Chapter 2
Algorithms of the TP Model Transformation

Abstract This chapter proposes the generalized TP model transformation that includes various TP model manipulation facilities into one conceptuar framework. The generalized TP model transformation includes extensions such as the HOSVD and quasi HOSVD canonical form, the Bilinear-, Multi, Pseudo, and convex TP model transformation which all serves the goal to have a Transformation technique that is capable of freely manipulating all components of the TP model according to various conditions.

Keywords Bi-linear- • Pseudo- • Muti- • Generalised TP model transformtion

2.1 Original TP Model Transformation

This section recalls the TP model transformation from [1, 3, 6] and restructures it in order to have a core algorithm that can readily assume further extensions to be introduced in the chapter.

Definition 2.1 (Discretization Space Ω). Ω is a space in which we intend to perform the discretization of a given function $f(\mathbf{x})$.

Definition 2.2 (Discretized Function). Tensor $\mathcal{F}^{D(\Omega,G)} \in \mathbb{R}^{G_1 \times \cdots \times G_N \times O_1 \times \cdots \times O_K}$ is the discretized variant of function $\mathcal{Y} = f(\mathbf{x}) \in \mathbb{R}^{O_1 \times \cdots \times O_K}$ in the discretization space Ω, and over the discretization grid $G = G_1 \times \cdots \times G_N$ (G_n denotes the number of gridpoints on dimensions $n \in \mathrm{N}$). Vector $\mathbf{g}_n$ defines the positions (typically, but not necessarily equidistantly located) of the grid as $\mathbf{g}_n = \left(g_{n,1} = \omega_n^{\min} \cdots g_{n,G_n} = \omega_n^{\max}\right)$ by dimensions. Thus, the $O_1 \times \ldots \times O_K$ sized elements $\mathcal{F}_{m_1,m_2,\ldots,m_N}$ of tensor $\mathcal{F}^{D(\Omega,G)}$ (such that $m_n = 1, \ldots, G_n$) are:

$$\mathcal{F}_{m_1,m_2,\ldots,m_N} = f(\mathbf{x}), \tag{2.1}$$

where $\mathbf{x} = \left(g_{1,m_1} \cdots g_{N,m_N}\right)$.

If we have a vector $\mathbf{w}(x)$ containing weighting functions $w_i(x)$, $(i = 1, \ldots, I)$ as

$$\mathbf{w}(x) = \left(w_1(x) \cdots w_I(x)\right) \tag{2.2}$$

P. Baranyi, *TP-Model Transformation-Based-Control Design Frameworks*,
DOI 10.1007/978-3-319-19605-3_2

then $\mathcal{W}^{D(\omega,G)} \in \mathbb{R}^{G \times I}$ is a matrix whose column vectors are the discretized variants of the functions $w_i(x)$ as:

$$\mathcal{W}^{D(\omega,G)} = \left((\mathrm{w}_1^{D(\omega,G)})^T \cdots (\mathrm{w}_I^{D(\omega,G)})^T \right), \tag{2.3}$$

where

$$\mathrm{w}_i^{D(\omega,G)} = \left(w_i(g_1) \cdots w_i(g_G) \right). \tag{2.4}$$

Thus

$$\mathcal{W}^{D(\omega,G)} = \begin{pmatrix} w_1(g_1) & \cdots & w_I(g_1) \\ \vdots & \ddots & \vdots \\ w_1(g_G) & \cdots & w_I(g_G) \end{pmatrix}. \tag{2.5}$$

Lemma 1. *The discretization of a given* $f(\mathbf{x}) = B \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n(x_n)$ *simplifies to the discretization of the weighting functions as:*

$$\mathcal{F}^{D(\Omega,G)} = B \underset{n \in \mathrm{N}}{\boxtimes} \mathcal{W}_n^{D(\omega_n,G_n)} \tag{2.6}$$

Note that the result of HOSVD has the same structure as the discretized TP functions. Thus, the key idea is that executing HOSVD on the discretized function $\mathcal{F}^{D(\Omega,G)}$, we obtain the discretized form of the HOSVD based canonical form of the TP function:

Algorithm 1 (TP Model Transformation). *Let us assume a function given as* $\mathcal{Y} = \mathcal{B} \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{v}_n(x_n)$, $\mathbf{x} \in \Omega \subset \mathbb{R}^N$. *The goal of the algorithm is to numerically reconstruct the HOSVD based canonical TP function:*

$$\mathcal{Y} = \left(\mathcal{S} \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n(x_n) \right) \underset{N+(k \in \mathrm{K})}{\boxtimes} \mathbf{T}_k \tag{2.7}$$

in Ω*:*

- *STEP 0: Numerical initialization: Define discretization grid G fit to* Ω.
- *STEP 1: Discretization: Determine* $\mathcal{F}^{D(\Omega,G)}$.
- *STEP 2: Reconstruct the core tensor of the model: Determine* $\mathcal{S}$ *and* $\mathbf{U}_n$ *by executing compact HOSVD (CHOSVD) on* $\mathcal{F}^{D(\Omega,G)}$ *(in case of rank reduction or complexity trade-off RHOSVD is executed in this step). This results in*

$$\mathcal{F}^{D(\Omega,G)} = \hat{\mathcal{S}} \underset{n \in \{1,\ldots,N+K\}}{\boxtimes} \mathbf{U}_n \tag{2.8}$$

Thus $\hat{\mathbf{T}}_k = \mathbf{U}_{N+k}$ $(k \in \mathrm{K})$.

- *STEP 3: Determine* $\mathbf{w}_n(x_n)$*: Let* $\hat{\mathfrak{W}}_n^{D(G_n,\omega_n)} = \mathbf{U}_n$*. Weighting functions* $\hat{\mathbf{w}}_n(x_n)$ *in*

$$\mathcal{Y} = \left(\hat{\mathcal{S}} \underset{n\in\mathrm{N}}{\boxtimes} \hat{\mathbf{w}}_n(x_n)\right) \underset{N+k\in\mathrm{K}}{\boxtimes} \hat{\mathbf{T}}_k \tag{2.9}$$

can be reconstructed over any point in ω_n*. For instance, let us calculate the weighting functions* $\hat{\mathbf{w}}_d(x_d)$ *on dimension d over a given point* x_d*. Let us define a new discretization grid* G' *as* $G_1 \times \cdots \times G_{d-1} \times 1 \times G_{d+1} \times \cdots \times G_N$ *and restrict the discretization space to* x_d *as* $\Omega' = \omega_1 \times \cdots \times \omega_{d-1} \times x_d \times \omega_{d+1} \times \cdots \times \omega_N$*, then define* $\mathfrak{F}^{D(G',\Omega')}$*. Then for* x_d*:*

$$\hat{\mathbf{w}}_d(x_d) = \mathfrak{F}_{(d)}^{D(G',\Omega')} \left(\mathcal{Q}_{(d)}\right)^+. \tag{2.10}$$

where

$$\mathcal{Q} = \left(\hat{\mathcal{S}} \underset{n\in\mathrm{N},n\neq d}{\boxtimes} \mathbf{U}_n(x_n)\right) \underset{N+k\in\mathrm{K}}{\boxtimes} \hat{\mathbf{T}}_k. \tag{2.11}$$

lower case "$()_{(d)}$*" denotes the n-mode layout of that dimension.*

- *STEP +1: Transformation error: This step is a numerical checking of the accuracy of the resulting TP function over a huge number of random points in* Ω*.*

Proof 3. Szeidl et al. [8] proves that the TP model transformation numerically reconstructs the HOSVD canonical form in case of scalar output functions, i.e., that if $G_n \to \infty$ then $\hat{\mathcal{S}} \to \mathcal{S}$ and $\mathbf{U}_n = \hat{\mathfrak{W}}_n^{D(\omega_n,G_n)} \to \mathfrak{W}_n^{D(\omega_n,G_n)}$. If we consider matrices $\hat{\mathbf{T}}_k$ resulting from SVD in the same way as matrices U_n are considered for $n \in \mathrm{N}$ in the proof presented in paper [8] (but without transforming them to functions), we arrive at a proof of the claim that the TP model transformation numerically reconstructs the HOSVD based canonical form ($\hat{\mathbf{T}} \to \mathbf{T}$, as well as a quasi-HOSVD based canonical form when we multiply by $\hat{\mathbf{T}}_k$ (see Theorems 1.1 and 1.2). Paper [8] also gives theorems for the speed of the convergence for the numerical reconstruction depending on whether we use equidistant or non-equidistant rectangular grids for discretization.

Remark 2.1. The numerical implementation limits the grid density, as $\forall n \in \mathrm{N} : G_n \to G_n^{\max} < \infty$. Furthermore, the computational load of HOSVD can easily explode as G_n and N grow larger. These factors form the bottlenecks of this algorithm. However, we can still say that the TP model transformation numerically reconstructs $\mathcal{S}$ and the weighting functions. Further, papers [4, 7] propose very effective computational complexity reduction techniques for the TP model transformation, especially for cases when it is executed on qLPV models.

Remark 2.2. The paper of Szeidl et al. [8] also derives theorems for the smallest grid density necessary for finding all the ranks of the TP function or model, thus, the discretization density should be set according to Szeidl's theorems and based on the fact that the maximum rank is determined by the number of the weighting functions

by dimensions of the given TP function or model. If we do not see the structure of the given TP function or model to be transformed, or the given model is not a TP function or model (see later), then we can practically use a grid with the highest density made possible by the numerical implementation. If we have limitations for the maximum number of weighting functions and vertexes to be accepted, then we may apply Szeidl's theorem as in the case where we have information about the maximum rank. As a matter of fact, this may lead to an approximation if the accepted number of weighting functions is less than the rank of the TP function.

Remark 2.3. If the density of the discretization grid is not sufficiently high to find the rank of the given TP function, then the TP model transformation results in an approximation only.

In the case of the above two remarks the transformation works as in the case of non-TP functions (see details later and also [9]).

2.1.1 Numerical Example

The example of this section presents a HOSVD based canonical from (e.g., transformed from the result of an identification) of a very simple qLPV state-space model. This numerical example will be used in this part of the book to study the different features of the generalized TP model transformation and the convex hull manipulation techniques to be discussed in the next sections. All examples in the book are computed using the TP-tool MATLAB toolbox [5].

Let us assume that we have the following qLPV model:

$$\begin{pmatrix} \dot{\mathbf{x}}(t) \\ \mathbf{y}(t) \end{pmatrix} = \mathbf{S}(\mathbf{p}(t)) \begin{pmatrix} \mathbf{x}(t) \\ \mathbf{u}(t) \end{pmatrix}, \tag{2.12}$$

where $\mathbf{p}(t) \in \Omega \subset \mathbb{R}^2$, $\Omega = [-5, 5] \times [-5, 5]$ and

$$\mathbf{S}(\mathbf{p}(t)) = \begin{pmatrix} p_1^2(t) & p_2^2 \sin(2p_2(t)) \\ 2 & p_1(t) + p_2(t) \end{pmatrix}. \tag{2.13}$$

Executing the TP model transformation over $G = 137 \times 137$ results in a very simple quasi-HOSVD based canonical form:

$$\mathbf{S}(\mathbf{p}(t)) = \mathcal{S} \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n(p_n(t)), \tag{2.14}$$

where $\mathcal{S} \in \mathbb{R}^{3\times3\times2\times2}$ and $\mathrm{N} = \{1, 2\}$. The normalized (and the original) singular values of the first dimension are $\sigma_{1,1} = 13.00166(1781.23)$, $\sigma_{1,2} = 4.54219(622.28)$ and $\sigma_{1,3} = 2.90790(398.382)$. The singular values of the second dimension are $\sigma_{2,1} = 11.88067(1627.65)$, $\sigma_{2,2} = 7.05744(966.869)$ and $\sigma_{2,3} = 2.67823(366.918)$.

This two-dimensional case can be easily given without tensor operations as:

$$\mathbf{S}(\mathbf{p}(t)) = \sum_{i=1}^{3}\sum_{j=1}^{3} w_{1,i}(p_1(t))w_{2,j}(p_2(t))\mathbf{S}_{i,j} = \sum_{r=1}^{9} w_r(\mathbf{p}(t))\mathbf{S}_r, \tag{2.15}$$

where vertexes $\mathbf{S}_r \in \mathbb{R}^{2\times 2}$ stored in $\mathcal{S}$ are equivalent to $\mathbf{S}_{i,j}$ and $w_r(\mathbf{p}(t)) = w_{1,i}(p_1(t))w_{2,j}(p_2(t))$, where $r = 1,\ldots,9$ is a one-dimensional linear indexing of 2 dimensional index i,j.

The vertex systems of the HOSVD based canonical form are:

$$\mathbf{S}_{1,1} = 1000 \cdot \begin{bmatrix} 1.5199 & -0.0000 \\ 0.2379 & -0.0000 \end{bmatrix} \tag{2.16}$$

$$\mathbf{S}_{2,1} = \begin{bmatrix} 324.4647 & 0.0000 \\ -135.9956 & -0.0000 \end{bmatrix} \tag{2.17}$$

$$\mathbf{S}_{3,1} = \begin{bmatrix} -0.0000 & 0.0000 \\ -0.0000 & -398.3823 \end{bmatrix} \tag{2.18}$$

$$\mathbf{S}_{1,2} = \begin{bmatrix} 0.0000 & 826.6437 \\ 0.0000 & 145.6048 \end{bmatrix} \tag{2.19}$$

$$\mathbf{S}_{2,2} = \begin{bmatrix} 0.0000 & -472.6149 \\ -0.0000 & -83.2463 \end{bmatrix} \tag{2.20}$$

$$\mathbf{S}_{3,2} = 10^{-12} \cdot \begin{bmatrix} -0.0000 & -0.2981 \\ -0.0001 & -0.2593 \end{bmatrix} \tag{2.21}$$

$$\mathbf{S}_{1,3} = \begin{bmatrix} 0.0000 & 55.2557 \\ -0.0000 & -313.7040 \end{bmatrix} \tag{2.22}$$

$$\mathbf{S}_{2,3} = \begin{bmatrix} 0.0000 & -31.5912 \\ -0.0000 & 179.3532 \end{bmatrix} \tag{2.23}$$

$$\mathbf{S}_{3,3} = 10^{-13} \cdot \begin{bmatrix} -0.0000 & -0.1992 \\ -0.0008 & 0.6839 \end{bmatrix} \tag{2.24}$$

and the assigned weighting functions are shown in Fig. 2.1.

If we would like to generate the "full" HOSVD based canonical form, we can execute the HOSVD on all dimensions of $\mathcal{S}$. This results in:

$$\mathbf{S}(\mathbf{p}(t)) = \mathcal{S}' \underset{n\in\mathrm{N}}{\boxtimes} \mathbf{w}_n(p_n(t)) \times_3 \mathbf{T}_1 \times_4 \mathbf{T}_2 \tag{2.25}$$

$$\mathbf{T}_1 = \begin{bmatrix} -0.987937 & -0.154851 \\ -0.154851 & 0.987937 \end{bmatrix}$$

