) Vector: $a$ contains elements $a_i$.

(iii) Matrix: $A$ contains elements $a_{ij}$.

(iv) Tensor: $A$ contains elements $a_{ijk,...}$.

(v) Set: $A = \{a, b, c, \ldots\}$, for example, $a \in A$.

(vi) Index $i$: the upper bounds of the indices are denoted by the uppercase letter, for example, $I$.

(vii) Index $i \in I$ denotes that index $i$ takes the elements of set $I \subseteq \{1, 2, \ldots, l\} \subset \mathbb{N}$, respectively. $I : \{1, 2, \ldots, l\}$ is understood as per default.

(viii) Interval: $\omega = [\omega_{\min}, \omega_{\max}]$. 
(ix) Space: \( \Omega : \omega_1 \times \omega_2 \times \cdots \times \omega_N \) is an \( N \) dimensional hypercube.

(x) \( \mathbf{x} \in \Omega \) expresses the fact that vector \( \mathbf{x} \) is within the space \( \Omega \). The dimensions of \( \mathbf{x} \) and \( \Omega \) are the same.

(xi) \( \Theta \subseteq \Omega \) denotes a dimensionality reduced subset in general as follows:

(a) In the case of spaces: \( \Theta \subseteq \Omega \) states that \( \Theta \) is a hypercube with the same sized intervals as \( \Omega \), but has a smaller number of dimensions.

(b) In the case of vectors: \( \mathbf{a} \subseteq \mathbf{b} \), where \( \mathbf{a} \in \Theta \subseteq \mathbb{R}^N \) and \( \mathbf{b} \in \Omega \subseteq \mathbb{R}^M \) means that \( N < M \) and \( \Theta \subseteq \Omega \).

(c) In the case of tensors: \( \mathcal{A} \subseteq \mathcal{B} \) means for instance that \( \mathcal{A} \) is obtained by deleting complete dimensions from tensor \( \mathcal{B} \).

(xii) Grid: \( \mathcal{G} : \mathcal{G}_1 \times \mathcal{G}_2 \times \cdots \times \mathcal{G}_N \) is a rectangular hypergrid (tensor), where \( \mathcal{G}_n = \{g_{n,1} < g_{n,2} < \cdots < g_{n,m_n}\} \) defines the locations of the \( m_n \) different grid points in increasing order.

(xiii) Pair \( (\Omega, \mathcal{G}) \): space \( \Omega \in \mathbb{R}^N \) and grid \( \mathcal{G} \) are in a pair, meaning that \( \forall n \in N : g_{n,1} = \omega_{n,\min} \) and \( g_{n,m_n} = \omega_{n,\max} \).

(xiv) Discretisation function \( \mathcal{F}^{D}_{\mathcal{G}}(\Omega, \mathcal{G}) \) of \( f(\mathbf{x}) \) denotes the sampling of \( f(\mathbf{x}) \) over pair \( (\Omega, \mathcal{G}) \). Thus, it is a tensor with the size of \( M_1 \times M_2 \times \cdots \times M_N \) and entries:

\[
f_{m_1,m_2,\ldots,m_N}(\mathbf{x}) = f((g_{1,m_1},g_{2,m_2},\ldots,g_{N,m_N})). \tag{1}
\]

(xv) \( \mathcal{A} \otimes \mathcal{B} \) is the tensor product (TP); for details, refer to [4, 5, 8]. A slight difference in notation here is that \( \mathcal{N} \) under the tensor product operation \( \otimes \) is only a set numbers \( n \in \mathbb{N} \) to which the product should be applied.

(xvi) \( f(\mathbf{x}) = \mathcal{A} \otimes w_a(x_n) \) represents the TP function, \( \mathbf{x} \in \mathbb{R}^N \), where \( w_a(x_n) = \left[w_{n,1}(x_n),w_{n,2}(x_n),\ldots,w_{n,m_n}(x_n)\right] \) is called the weighting function system.

(xvii) Types of the weighting functions are as follows:

(a) SN: sum normalised
(b) NN: nonnegativeness
(c) NO: normalised
(d) CNO: close to normalised
(e) RNO: relaxed normalised
(f) INO: inverse normalised
(g) IRNO: inverse relaxed normalised.

For further details, refer to [4, 5].