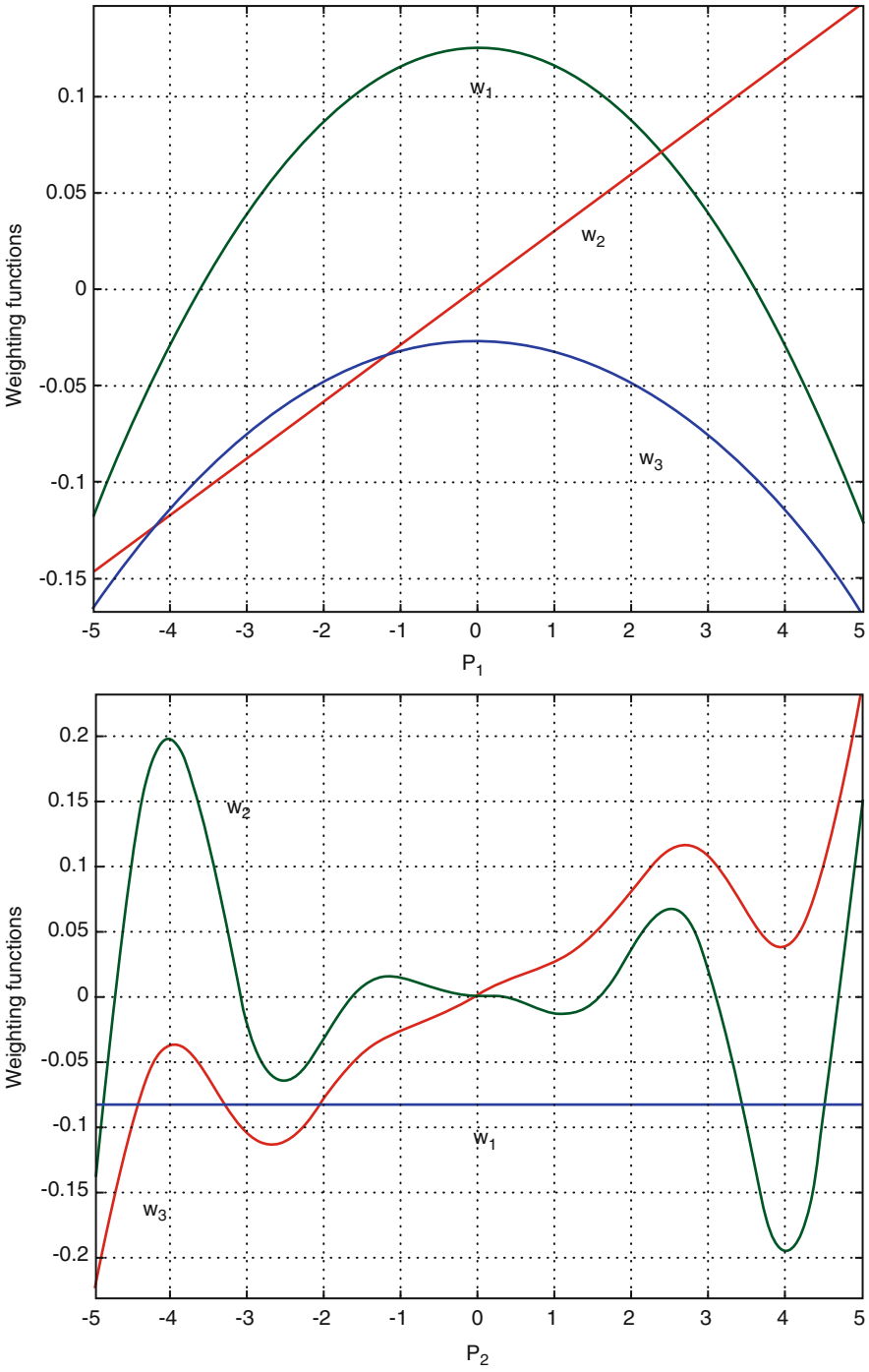

Fig. 2.1 Weighting functions of the exact HOSVD and the quasi-HOSVD based canonical form

$$\mathbf{T}_2 = \begin{bmatrix} -0.999837 & -0.018028 \\ -0.018028 & 0.999837 \end{bmatrix}$$

and the new vertexes are:

$$\mathbf{S}'_{1,1} = \begin{bmatrix} 131.9268 & -0.0000 \\ 17.6940 & 0.0000 \end{bmatrix} \tag{2.26}$$

$$\mathbf{S}'_{2,1} = \begin{bmatrix} 31.8852 & 0.0000 \\ -9.3233 & -0.0000 \end{bmatrix} \tag{2.27}$$

$$\mathbf{S}'_{3,1} = \begin{bmatrix} -0.0000 & -0.0000 \\ 0.0000 & -31.9142 \end{bmatrix} \tag{2.28}$$

$$\mathbf{S}'_{1,2} = \begin{bmatrix} -0.0000 & -82.8854 \\ -0.0000 & 5.1783 \end{bmatrix} \tag{2.29}$$

$$\mathbf{S}'_{2,2} = \begin{bmatrix} -0.0000 & 43.6737 \\ 0.0000 & -2.7285 \end{bmatrix} \tag{2.30}$$

$$\mathbf{S}'_{3,2} = 10^{-12} \cdot \begin{bmatrix} 0.0000 & -0.1790 \\ -0.0000 & 0.1688 \end{bmatrix} \tag{2.31}$$

$$\mathbf{S}'_{1,3} = \begin{bmatrix} 0.0000 & -1.7340 \\ 0.0000 & -27.7556 \end{bmatrix} \tag{2.32}$$

$$\mathbf{S}'_{2,3} = \begin{bmatrix} 0.0000 & 0.9137 \\ -0.0000 & 14.6249 \end{bmatrix} \tag{2.33}$$

$$\mathbf{S}'_{3,3} = 10^{-14} \cdot \begin{bmatrix} -0.0000 & -0.0079 \\ 0.0000 & -0.1421 \end{bmatrix} \tag{2.34}$$

Obviously the weighting functions do not change (Fig. 2.1).

In order to perform complexity trade-off we can execute RHOSVD during the TP model transformation and discard the singular values in increasing order, namely $\sigma_{2,3}$, $\sigma_{1,3}$, $\sigma_{1,2}$, $\sigma_{2,2}$. Figures 2.2, 2.3, 2.4, and 2.5 show the weighting functions of the relaxed quasi and "full" HOSVD based canonical forms.

2.2 Bi-Linear TP Model Transformation

In various engineering cases we have different accuracy requirements for different components of the TP function; hence, it is not always necessary to find all points of the weighting functions in Step 3 of the TP model transformation. For instance, in case of robust control design the precise core tensor $\mathcal{S}$ is important to the extent

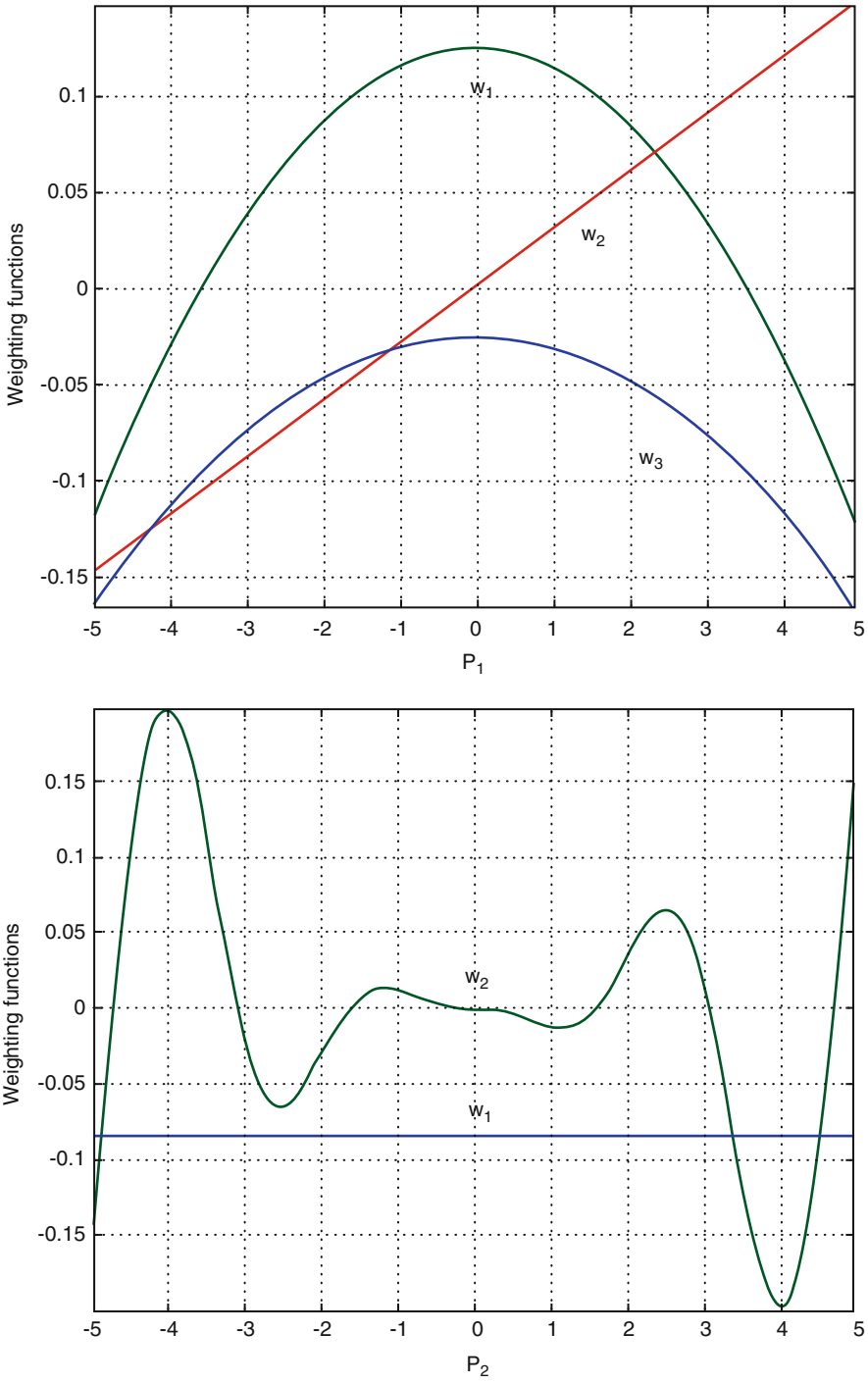

Fig. 2.2 Weighting functions of the RHOSVD based canonical form where 5 singular values are kept

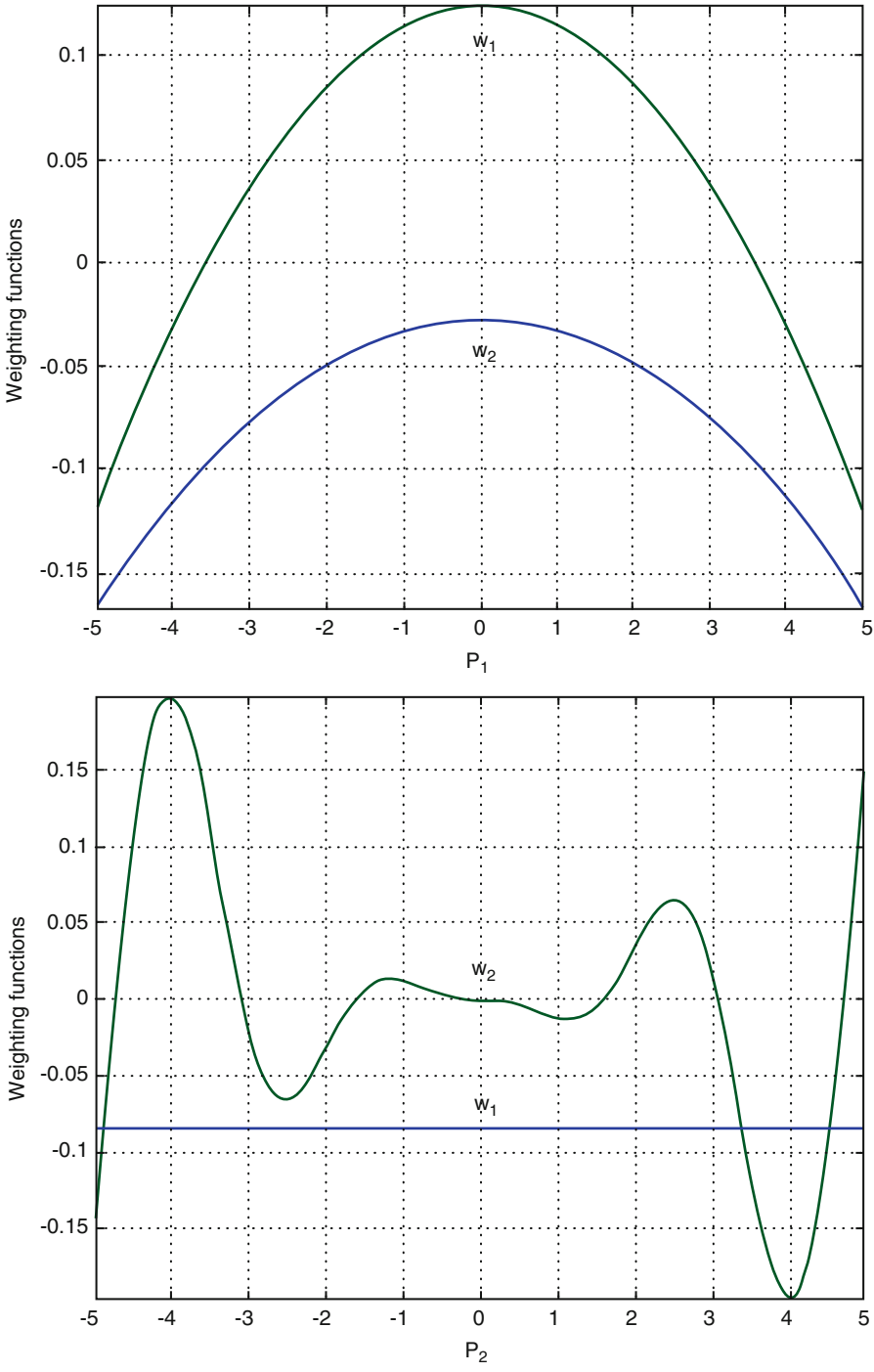

Fig. 2.3 Weighting functions of the RHOSVD based canonical form where 4 singular values are kept

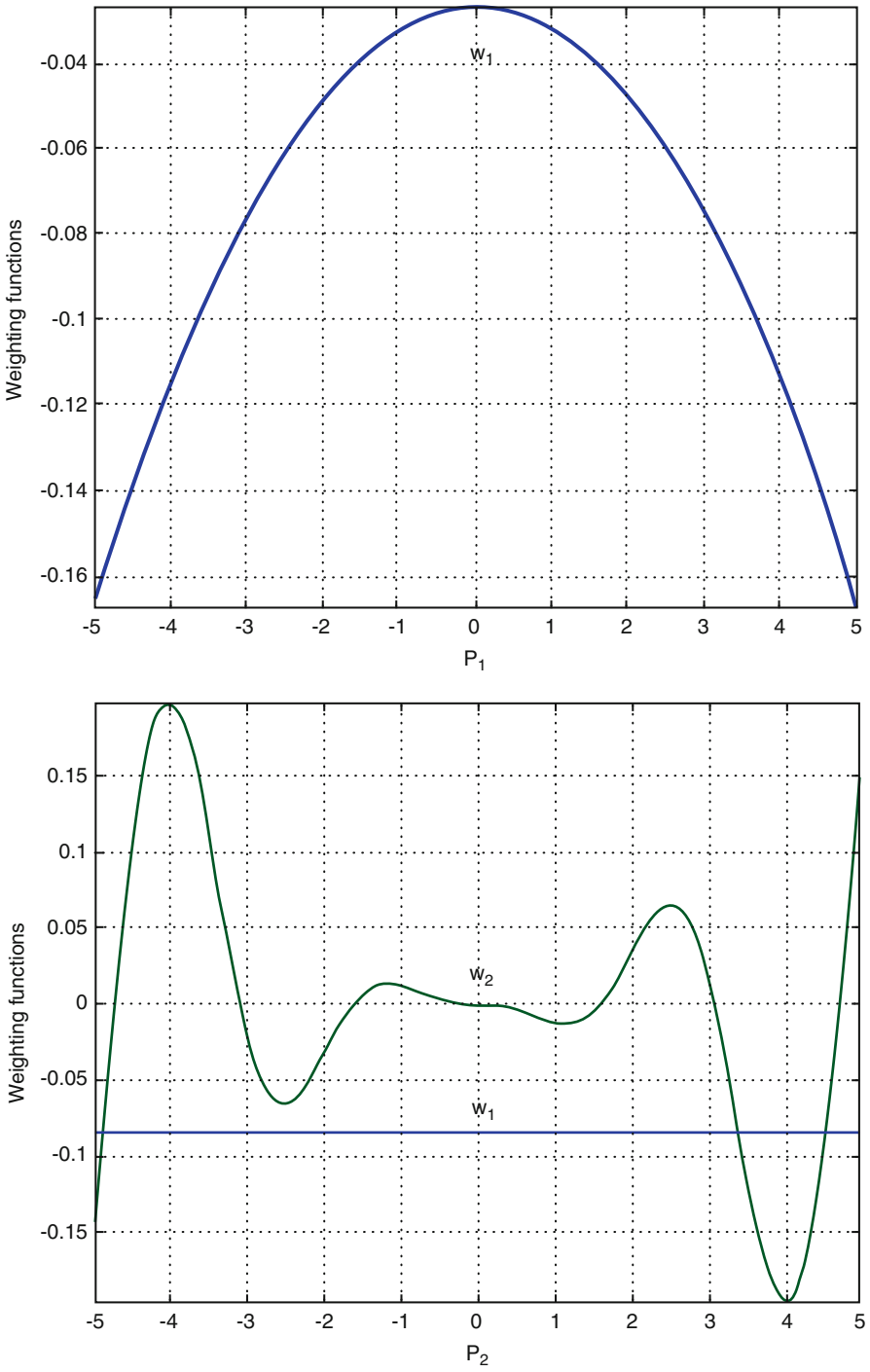

Fig. 2.4 Weighting functions of the RHOSVD based canonical form where 3 singular values are kept