3. The Proposed TP Model Transformation

Assume that a set of functions is given as \( \mathcal{Y}_1 = f_1(\mathbf{x}_i), \mathbf{x}_i \subseteq \mathbf{x} \in \Omega \); thus \( \mathbf{x}_i \in \Omega_i \subseteq \Omega \subseteq \mathbb{R}^N, \mathbf{x}_i \in \mathbb{R}^{N_i} \). The output tensor \( \mathcal{Y}_1 \) of each function \( f_i(\mathbf{x}_i) \) may differ in the number of dimensions and its size as \( \mathcal{Y}_1 \in \mathbb{R}^{O_1 \times O_2 \times \cdots \times O_{K_1}} \), where \( O_1 \) denotes the number of dimensions of the output and \( O_2 \) denotes the number of elements in dimension \( k \).

The goal of the TP model transformation is to transform \( \forall i : f_i(\mathbf{x}_i) \) into TP function form as

\[
\mathcal{Y}_1 = f_i(\mathbf{x}_i) = \mathcal{S}_1 \otimes w_{\mathcal{S}_1}(x_n) \tag{2}
\]

under the following constraints given on the weighting functions.

(i) Unified Constraints for \( \forall i : f_i(\mathbf{x}_i) \). All resulting TP functions will have the same weighting function system on each dimension defined by the set \( V \subseteq \mathbb{N} \) (obviously, if the function has that input dimension):

(a) Weighting function systems \( w_i(x_n), a \in A \subseteq \mathbb{V} \), are predefined.

(b) Weighting function systems \( w_i(x_n), b \in B \subseteq \mathbb{V} \) will be derived by the transformation; only their types are predefined (i.e., SN, NN, NO, CNO, RNO, INO, and IRNO). Further the number of the weighting functions are minimised.

(ii) Different Constraints for Each \( f_i(\mathbf{x}_i) \). The resulting TP functions have different weighting functions on dimensions \( Z \subseteq \mathbb{N} \):

(a) Weighting function systems \( w_{i,c}(x_n), c \in C \subseteq Z \) are predefined for each \( f_i(\mathbf{x}_i) \).

(b) The types (i.e., SN, NN, NO, CNO, RNO, INO, and IRNO) of the weighting function systems \( w_{i,d}(x_n) \) are predefined for dimensions \( d \in D \subseteq Z \) of each \( f_i(\mathbf{x}_i) \).

Thus, (2) can be given as follows:

\[
\mathcal{Y}_1 = f_i(\mathbf{x}_i) = \mathcal{S}_1 \otimes w_{\mathcal{S}_1}(x_n) \otimes w_{\mathcal{C}_1}(x_n) \otimes w_{\mathcal{D}_1}(x_n). \tag{3}
\]

4. The Computation of the Proposed TP Model Transformation

Step 1 (discretisation).

(i) Discretisation of all \( f_i(\mathbf{x}_i) \) results in tensor \( \mathcal{F}^{D}_{\mathcal{G}_i}(\Omega_i, \mathcal{G}_i) \), \( (G_i \subseteq G) \). The size of \( \mathcal{F}^{D}_{\mathcal{G}_i}(\Omega_i, \mathcal{G}_i) \) in dimension \( n \in N_i \) is \( M_i \).

(ii) Discretise the predefined weighting functions over the dimensions of \( \Omega \):

\[
\mathcal{F}^{D}_{\mathcal{G}_i}(\mathcal{S}_{i,a}, \mathcal{S}_{i,b}), \quad \mathcal{F}^{D}_{\mathcal{G}_i}(\mathcal{C}_{i,c}, \mathcal{D}_{i,d}). \tag{4}
\]

Remark 1. This step is executed in the same way as in the case of the original TP model transformation; see [4, 5, 8].

Step 2 (defining TP structures). Execute the following steps in each dimension \( n \in \mathbb{N} \):

(i) Lay out tensors \( \mathcal{F}^{D}_{\mathcal{G}_i}(\Omega_i, \mathcal{G}_i) \) in dimensions \( n \) if vector \( \mathbf{x}_i \) has the following dimension:

\[
\mathbf{H}_i = \left( \mathcal{F}^{D}_{\mathcal{G}_i}(\Omega_i, \mathcal{G}_i) \right)_{(n)}. \tag{5}
\]
(ii) If \( n \in B \) then create
\[
T_n = [H_1 \quad H_2 \cdots H_L].
\] (6)
Execute SVD on \( T_n \) and SN, NN, NO, CNO and complexity trade-off by discarding singular values in the same way as in the original TP model transformation, which results in
\[
T_n = U_n D_n V_n^T.
\] (7)
As a matter of fact, if nonzero singular values are discarded then it is only an approximation. Let
\[
\psi_n^{(D(\omega, \Omega): \mathbb{R}^n)} = U_n.
\] (8)