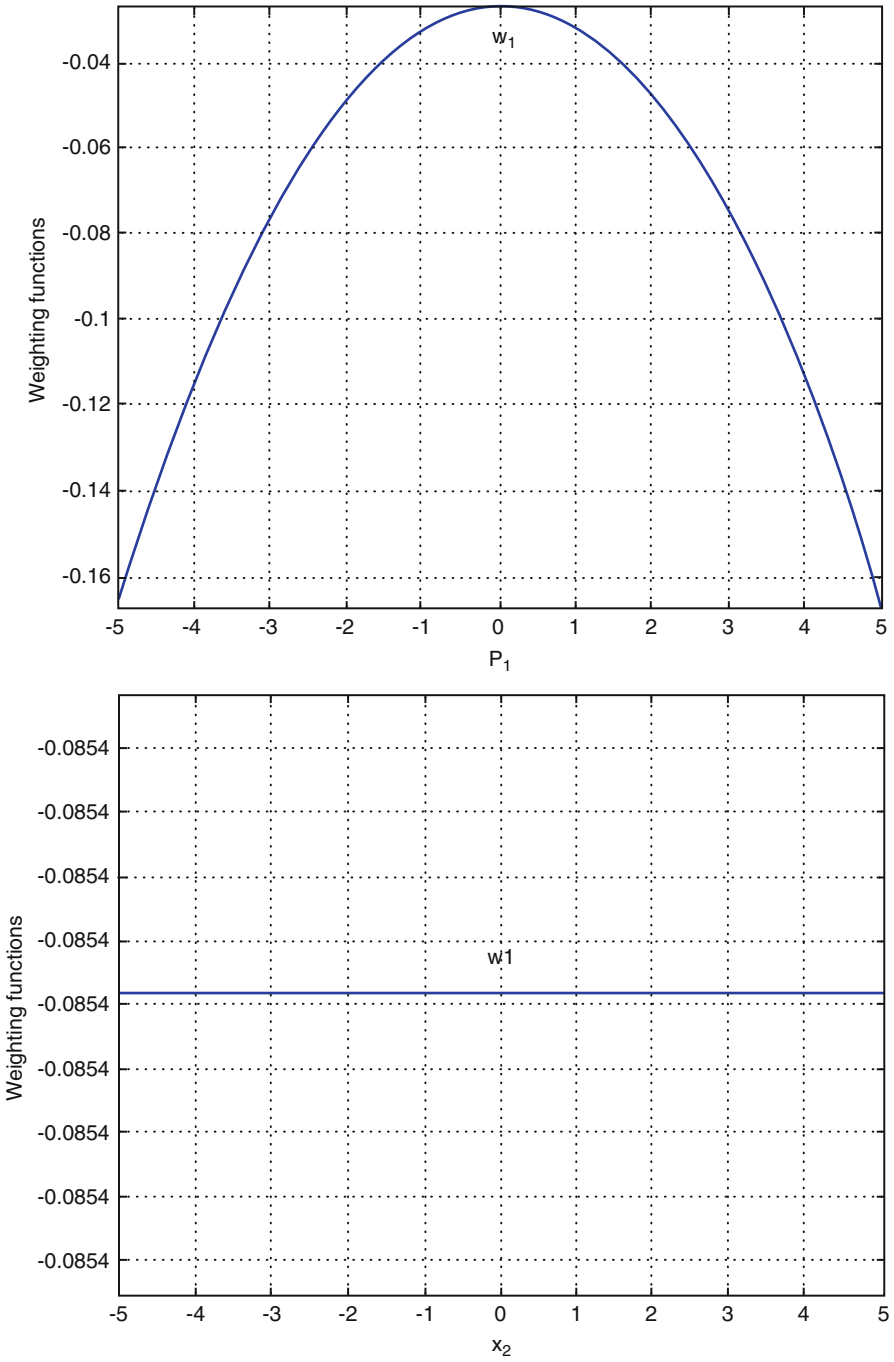

Fig. 2.5 Weighting functions of the RHOSVD based canonical form where 2 singular values are kept

that the control design is based on it; however, in the final implementation of the TP controller, we can accept a good piece-wise approximation of the weighting functions. This idea leads to a practically useful engineering implementation, where we simply use the piece-wise linear variant of the weighting function in the controller.

Definition 2.3 (Piece-Wise Linear Weighting Function System Denoted by $\bar{\mathbf{w}}(x)$). Function $\bar{\mathbf{w}}(x)$, including functions $\bar{w}_i(x)$, is defined by matrix $\mathbf{U}$ and grid G over $x \in \omega$ in such a way that $\mathbf{U} = \bar{\mathfrak{W}}^{D(\omega,G)}$. A linear interpolation between neighboring values in each column of $\mathbf{U}$ fully defines the piece-wise linear functions $\bar{w}_i(x)$.

Algorithm 2 (Bi-Linear TP Model Transformation). *The Bi-linear TP model transformation results in a bi-linear approximation* $f(\mathbf{x}) \approx \mathcal{S} \overset{N}{\underset{n=1}{\boxtimes}} \bar{\mathbf{w}}_n(x_n)$ *of the given function fit to a given grid G. It differs only in Step 3 as:*

STEP 3: $\bar{\mathbf{w}}_n(x_n)$ *is directly defined by* $\mathbf{U}_n$ *(determined in Step 2) and grid G.*

2.2.1 Numerical Example

Let us take the simple dynamic TP model from the previous section and execute the Bi-linear TP model transformation over a sparse grid G ($G_1 \times G_2 = 10 \times 10$) to define the quasi-HOSVD based canonical form. Figure 2.6 shows the piece-wise linear weighting functions of the resulting TP model.

The vertexes are:

$$\mathbf{S}_{1,1} = \begin{bmatrix} 131.9268 & -0.0000 \\ 17.6940 & 0.0000 \end{bmatrix} \tag{2.35}$$

$$\mathbf{S}_{2,1} = \begin{bmatrix} 31.8852 & 0.0000 \\ -9.3233 & -0.0000 \end{bmatrix} \tag{2.36}$$

$$\mathbf{S}_{3,1} = \begin{bmatrix} -0.0000 & 0.0000 \\ -0.0000 & -31.9142 \end{bmatrix} \tag{2.37}$$

$$\mathbf{S}_{1,2} = \begin{bmatrix} -0.0000 & -82.8854 \\ -0.0000 & 5.1783 \end{bmatrix} \tag{2.38}$$

$$\mathbf{S}_{2,2} = \begin{bmatrix} -0.0000 & 43.6737 \\ 0.0000 & -2.7285 \end{bmatrix} \tag{2.39}$$

$$\mathbf{S}_{3,2} = 10^{-13} \cdot \begin{bmatrix} 0.0000 & 0.0329 \\ 0.0000 & 0.1332 \end{bmatrix} \tag{2.40}$$

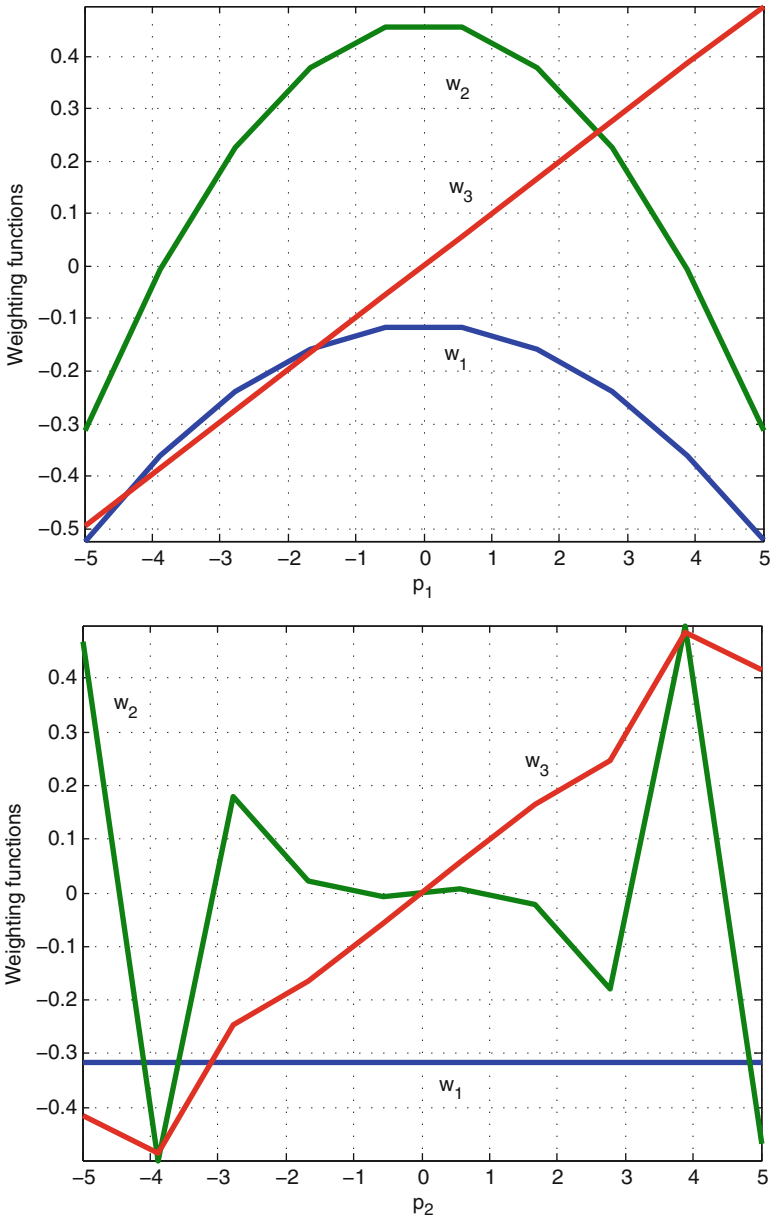

Fig. 2.6 Weighting functions of the bi-linear canonical form

$$\mathbf{S}_{1,3} = \begin{bmatrix} 0.0000 & -1.7340 \\ 0 & -27.7556 \end{bmatrix} \tag{2.41}$$

$$\mathbf{S}_{2,3} = \begin{bmatrix} 0.0000 & 0.9137 \\ 0 & 14.6249 \end{bmatrix} \tag{2.42}$$

$$\mathbf{S}_{3,3} = 10^{-14} \cdot \begin{bmatrix} -0.0000 & 0.0069 \\ -0.0000 & -0.7105 \end{bmatrix} \tag{2.43}$$

2.3 Enriched TP Model Transformation

If the grid density is sufficient to find the precise core tensor, but is too sparse to determine the weighting functions $\bar{\mathbf{w}}_n(x_n)$ with good resolution, then we may combine the third steps of the TP model transformation and the bi-linear TP model transformation. Step 3 of the TP model transformation does not require the execution of HOSVD. Only the memory available limits the off-line storage of a number of points of $\mathbf{w}_n(x_n)$ in Step 3, which can readily be calculated over any $\mathbf{x}$. Therefore, we may simply determine H_n new gridpoints on dimension n in Step 3 of the TP model transformation, where H_n can be considerably larger than $G_n^{\max}$, and we can determine $\mathfrak{W}_n^{D(\omega_n,H_n)}$ in Step 3 of the TP model transformation, which leads to a better resolution of $\bar{\mathbf{w}}_n(x_n)$.

Remark 2.4. The enriched TP model transformation can be used as a tool for relaxing the computational complexity of the TP model transformation. If we know the ranks of the given function in each dimension (i.e., if we know the TP structure of the given function) we can set the minimal grid density accordingly [8]. Alternatively, we may increase the grid density gradually until we find all the nonzero ranks (i.e., the point after which only the number of the zero singular values are found to increase with G_n); following this point, we do not need to further increase the grid density, but may instead use the enriched TP model transformation to define a high resolution for the weighting functions. If we are not sure whether we have found all ranks, we may still proceed further with the enriched TP model transformation. In either case, the step where we numerically check the solution (Step +1) will indicate us whether the system has a sufficient number of weighting functions per dimension. This helps in cases where N is large, so that we can set a very sparse grid per dimension in order to be able to execute the computationally expensive HOSVD then we can define the higher resolution of the piece-wise weighting functions.

2.3.1 Numerical Example

Let us continue the example of the previous section and increase the grid density to 20-by-20 and to 500-by-500. Using the enriched TP model transformation, we have the weighting functions shown in Figs. 2.7 and 2.8. It is important to remark that the vertexes do not change in obvious ways in the present case (only Step 3 of the TP model transformation is executed for the extra, dense grid); it is mostly the resolution of the weighting functions that is improved without having to execute HOSVD on a huge tensor that would be obtained through discretization over a dense 20-by-20 or 500-by-500 grid. Note that the weighting functions will converge to the weighting functions defined by the points taken over the 10×10 grid. **This is different from the case where the density of the discretization grid is increased in the first step of the TP model transformation, and the weighting functions converge to the singular weighting functions of the HOSVD based canonical form.**

2.4 Convex TP Model Transformation: Convex Hull Manipulation

The goal is to transform the given function to a convex TP function $\mathcal{S} \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n^{Co}(x_n)$. This section focuses on the step where the weighting functions can be manipulated and where the already published convex hull generation methods can be incorporated in the algorithm of the TP model transformation.

Algorithm 3 (Convex TP Model Transformation). *Let us assume a given TP function $f(\mathbf{x})$, $\mathbf{x} \in \Omega \subset \mathbb{R}^N$. The goal is to numerically reconstruct TP function $f(\mathbf{x}) = \mathcal{S} \underset{n \in N}{\boxtimes} \mathbf{w}_n^{Co}(x_n)$ and to include a complexity trade-off if needed. The steps of this transformation are the same as in the TP model transformation. Only Step 2 is extended by the convex hull generation. We also add evaluation Step +2 to the algorithm to be executed after Step 3 or Step +1.*

- *STEP 2: Reconstruct the core of the TP structure: Determine $\mathcal{S}$ and $\mathbf{U}_n$ via HOSVD, then use the SN and NN transformations introduced by Yam in [11], which transform $\mathbf{U}_n$ to $\mathbf{U}_n^{Co}$, which will be considered as $\mathfrak{W}^{Co,D(\omega_n,G_n)}$ in Step 3; further, define $\mathcal{S}$ for $\mathfrak{F}^{D(\Omega,G)} = \mathcal{S} \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{U}_n^{Co}$. For instance $\mathcal{S} = \mathfrak{F}^{D(\Omega,G)} \underset{n \in \mathrm{N}}{\boxtimes} \left(\mathbf{U}_n^{Co}\right)^{+}$, where "+" means pseudo inverse.*
- *STEP +2: The weighting functions are $\mathbf{w}_n^{Co}(x_n)$ in the case of the bi-linear TP model transformation; however one has to check this between the grid if all points of the weighting functions are recalculated.*

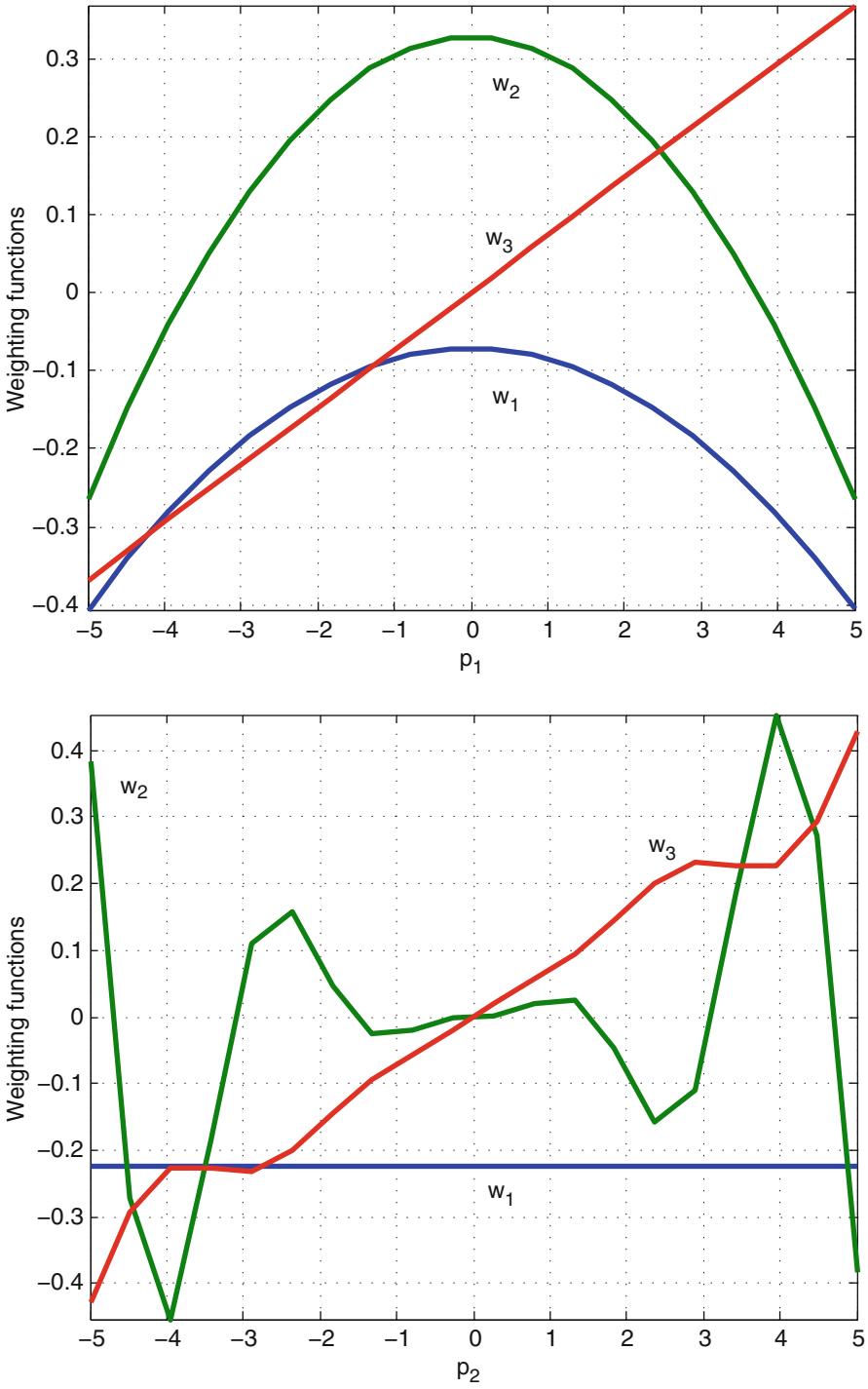

Fig. 2.7 Results of the enriched TP model transformation when G is increased from 10 to 20

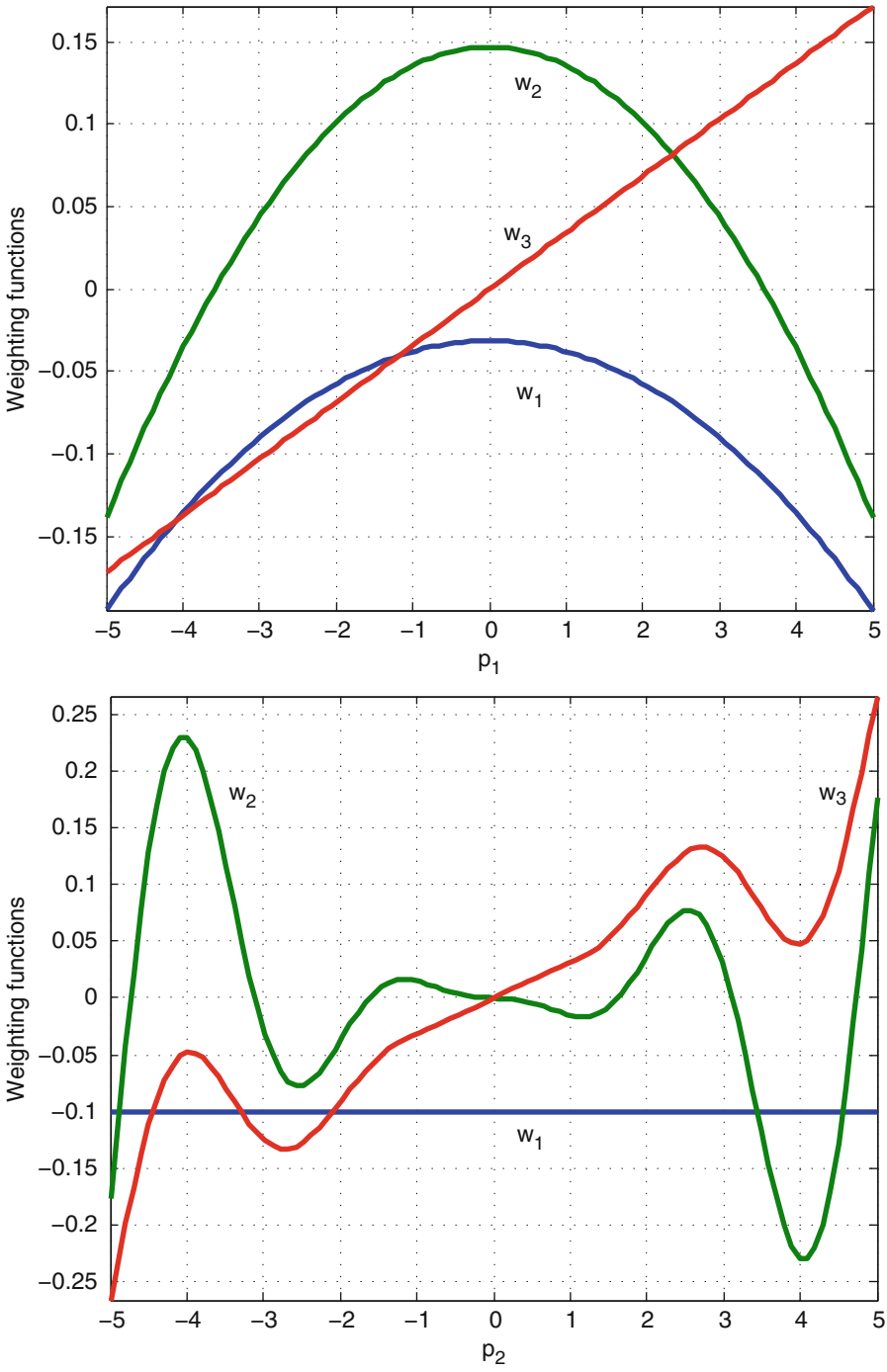

Fig. 2.8 Results of the enriched TP model transformation when G is increased from 20 to 500

Further types of convex TP functions can be generated by using Normalized (NO), Close to NO (CNO), Inverse NO (INO), Relaxed NO (RNO), Inverse RNO (IRNO), etc. transformations in the same way as SN and NN transformation is executed in Step 2 above, see for instance [2, 6, 10, 11].

2.4.1 Numerical Example

Let us continue the example of the previous section. Using the convex TP model transformation the following models are derived:

2.4.1.1 SNNN Type TP Model

The vertexes of the model are:

$$\mathbf{S}_{1,1} = \begin{bmatrix} 122.9804 & -62.4297 \\ 2.0000 & 28.5317 \end{bmatrix} \tag{2.44}$$

$$\mathbf{S}_{2,1} = \begin{bmatrix} -4.5476 & -62.4297 \\ 2.0000 & 77.5678 \end{bmatrix} \tag{2.45}$$

$$\mathbf{S}_{3,1} = \begin{bmatrix} -4.5476 & -62.4297 \\ 2.0000 & 28.5317 \end{bmatrix} \tag{2.46}$$

$$\mathbf{S}_{1,2} = \begin{bmatrix} 122.9804 & 124.7210 \\ 2.0000 & 22.1938 \end{bmatrix} \tag{2.47}$$

$$\mathbf{S}_{2,2} = \begin{bmatrix} -4.5476 & 124.7210 \\ 2.0000 & 71.2299 \end{bmatrix} \tag{2.48}$$

$$\mathbf{S}_{3,2} = \begin{bmatrix} -4.5476 & 124.7210 \\ 2.0000 & 22.1938 \end{bmatrix} \tag{2.49}$$

$$\mathbf{S}_{1,3} = \begin{bmatrix} 122.9804 & -13.0489 \\ 2.0000 & -17.7209 \end{bmatrix} \tag{2.50}$$

$$\mathbf{S}_{2,3} = \begin{bmatrix} -4.5476 & -13.0489 \\ 2.0000 & 31.3152 \end{bmatrix} \tag{2.51}$$

$$\mathbf{S}_{3,3} = \begin{bmatrix} -4.5476 & -13.0489 \\ 2.0000 & -17.7209 \end{bmatrix} \tag{2.52}$$

The weighting functions of the resulting TP model are shown in Fig. 2.9. The weighting functions of the relaxed SNNN type TP models are shown in Figs. 2.10, 2.11, and 2.12.

Remark 2.5. Figure 2.12 shows an interesting case when the number of weighting functions is still 4 (the number of singular values kept is 3). This means that the SNNN transformation increased the number of weighting functions to achieve convexity. Thus, in this case we do not have complexity reduction, but the approximation accuracy is degraded because of the rank reduction (the added weighting functions are not linearly independent). As a result, this solution actually has no meaning in the sense of TP model relaxation. **The previous solution has a better approximation accuracy with the same complexity!**

2.4.1.2 CNO Type TP Model

The vertexes of the model are:

$$\mathbf{S}_{1,1} = \begin{bmatrix} -18.5023 & 8.2775 \\ 2.0000 & -12.9462 \end{bmatrix} \tag{2.53}$$

$$\mathbf{S}_{2,1} = \begin{bmatrix} 25.0000 & 8.2775 \\ 2.0000 & -7.8897 \end{bmatrix} \tag{2.54}$$

$$\mathbf{S}_{3,1} = \begin{bmatrix} 25.0000 & 8.2775 \\ 2.0000 & -18.0028 \end{bmatrix} \tag{2.55}$$

$$\mathbf{S}_{1,2} = \begin{bmatrix} -18.5023 & -21.1903 \\ 2.0000 & -2.1129 \end{bmatrix} \tag{2.56}$$

$$\mathbf{S}_{2,2} = \begin{bmatrix} 25.0000 & -21.1903 \\ 2.0000 & 2.9436 \end{bmatrix} \tag{2.57}$$

$$\mathbf{S}_{3,2} = \begin{bmatrix} 25.0000 & -21.1903 \\ 2.0000 & -7.1695 \end{bmatrix} \tag{2.58}$$

$$\mathbf{S}_{1,3} = \begin{bmatrix} -18.5023 & 77.4772 \\ 2.0000 & 90.3550 \end{bmatrix} \tag{2.59}$$

$$\mathbf{S}_{2,3} = \begin{bmatrix} 25.0000 & 77.4772 \\ 2.0000 & 95.4115 \end{bmatrix} \tag{2.60}$$

$$\mathbf{S}_{3,3} = \begin{bmatrix} 25.0000 & 77.4772 \\ 2.0000 & 85.2984 \end{bmatrix} \tag{2.61}$$

The weighting functions of the resulting TP model are shown in Fig. 2.13.

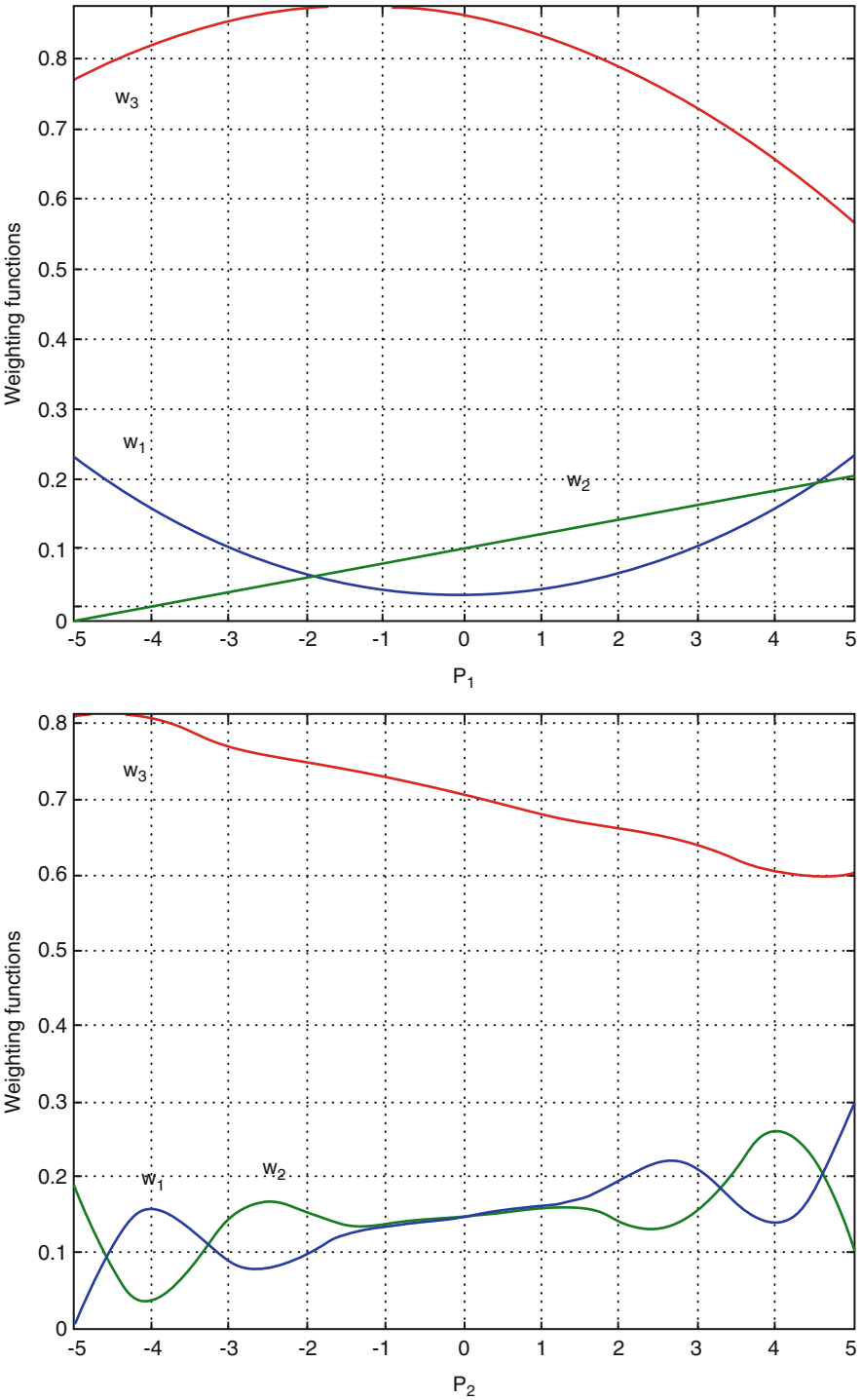

Fig. 2.9 Weighting functions of the SNNN type exact TP model

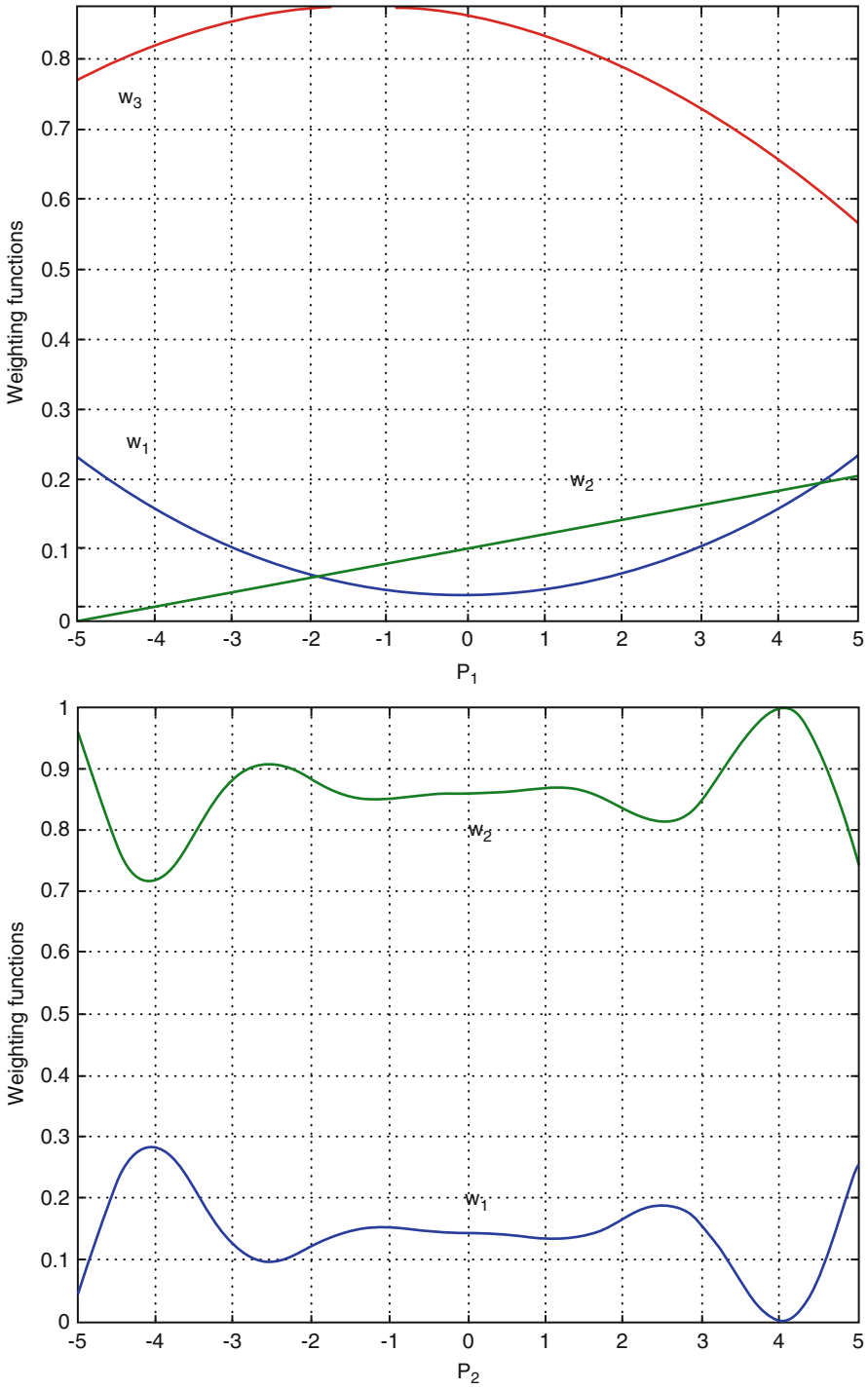

Fig. 2.10 Weighting functions of the SNNN type relaxed TP model where 5 singular values are kept