(iii) If \( n \in D \) then execute SVD on \( H_j \) as
\[
H_j = U_j D_j V_j^T;
\] (9)
and, according to the conditions, execute SN, NN, NO, CNO, and complexity trade-off by discarding singular values in the same way as in the original TP model transformation:
\[
H_j = U_j' D_j' V_j'^T.
\] (10)
Again, if nonzero singular values are discarded then it is only an approximation. Let
\[
\psi_n^{(D(\omega, \Omega): \mathbb{R}^n)} = U_j'.
\] (11)

(iv) Finally,
\[
\delta_j = \mathcal{F}_I(D(\Omega, \Omega)) \boxtimes K_n \boxtimes K_{j,n},
\] (12)
where
\[
K_n = (\psi_n^{D(\omega, \Omega): \mathbb{R}^n})^+, \quad K_{j,n} = (\psi_n^{D(\omega, \Omega): \mathbb{R}^n})^+,
\] (13)
where \((\cdot)^+\) denotes the pseudoinverse.

**Step 3** (reconstruction of the weighting functions). This step is the same as in the multi-TP model transformation [4, 5, 8]. Having the result of the above steps, \( \mathcal{F}_I(D(\Omega, \Omega)) \) and \( \delta_j \), we can recalculate the weighting functions at any point. We may calculate the first two steps over a grid, which is not too dense, but calculate the weighting function over a very dense grid (as suggested in [5]), and then construct piecewise linear functions. As a result we have \( w_n(x_n) \) and \( w_{j,n}(x_j) \).

Then we achieved the goal. We have the TP model form of all functions with the given constraints:
\[
f_1(x) = \delta_j \boxtimes w_{j,n}(x_n),
\] (14)
or
\[
f_1(x) = \delta_j \boxtimes w_{j,n}(x_n),
\] (15)
if a complexity trade-off is executed (nonzero singular values are discarded), where
\[
\forall l : w_{j,a} (x_l) = w_a (x_l),
\] (16)
Or in other words,
\[
f_1(x) = \delta_j \boxtimes w_{a}(x_n) \boxtimes w_b(x_n) \boxtimes w_{c}(x_n) \boxtimes w_{d}(x_n) \boxtimes w_{i,a}(x_l). \] (17)

Remark 2. The convex hull manipulation and the complexity trade-off are done in the second stage. Therefore the approximation accuracy is controlled here by the discarded nonzero singular values. However, the discarded nonzero singular values lead to approximation error. If the given weighting function system is not sufficient (i.e., the number of the weighting functions is less than the rank of that dimension) then we arrive at an approximation only. The use of the pseudoinverse guarantees, however, that it will be the best approximation.

### 5. Example

5.1. The System. Consider a multicomponent system with input vector \( x = [x_1, x_2, x_3] \), where \( x \in \Omega = \omega_1 \times \omega_2 \times \omega_3 \in \mathbb{R}^3 \) and \( \omega_n = [0, 1] \). The system has four subsystems, \( l \in L = \{1, \ldots, 4\} \) as shown in Figure 1.

**System 3.** In order to have a systematic notation, we denote the input vector of System 3 as \( x_i = x \) that is \( x_i \in \Omega_1 = \Omega \). It is a neural network; see Figure 3:
\[
y_1 = f_a \left( \sum_{i=1}^{\infty} f_a(x_i) b_{i,j} \right),
\] (18)
\[
y_2 = f_a \left( \sum_{i=1}^{\infty} f_a(x_i) b_{i,j} \right),
\] (19)
where \( f_a(z) \) is the activation function (let it be a very simple one in the present case: \( f_a(z) = z \)) of the neurons and \( b_{i,j} \) are the weights connecting the \( i \)th input neuron to the \( j \)th output neuron. Thus the output of the system is
\[
y_1 = f_1(x_1) = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}.
\] (19)

**System 4.** The input vector of System 4 is \( x_2 = [x_2, x_3] \in \Omega_2 = \omega_2 \times \omega_3 \), where \( x_2 \in \mathbb{R}^2 \) and \( \Omega_2 \subset \Omega \).
This system is given by formulas such as

\[
Y_2 = f_2(x_2) = \begin{bmatrix} x_2^2 & 2x_3 \\ 3x_3^{-1} & 5 \end{bmatrix}.
\]  

System 5. The input vector of System 5 is \( x_3 = x_1 \in \Omega_3 = \omega_1 \), where \( x_3 \subset x \) and \( \Omega_3 \subset \Omega \).

This is given by a fuzzy logic model. Assume that two rules are given \((i = 1, 2)\):

\( \text{IF } A_i \text{ THEN } B_i \).

Further assume that the membership functions are in Ruspini partition:

\[
\mu_{A_1}(x_3) = x_3; \quad \mu_{A_2}(x) = 1 - x_3;
\]

and the consequent sets are singleton sets located at elements 5 and 6 of the output universe. It is a TS fuzzy model and, therefore, the transfer function (product-sum-gravity) of the model is

\[
y_3 = f_3(x_3) = 5x_3 + 6(1 - x_3).
\]

System 6. The input vector of System 6 is \( x_4 = [x_1 \ x_3] \in \Omega_4 = \omega_1 \times \omega_3 \), where \( x_4 \subset x \) and \( \Omega_4 \subset \Omega \).

This is a black-box model that can provide \( y_4 \) for any input \( x_1, x_3 \).

\[
y_4 = f_4(x_4).
\]

(In order to follow all computational steps of the example, let us reveal what is the output of the black-box model \( y_4 = 3x_1 + x_3 \).)

5.2. Conditions of the TP Model Transformation. The goal of the example is to transform all the four systems to TS fuzzy representations (or TP model if the resulting weighting functions cannot be represented as antecedent fuzzy states), with the following conditions:

(i) All systems must have the same antecedent function system on the input interval of \( x_1 \). The antecedent functions must be in Ruspini partition, namely, in SN and NN type. In order to have a complexity minimised representation, a further requirement is that the number of antecedent functions must be minimal.

(ii) The same antecedent function system of variable \( x_2 \) is predefined for all systems:

\[
w_2(x_2) = [w_{2,1}(x_2) \ w_{2,2}(x_2) \ w_{2,3}(x_2)],
\]

where “\( p \)” denotes “predefined,” and

\[
w_{2,1}(x_2) = 0.5x_2^2; \\
w_{2,2}(x_2) = 0.5(1 - x_2)^2; \\
w_{2,3}(x_2) = 1 - (w_{2,1}(x_2) + w_{2,2}(x_2)).
\]

(iii) The only requirement for the weighting function system of the input \( x_3 \) of each system is that they must be the singular functions of the HOSVD canonical form (orthonormed system ordered by the higher-order singular values). These functions are not representable as antecedent functions of fuzzy sets, since they may take negative values as well. Obviously they will not be the same for all systems.

5.3. Execution of the Proposed TP Model Transformation. It is worth emphasizing again that the previous methods for TP model representation cannot be applied in the present case, since the elements of the input vectors are different.

Step 1.

(i) Let us define grid \( G \) to \( \Omega = [0, 1] \times [0, 1] \times [0, 1] \):

\[
g_1 = g_2 = g_3 = [0 \ 0.01 \ 0.02 \ \cdots \ 1].
\]

Thus the number of points on the discretisation grid is \( M_1 = M_2 = M_3 = 101 \).

(ii) Let us discretise the systems over the rectangular grid defined by vectors \( g_n, n = 1, 2, 3 \). The discretisation of System 1 results in \( \mathcal{F}_1^{D(\Omega, G_1)} \), \( l \in L \). In case of System 3,

\[
\mathcal{F}_3^{D(\Omega, G_1)} \in \mathbb{R}^{M_1 \times M_2 \times M_3 \times 2},
\]

where the first three dimensions are assigned to the input variables and the last dimension is assigned to the output vector. The discretisation of System 4 yields the following vector:

\[
\mathcal{F}_4^{D(\Omega, G_1)} \in \mathbb{R}^{M_1 \times 2 \times 2},
\]

where the first two dimensions are assigned to the input variables \( x_2, x_3 \) and the last two dimensions are assigned to the output matrix. The discretisation of System 5 yields the following vector:

\[
\mathcal{F}_5^{D(\Omega, G_1)} \in \mathbb{R}^{M_1},
\]

The discretisation of System 6 results in

\[
\mathcal{F}_6^{D(\Omega, G_1)} \in \mathbb{R}^{M_1 \times M_2},
\]

where the first two dimensions are assigned to the input variables \( x_1, x_3 \) and the last dimension is assigned to the output vector.