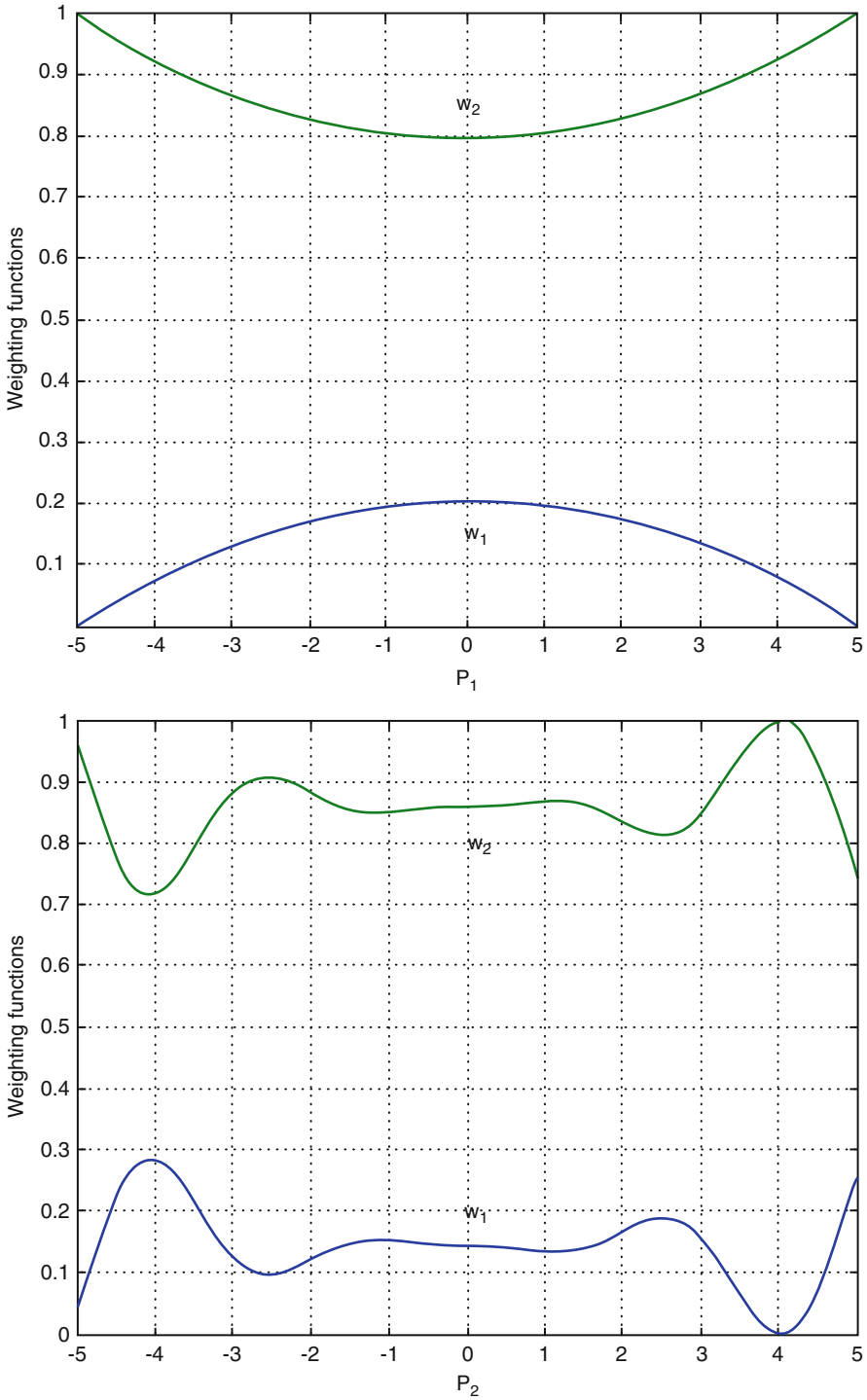

Fig. 2.11 Weighting functions of the SNNN type relaxed TP model where 4 singular values are kept

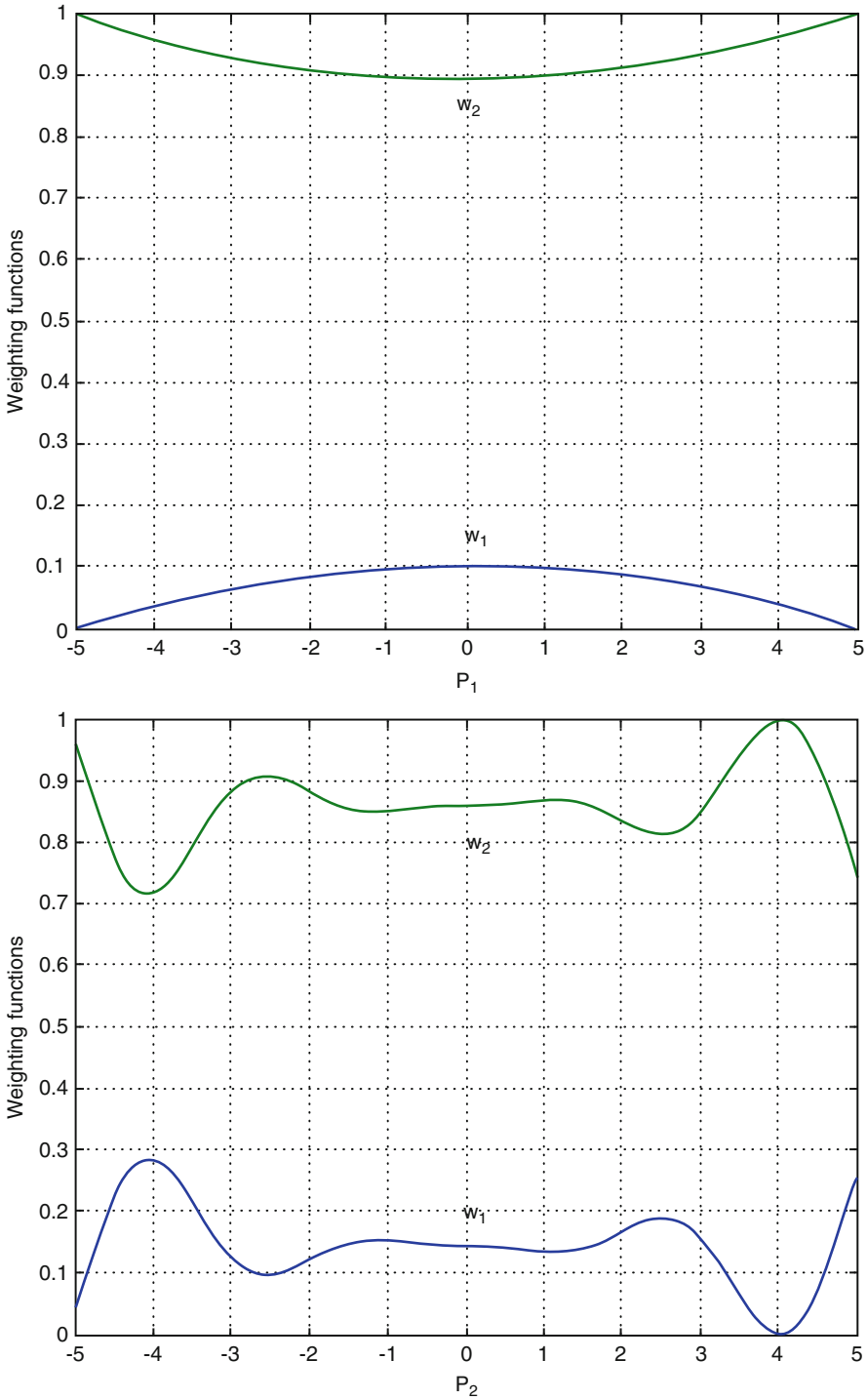

Fig. 2.12 Weighting functions of the SNNN type relaxed TP model where 3 singular values are kept

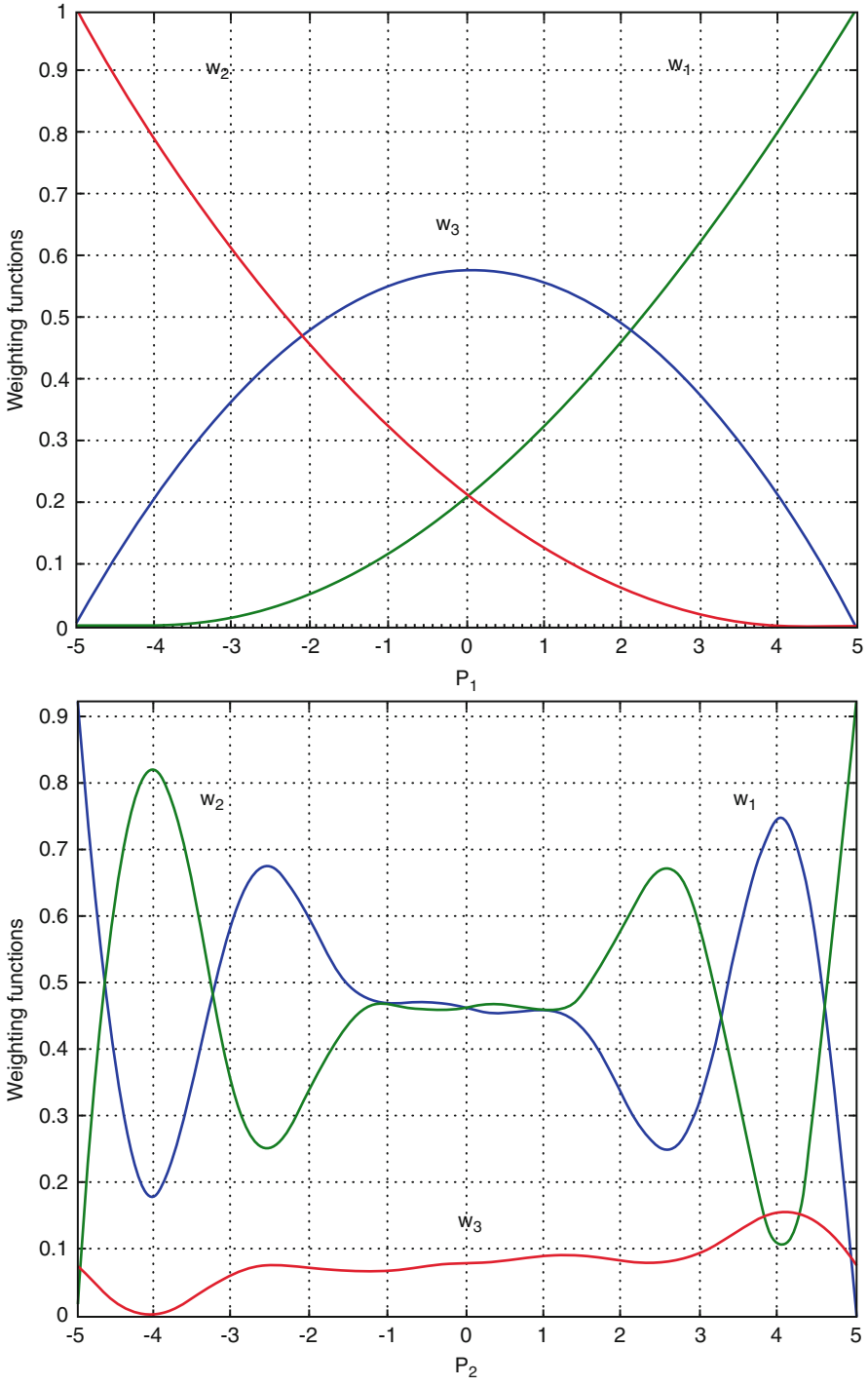

Fig. 2.13 Weighting functions of the CNO type exact TP model

The weighting functions of the relaxed CNO type TP models are shown in Figs. 2.14, 2.15, and 2.16. We have the same situation with the number of weighting functions as in the previous case when 3 singular values are kept.

2.4.1.3 IRNO Type TP Model

The vertexes of the model are:

$$\mathbf{S}_{1,1} = \begin{bmatrix} 48.2074 & -32.7238 \\ 2.0000 & 6.2024 \end{bmatrix} \tag{2.62}$$

$$\mathbf{S}_{2,1} = \begin{bmatrix} 7.8944 & -32.7238 \\ 2.0000 & 18.9717 \end{bmatrix} \tag{2.63}$$

$$\mathbf{S}_{3,1} = \begin{bmatrix} -6.8121 & -32.7238 \\ 2.0000 & -0.0777 \end{bmatrix} \tag{2.64}$$

$$\mathbf{S}_{1,2} = \begin{bmatrix} 48.2074 & 40.0422 \\ 2.0000 & -0.4252 \end{bmatrix} \tag{2.65}$$

$$\mathbf{S}_{2,2} = \begin{bmatrix} 7.8944 & 40.0422 \\ 2.0000 & 12.3441 \end{bmatrix} \tag{2.66}$$

$$\mathbf{S}_{3,2} = \begin{bmatrix} -6.8121 & 40.0422 \\ 2.0000 & -6.7053 \end{bmatrix} \tag{2.67}$$

$$\mathbf{S}_{1,3} = \begin{bmatrix} 48.2074 & -7.3183 \\ 2.0000 & -12.8303 \end{bmatrix} \tag{2.68}$$

$$\mathbf{S}_{2,3} = \begin{bmatrix} 7.8944 & -7.3183 \\ 2.0000 & -0.0610 \end{bmatrix} \tag{2.69}$$

$$\mathbf{S}_{3,3} = \begin{bmatrix} -6.8121 & -7.3183 \\ 2.0000 & -19.1104 \end{bmatrix} \tag{2.70}$$

The weighting functions of the resulting TP model are shown in Fig. 2.17.

The weighting functions of the relaxed IRNO type TP models are shown in Figs. 2.18, 2.19, and 2.20. We have the same situation with the number of weighting functions as in the previous case when 3 singular values are kept.

2.5 Pseudo TP Model Transformation

We may want to find an equivalent TP function with a predefined weighting function system, namely, to transform a given function to a TP function with given weighting functions. For such purposes, we propose the pseudo TP model transformation as follows:

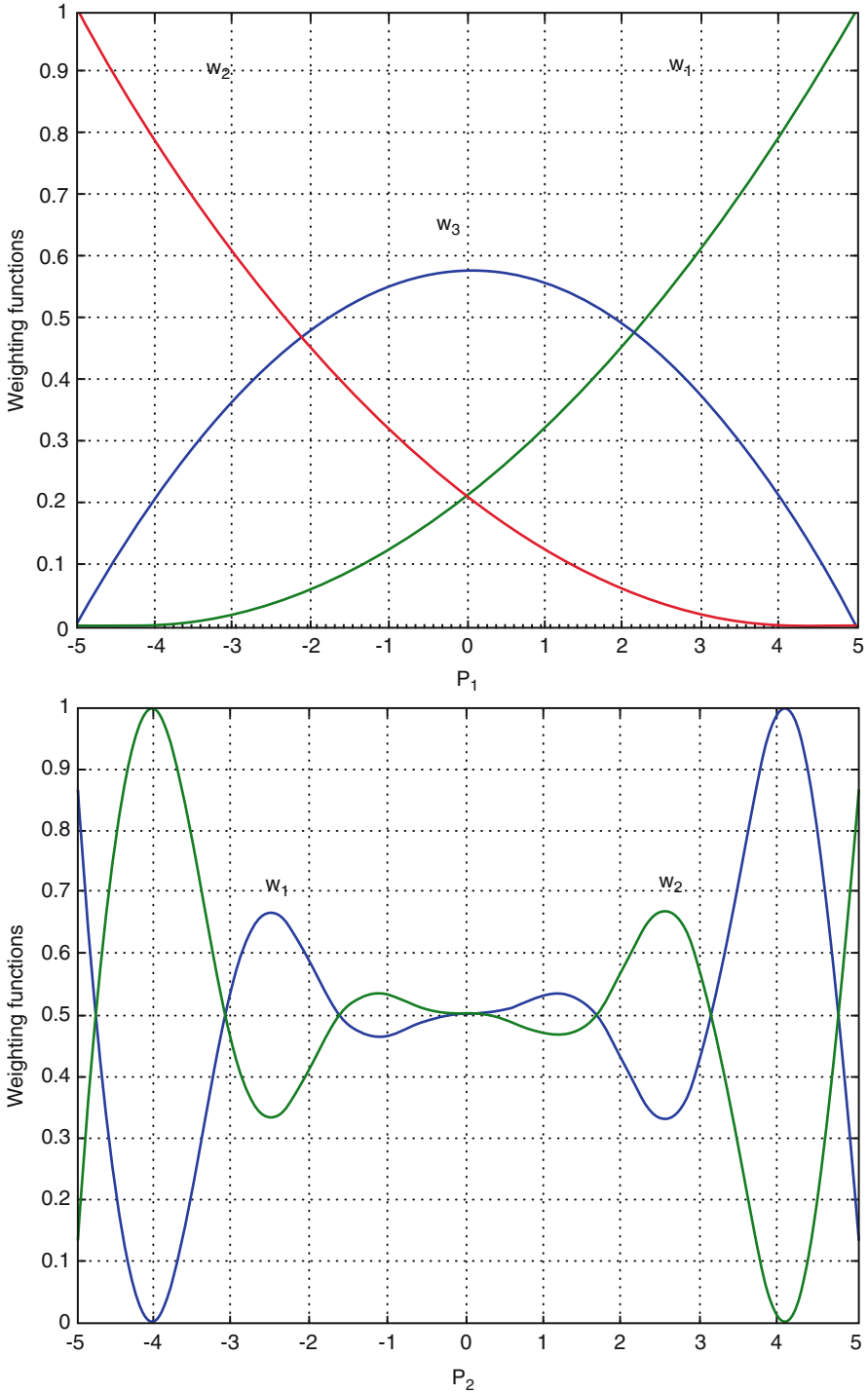

Fig. 2.14 Weighting functions of the CNO type relaxed TP model where 5 singular values are kept

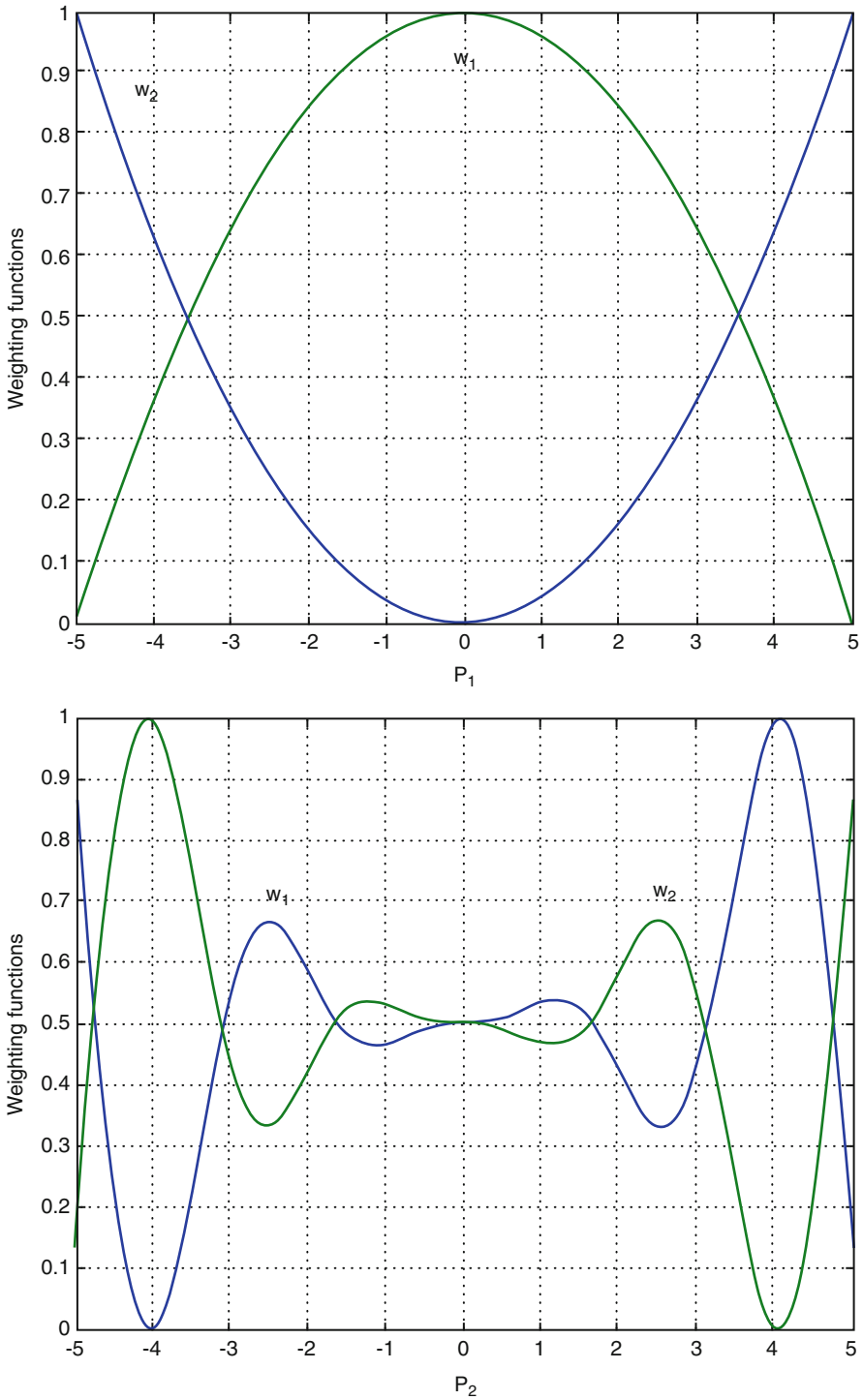

Fig. 2.15 Weighting functions of the CNO type relaxed TP model where 4 singular values are kept

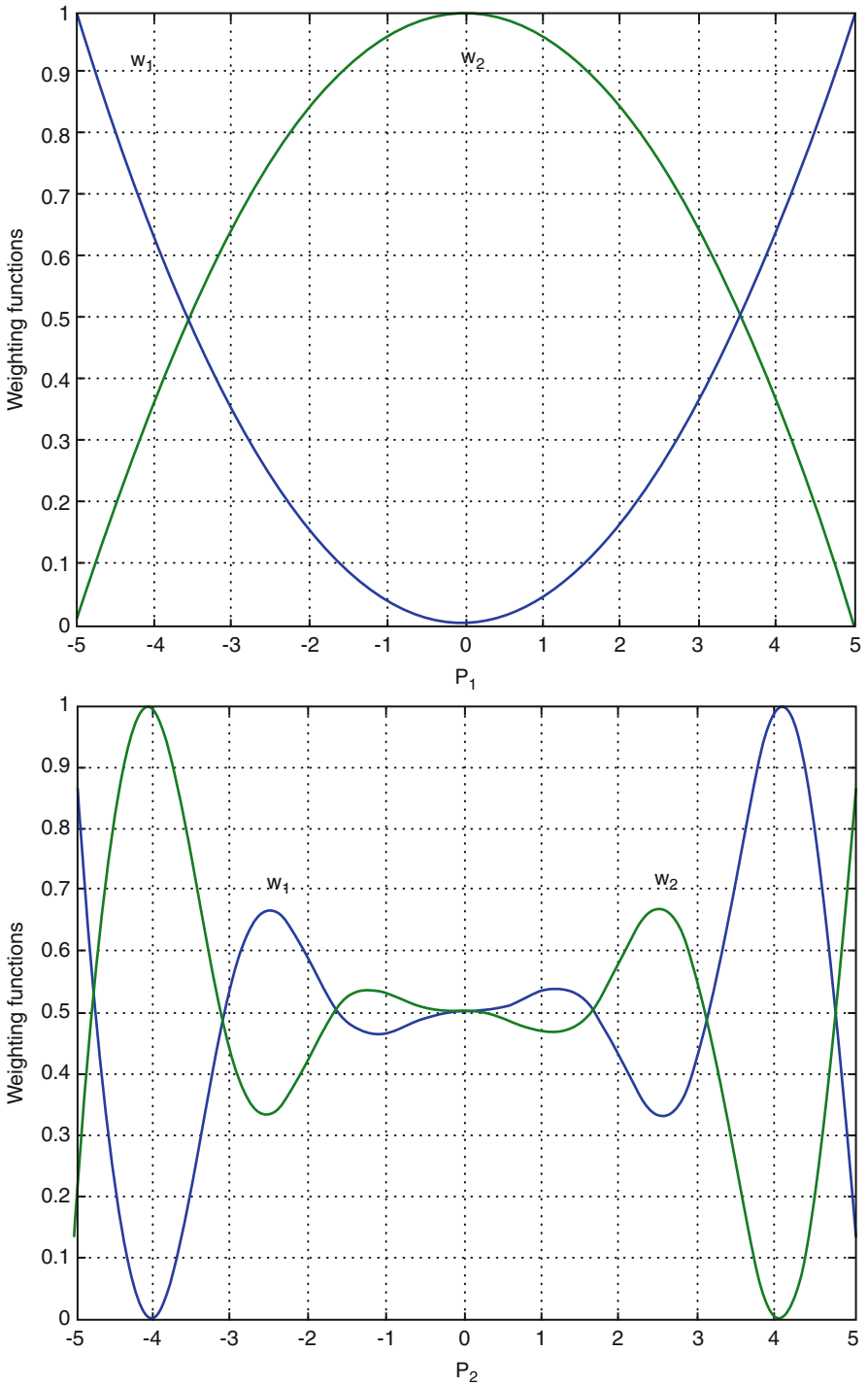

Fig. 2.16 Weighting functions of the CNO type relaxed TP model where 3 singular values are kept

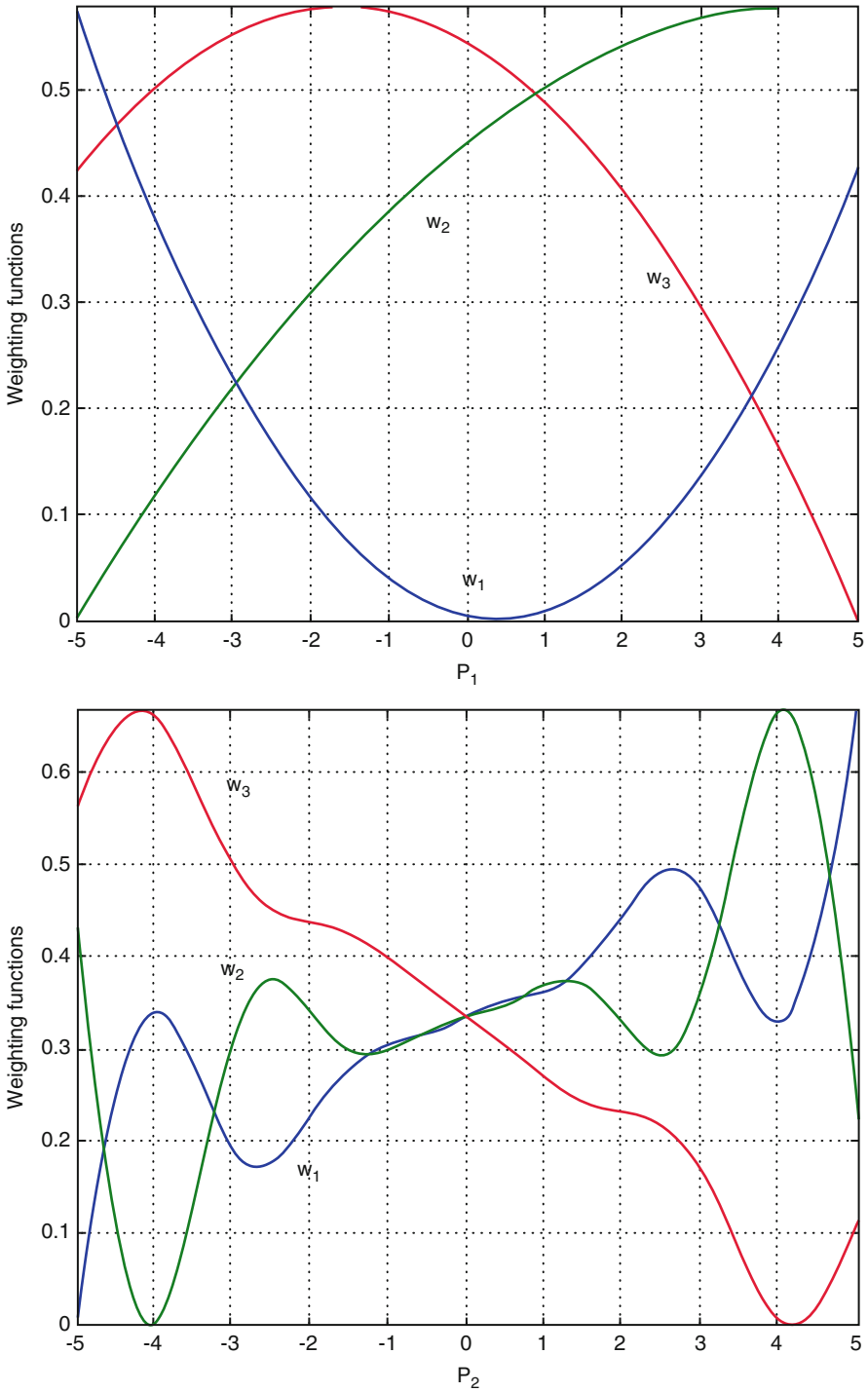

Fig. 2.17 Weighting functions of the IRNO type exact TP model

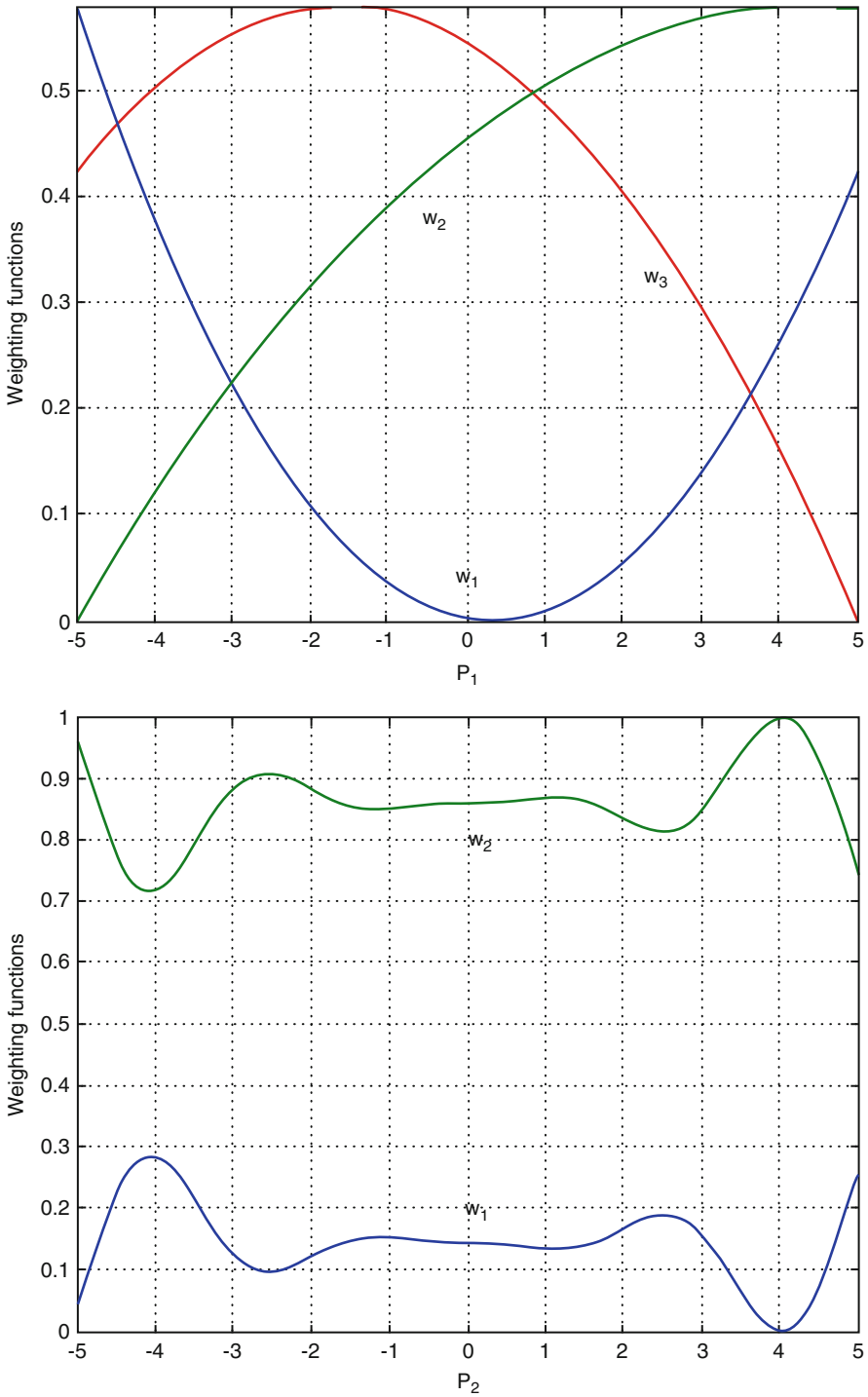

Fig. 2.18 Weighting functions of the IRNO type relaxed TP model where 5 singular values are kept

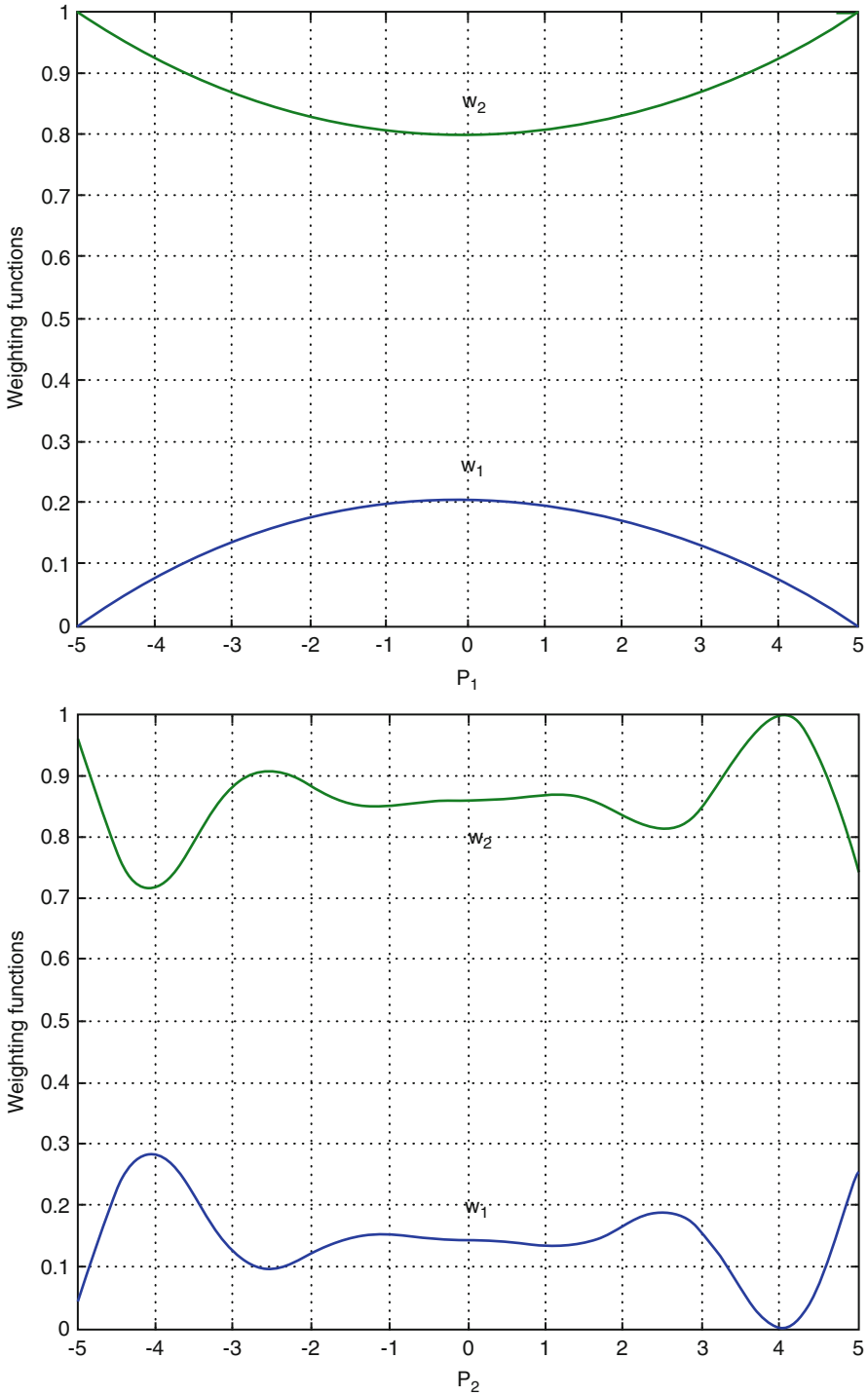

Fig. 2.19 Weighting functions of the IRNO type relaxed TP model where 4 singular values are kept