Let us discretise the predefined weighting function as well:

\[
\mathcal{F}_2^{D(\mathbb{R}^2 \times \mathbb{R})} = \begin{bmatrix} w_{2,1}(g_{2,1}) & w_{2,2}(g_{2,1}) & w_{2,3}(g_{2,1}) \\ \vdots & \vdots & \vdots \\ w_{2,1}(g_{2,M_2}) & w_{2,2}(g_{2,M_2}) & w_{2,3}(g_{2,M_2}) \end{bmatrix}.
\]
Step 2.

(i) Dimension $x_1$

Lay out tensors $F_l^{D(\omega_1, G_l)}$, $l \in \{1, 3, 4\}$ in the dimension assigned $x_1$:

$$H_1 = (F_1^{D(\omega_1, G_1)})_{(3), i}.$$  \hfill (32)

Create

$$K = [H_1 \quad H_3 \quad H_4].$$ \hfill (33)

Execute SVD on $K$ incorporating SN and NN condition [4] (only nonzero singular values are kept):

$$K = U_1D_1V_1^T.$$ \hfill (34)

The result of this step to be used later is $U_1$.

(ii) Dimension $x_2$

Let

$$U_2 = F_2^{D(\omega_2, G_3)}.$$ \hfill (35)

(iii) Dimension $x_3$

Lay out tensors $F_l^{D(\omega_1, G_l)}$, $l \in \{1, 2, 4\}$ in the dimension assigned $x_3$:

$$H_1 = (F_1^{D(\omega_1, G_1)})_{(3), i}.$$ \hfill (36)

Then execute HOSVD on each $H_i$, $l \in \{1, 2, 4\}$ (only the non-zero singular values are kept):

$$H_1 = U_{3,1}D_1V_1^T.$$ \hfill (37)

$$H_2 = U_{3,2}D_2V_2^T.$$ \hfill (38)

$$H_4 = U_{3,4}D_4V_4^T.$$ \hfill (39)

The result of this step is $U_{3,1}$.

(iv) Reconstructing the core tensors

$$\forall l : \delta_l = F_1^{D(\omega_1, G_1)} \otimes (U_{3,1})^T.$$ \hfill (40)

Step 3. Let

$$F_1^{D(\omega_1, G_1)} = U_1,$$ \hfill (41)

$$F_3^{D(\omega_1, G_3)} = U_{3,1},$$ \hfill (42)

$$l \in \{1, 2, 4\}.$$

Then having the discretised tensors and weighting functions of all systems we can numerically reconstruct the weighting functions [4, 5] as

$$F_1^{D(\omega_1, G_1)} = U_1 \rightarrow w_1(x_1),$$ \hfill (43)

$$F_3^{D(\omega_1, G_3)} = U_{3,1} \rightarrow w_{31}(x_3),$$ \hfill (44)

$$l \in \{1, 2, 4\}.$$

Thus, we have achieved our goal:

$$f_1(x_1, x_2, x_3) = \delta_1 x_1 w_1(x_1) \times_2 w_2^T(x_2) \times_3 w_{1,3}(x_3);$$ \hfill (45)

$$f_2(x_2, x_3) = \delta_2 x_2 w_2^T(x_2) \times_3 w_{2,3}(x_3);$$ \hfill (46)

$$f_3(x_1) = \delta_3 x_1 w_1(x_1);$$ \hfill (47)

$$f_4(x_1, x_3) = \delta_4 x_1 w_{1,3}(x_1) \times_3 w_{4,3}(x_3).$$ \hfill (48)

6. Conclusion

The proposed TP model transformation can be executed on a set of models where the dimensionality of the inputs may differ. The proposed TP model transformation has all the advantages of the previous ones, including easy convex hull manipulation, complexity trade-offs, pseudo TP model transformation, and automatic and numerical execution.

Conflicts of Interest

The author declares that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the FIEK Program (Center for Cooperation between Higher Education and the Industries at the Széchenyi István University, GINOP-2.3.4-15-2016-00003).

References


A. Hajiloo, Robust and multi-objective model predictive control design for nonlinear systems [Ph.D. thesis], Concordia University, 2016.