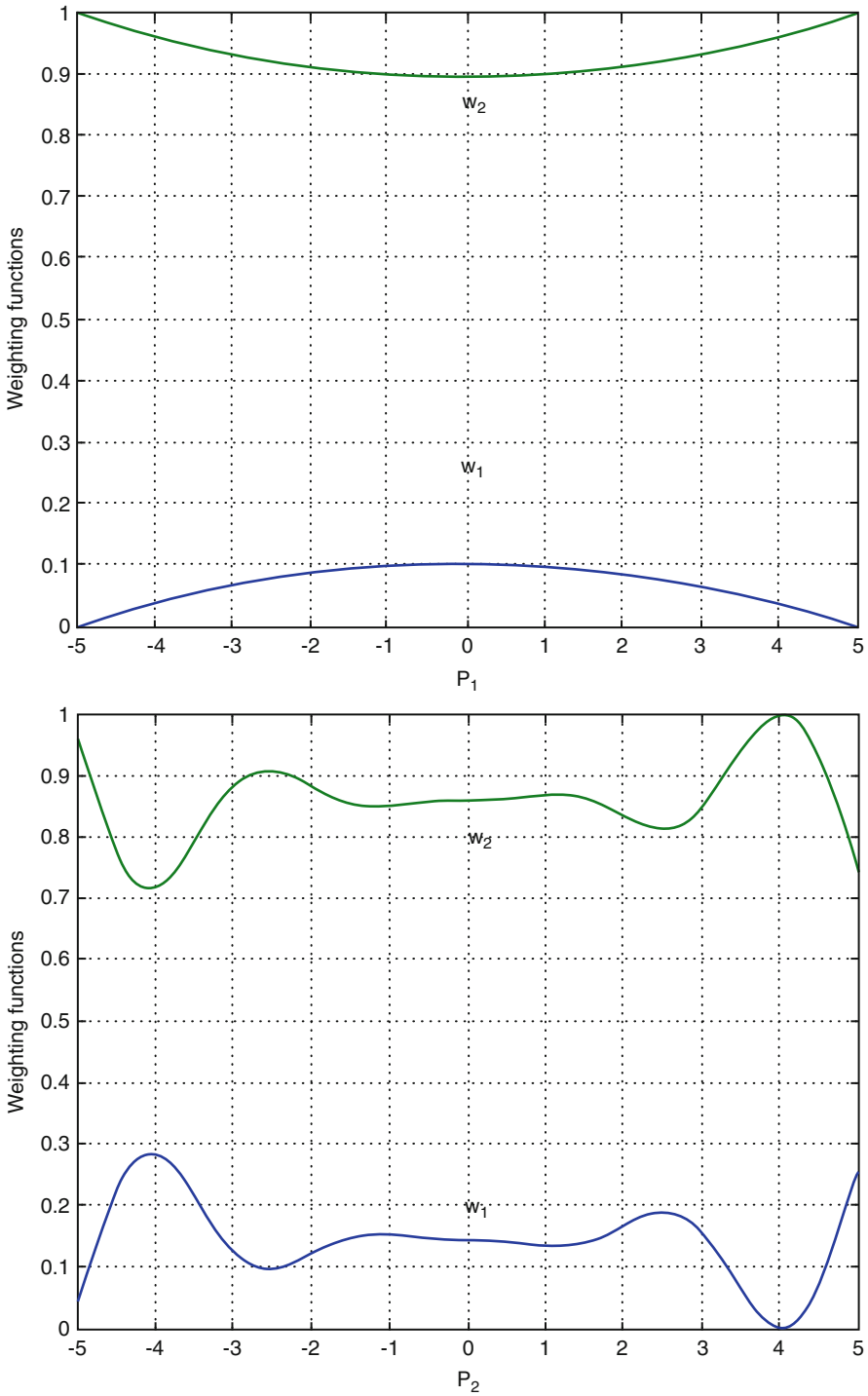

Fig. 2.20 Weighting functions of the IRNO type relaxed TP model where 3 singular values are kept

Algorithm 4 (Pseudo TP Model Transformation, TP$^+$ Model Transformation for Short). *Assume a given function $\mathcal{Y} = f(\mathbf{x})$, $\mathbf{x} \in \Omega \subset \mathbb{R}^N$ and weighting function system $\mathbf{w}_n(x_n)$. The goal is to determine $\mathcal{S}$ such that $f(\mathbf{x}) = \mathcal{S} \underset{n \in N}{\boxtimes} \mathbf{w}_n(x_n)$, or if this is not possible, then the goal is to find $\hat{f}(\mathbf{x}) = \mathcal{S} \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n(x_n)$, where $\hat{f}(\mathbf{x})$ is the best or at least a good approximation under the rank constraints implicitly given by $\mathbf{w}_n(x_n)$ (e.g., the number of linearly independent weighting functions may be less in dimension n than $\text{rank}_n(f(\mathbf{x}))$). Steps 0 and +1 are the same as in the TP model transformation, and only the following steps are extended:*

- *STEP 1: Discretization: Determine $\mathfrak{F}^{D(\Omega,G)}$ and $\mathfrak{W}_n^{D(\omega_n,G_n)}$.*
- *STEP 2: Determine the core tensor:*

$$\mathcal{S} = \mathfrak{F}^{D(\Omega,G)} \underset{n \in \mathrm{N}}{\boxtimes} \left(\mathfrak{W}_n^{D(\omega_n,G_n)}\right)^+. \tag{2.71}$$

If $\mathfrak{W}_n^{D(\omega_n,G_n)}$ introduces rank reduction, then we arrive at $\hat{f}(\mathbf{x}) = \mathcal{S} \underset{n \in N}{\boxtimes} \mathbf{w}_n(x_n)$. This works like in the case of complexity trade-off via TP model transformation. If we have predefined transformations $\mathbf{T}$:

$$\mathcal{S} = \mathcal{Q} \underset{N+k \in \mathrm{K}}{\boxtimes} \mathbf{T}_k^+, \tag{2.72}$$

where

$$\mathcal{Q} = \mathfrak{F}^{D(G,\Omega)} \underset{n \in \mathrm{N}}{\boxtimes} \left(\mathfrak{W}_n^{D(\omega_n,G_n)}\right)^+. \tag{2.73}$$

- *STEP +2: Checking the weighting functions: Once we have the core tensor $\mathcal{S}$ we may recalculate the weighting functions between the points of $\mathfrak{W}_n^{D(\omega_n,G_n,)}$ through Step 3 and numerically compare to the given $\mathbf{w}_n(x_n)$.*

2.6 Partial TP$^+$ Model Transformation

Algorithm 5 (Partial TP$^+$ Model Transformation). *Assume a given TP function $\mathcal{Y} = f(\mathbf{x})$, $\mathbf{x} \in \Omega \subset \mathbb{R}^N$. Further, assume a given weighting function system $\mathbf{w}_d(x_d)$, $d \in D \subset \mathrm{N}$. The goal is to determine $\mathcal{S}$ such that $f(\mathbf{x}) = \mathcal{S} \underset{n \in N}{\boxtimes} \mathbf{w}_n(x_n)$, where weighting functions $\mathbf{w}_n(x_n)$, $n \notin D$, are the same as in the case of the TP model transformation. If this is not possible, or if we need a complexity trade-off, then the goal is to find $\hat{f}(\mathbf{x}) = \mathcal{S} \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n(x_n)$, where $\hat{f}(\mathbf{x})$ is the best, or at least a good approximation under the rank constraint implicitly given by $\mathbf{w}_d(x_d)$. Steps 0,1,3,+1, and +2 are the same as in the case of the TP$^+$ model transformation:*

STEP 2: Determine the core tensor as $\mathcal{K} = \mathfrak{F}^{D(\Omega,G)} \underset{d\in D}{\boxtimes} \left(\mathfrak{W}_d^{D(\omega_d,G_d)}\right)^+$. *Execute HOSVD on* $\mathcal{K}$ *in all dimensions* $n \notin \mathrm{D}$ *to obtain:*

$$\mathcal{K} = \mathcal{S} \underset{\substack{n\in N\\ n\notin D}}{\boxtimes} \mathbf{U}_n. \tag{2.74}$$

Let $\mathfrak{W}_n^{D(\omega_n,G_n)} = \mathbf{U}_n$, $n \notin D$, *in which case:*

$$\mathfrak{F}^{D(G,\Omega)} = \left(\mathcal{S} \underset{\substack{n\in N\\ n\notin D}}{\boxtimes} \mathbf{U}_n\right) \underset{d\in D}{\boxtimes} \mathfrak{W}_d^{D(\omega_d,G_n)} = \tag{2.75}$$

$$= \mathcal{S} \underset{n\in \mathrm{N}}{\boxtimes} \mathfrak{W}_n^{D(\omega_n,G_n)}. \tag{2.76}$$

2.6.1 Numerical Example

Let us use the model discussed in the previous examples. We assume that the following weighting function systems

$$\mathbf{w}_1(p_1(t)) = \left(0.5\sin(p_1(t)/5)\ 0.000025p_1^2(t)\ 0.005p_1(t)\right), \tag{2.77}$$

and

$$\mathbf{w}_2(p_2(t)) = \left(0.2p_2(t)\ -0.2p_2(t)\ 20\right). \tag{2.78}$$

are given, see Fig. 2.21.

The TP$^+$ model transformation results in the following vertexes

$$\mathbf{S}_{1,1} = \begin{bmatrix} -0.0000 & 0.0295 \\ -0.0000 & 0.0343 \end{bmatrix} \tag{2.79}$$

$$\mathbf{S}_{2,1} = \begin{bmatrix} -0.0000 & -1.1354 \\ 0.0000 & -1.3181 \end{bmatrix} \tag{2.80}$$

$$\mathbf{S}_{3,1} = \begin{bmatrix} -0.0000 & 0.9292 \\ -0.0000 & 1.0787 \end{bmatrix} \tag{2.81}$$

$$\mathbf{S}_{1,2} = \begin{bmatrix} 0.0000 & -0.0295 \\ 0.0000 & -0.0343 \end{bmatrix} \tag{2.82}$$

$$\mathbf{S}_{2,2} = \begin{bmatrix} 0.0000 & 1.1354 \\ -0.0000 & 1.3181 \end{bmatrix} \tag{2.83}$$

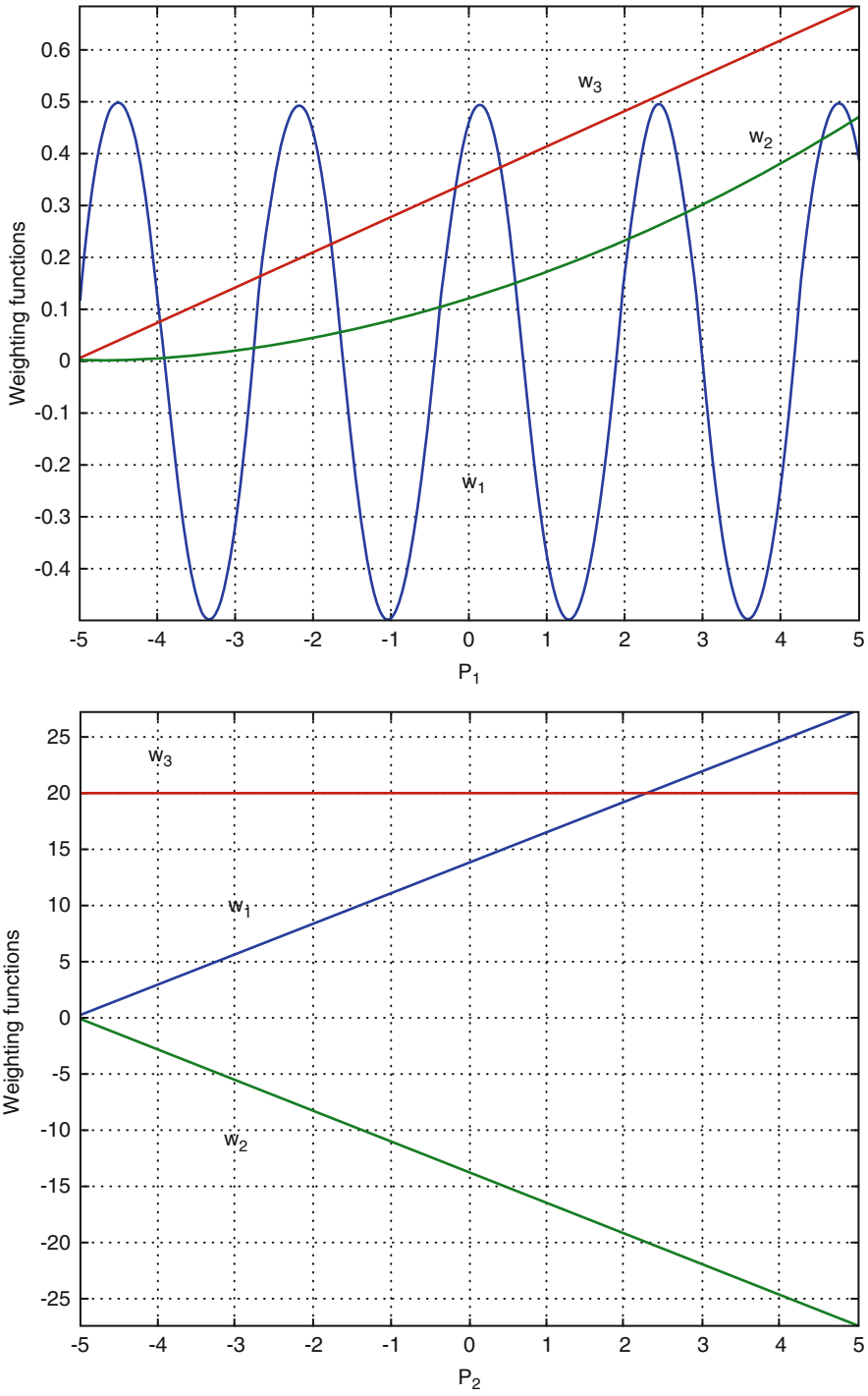

Fig. 2.21 Predefined weighting function systems

$$\mathbf{S}_{3,2} = \begin{bmatrix} 0.0000 & -0.9292 \\ 0.0000 & -1.0787 \end{bmatrix} \tag{2.84}$$

$$\mathbf{S}_{1,3} = \begin{bmatrix} 0.2399 & -0.0407 \\ 0.0186 & -0.0946 \end{bmatrix} \tag{2.85}$$

$$\mathbf{S}_{2,3} = \begin{bmatrix} 1.5845 & 1.5668 \\ -0.7171 & 3.6380 \end{bmatrix} \tag{2.86}$$

$$\mathbf{S}_{3,3} = \begin{bmatrix} 0.0914 & -1.2822 \\ 0.5868 & -2.2419 \end{bmatrix} \tag{2.87}$$

Let us keep the weighting function system on the first dimension and assume that we need a CNO type system of weighting functions in the second dimension. The partial TP$^+$ model transformation results in the following vertexes:

$$\mathbf{S}_{1,1} = \begin{bmatrix} 4.7985 & -0.2571 \\ 0.3728 & 0.9948 \end{bmatrix} \tag{2.88}$$

$$\mathbf{S}_{2,1} = \begin{bmatrix} 31.6892 & 9.8893 \\ -14.3410 & -38.2666 \end{bmatrix} \tag{2.89}$$

$$\mathbf{S}_{3,1} = \begin{bmatrix} 1.8274 & -8.0930 \\ 11.7362 & 46.0220 \end{bmatrix} \tag{2.90}$$

$$\mathbf{S}_{1,2} = \begin{bmatrix} 4.7985 & 5.2362 \\ 0.3728 & -1.0247 \end{bmatrix} \tag{2.91}$$

$$\mathbf{S}_{2,2} = \begin{bmatrix} 31.6892 & -201.4098 \\ -14.3410 & 39.4138 \end{bmatrix} \tag{2.92}$$

$$\mathbf{S}_{3,2} = \begin{bmatrix} 1.8274 & 164.8268 \\ 11.7362 & -17.5490 \end{bmatrix} \tag{2.93}$$

$$\mathbf{S}_{1,3} = \begin{bmatrix} 4.7985 & -29.8745 \\ 0.3728 & -12.1163 \end{bmatrix} \tag{2.94}$$

$$\mathbf{S}_{2,3} = 1000 \cdot \begin{bmatrix} 0.0317 & 1.1491 \\ -0.0143 & 0.4661 \end{bmatrix} \tag{2.95}$$

$$\mathbf{S}_{3,3} = \begin{bmatrix} 1.8274 & -940.4026 \\ 11.7362 & -366.6966 \end{bmatrix}. \tag{2.96}$$

The related weighting functions of the second dimension are shown in Fig. 2.22. If we have the same predefined weighting functions on the first dimension and we need IRNO type weighting function system on the second dimension we obtain the results shown on second image of Fig. 2.22.

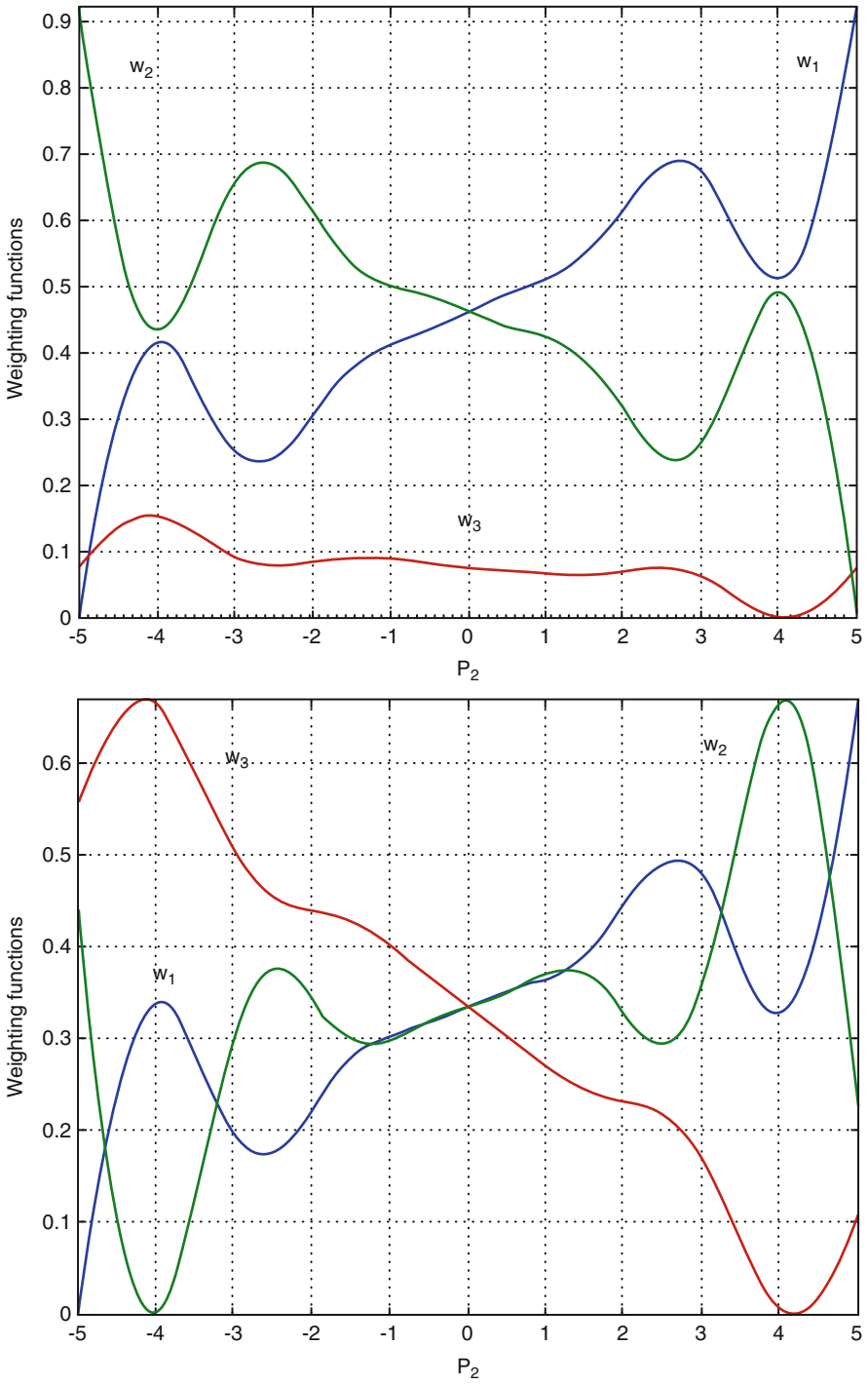

Fig. 2.22 The CNO and the IRNO type weighting functions obtained by the partial TP$^+$ model transformation

2.7 Multi TP Model Transformation

We may want to transform a set of functions simultaneously to a set of TP functions with the same weighting functions system. First we investigate a simple case, when the functions have outputs with same size, then we discuss the general case when each functions may have different sized output.

Algorithm 6 (Multi TP Model Transformation-Simple Case). *Let us assume that we have parameter dependent scalar, vector, matrix, or tensor functions* $\mathcal{Y} = f_l(\mathbf{x})$, $l \in \mathrm{L}$, $\mathbf{x} \in \Omega \subset \mathbb{R}^N$. *An important property is that they have the same size and dimensionality as* $\forall l : f_l(\mathbf{x}) \in \mathbb{R}^{O_1 \times O_2 \times \cdots \times O_K}$. *The goal is to find their TP function representations over the same weighting function system as* $\forall l : f_l(\mathbf{x}) = \mathcal{S}_l \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n(x_n)$.

- *STEP 0: Define discretization grid G fit to* Ω.
- *Step 1: Construct an* $N + K + 1$ *dimensional tensor* $\mathcal{H}$ *storing all the discretized function* $\mathfrak{F}_l^{D(\Omega,G)}$ *into the* $(N + K + 1)$*th dimension.*
- *Step 2: Execute Step 2 as in previous algorithms, but on tensor* $\mathcal{H}$ *that result in (SVD is not executed in the* $N + K + 1$*th dimension)*

$$\mathcal{H} = \mathcal{M} \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n(x_n). \tag{2.97}$$

Then decompose tensor $\mathcal{M}$ *in dimension* $N + K + 1$ *into tensors* $\mathcal{S}_l$ *just oppose to the construction done in Step 1. This results in:*

$$\mathfrak{F}_l^{D(\Omega,G)} = \mathcal{S}_l \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n(x_n). \tag{2.98}$$

The more complex case:

Algorithm 7 (Multi TP Model Transformation). *Let us assume that we have parameter dependent scalar, vector, matrix, or tensor functions* $\mathcal{Y} = f_l(\mathbf{x})$, $l \in \mathrm{L}$, $\mathbf{x} \in \Omega \subset \mathbb{R}^N$. *An important property is that they may have different sizes and dimensionality as* $f_l(\mathbf{x}) \in \mathbb{R}^{O_{l,1} \times O_{l,2} \times \cdots \times O_{l,K_l}}$. *The goal is to find their TP function representations over the same weighting function system as* $f_l(\mathbf{x}) = \mathcal{S}_l \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n(x_n)$*:*

- *STEP 0: Define discretization grid G fit to* Ω.
- *STEP 1: Discretization: store all the output elements of all* $f_l(\mathbf{x})$ *in a vector that is actually a construction of the function* $v(\mathbf{x}) = \big(h_1(\mathbf{x})\ h_2(\mathbf{x})\ \ldots\ h_Z(\mathbf{x})\ \big)$, *where* $Z = \sum_{l=1}^{L} \prod_{k=1}^{K_l} O_{l,k_l}$*; or directly arrange the discretized values according to this ordering that yields the* $N + 1$ *dimensional tensor* $\mathfrak{F}^{D(\Omega,G)}$ *of size* $G_1 \times G_2 \times \cdots \times G_N \times Z$.
- *STEP 2–3: These two steps are the same as in the case of the TP model transformation (including trade-off and convex manipulation etc). As a result we have* $v(\mathbf{x}) = \mathcal{B} \underset{n \in N}{\boxtimes} \mathbf{w}_n(x_n)$, *where* $\mathcal{B}$ *is* $N + 1$ *dimensional.*

- *STEP 4: By repartitioning tensor $\mathcal{B}$ in the $N+1$th dimension, in a fashion opposite to Step 1, we obtain tensors $\mathcal{S}_l$ containing elements with size of $O_{l,1} \times O_{l,2} \times \cdots \times O_{l,K_l}$. Thus we have $f_l(\mathbf{x}) = \mathcal{S}_l \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n(x_n)$.*
- *STEP +1 and +2: These steps have the same error checking role as in the case of the previously discussed variants of the TP model transformation.*

2.7.1 Numerical Example

We assume that we have two different system matrices. The first one $\mathbf{S}_1(\mathbf{p}(t))$ is taken from the previous example, the second one is:

$$\mathbf{S}_2(\mathbf{p}(t)) = \begin{pmatrix} p_1^2 & p_2^2 \cdot \sin(p_2) & p_1 + p_2 & 11 \\ p_2^2 \cdot \cos(2 \cdot p_2) & 7 & 8 & p_1 + 0.625 \cdot p_2 \\ 2 & 5 & 12460 & p_1 \end{pmatrix}. \tag{2.99}$$

If we execute the Multi-TP model transformation with CNO transformation on these two models then we obtain weighting functions as shown in Fig. 2.23. The common rank of the two systems is 3 in the first dimension and 5 in the second dimension. Note that if we execute the TP model transformation on $\mathbf{S}_2(\mathbf{p}(t))$ only, we will obtain three weighting functions on the first and only four weighting functions on the second dimension.

The vertexes of $\mathbf{S}_1(\mathbf{p}(t))$ are:

$$\mathbf{S}_{1,1} = 10^3 \cdot \begin{bmatrix} 0.0025 & 0.0165 \\ 0.0002 & 0.0019 \end{bmatrix} \tag{2.100}$$

$$\mathbf{S}_{2,1} = 10^3 \cdot \begin{bmatrix} 0.0025 & 0.0165 \\ 0.0002 & 0.0030 \end{bmatrix} \tag{2.101}$$

$$\mathbf{S}_{3,1} = 10^3 \cdot \begin{bmatrix} -0.0019 & 0.0165 \\ 0.0002 & 0.0024 \end{bmatrix} \tag{2.102}$$

$$\mathbf{S}_{1,2} = 10^3 \cdot \begin{bmatrix} 0.0025 & -0.0007 \\ 0.0002 & -0.0014 \end{bmatrix} \tag{2.103}$$

$$\mathbf{S}_{2,2} = 10^3 \cdot \begin{bmatrix} 0.0025 & -0.0007 \\ 0.0002 & -0.0004 \end{bmatrix} \tag{2.104}$$

$$\mathbf{S}_{3,2} = 10^3 \cdot \begin{bmatrix} -0.0019 & -0.0007 \\ 0.0002 & -0.0009 \end{bmatrix} \tag{2.105}$$

$$\mathbf{S}_{1,3} = 10^3 \cdot \begin{bmatrix} 0.0025 & -0.0013 \\ 0.0002 & 0.0000 \end{bmatrix} \tag{2.106}$$

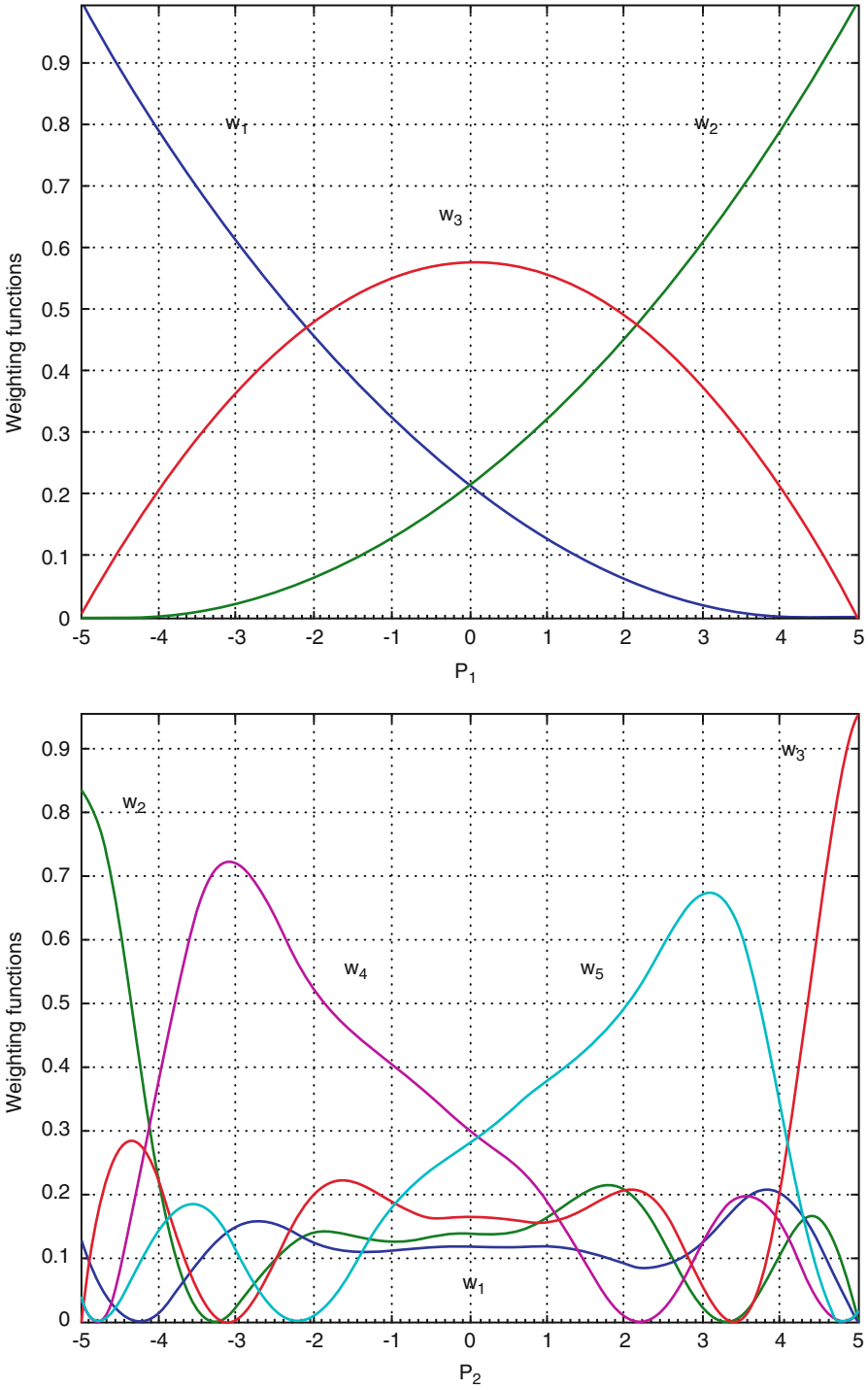

Fig. 2.23 The CNO weighting functions obtained by the multi TP model transformation

$$\mathbf{S}_{2,3} = 10^3 \cdot \begin{bmatrix} 0.0025 & -0.0013 \\ 0.0002 & 0.0010 \end{bmatrix} \tag{2.107}$$

$$\mathbf{S}_{3,3} = 10^3 \cdot \begin{bmatrix} -0.0019 & -0.0013 \\ 0.0002 & 0.0005 \end{bmatrix} \tag{2.108}$$

$$\mathbf{S}_{1,4} = 10^3 \cdot \begin{bmatrix} 0.0025 & -0.0028 \\ 0.0002 & -0.0004 \end{bmatrix} \tag{2.109}$$

$$\mathbf{S}_{2,4} = 10^3 \cdot \begin{bmatrix} 0.0025 & -0.0028 \\ 0.0002 & 0.0006 \end{bmatrix} \tag{2.110}$$

$$\mathbf{S}_{3,4} = 10^3 \cdot \begin{bmatrix} -0.0019 & -0.0028 \\ 0.0002 & 0.0001 \end{bmatrix} \tag{2.111}$$

$$\mathbf{S}_{1,5} = 10^3 \cdot \begin{bmatrix} 0.0025 & -0.0026 \\ 0.0002 & -0.0014 \end{bmatrix} \tag{2.112}$$

$$\mathbf{S}_{2,5} = 10^3 \cdot \begin{bmatrix} 0.0025 & -0.0026 \\ 0.0002 & -0.0004 \end{bmatrix} \tag{2.113}$$

$$\mathbf{S}_{3,5} = 10^3 \cdot \begin{bmatrix} -0.0019 & -0.0026 \\ 0.0002 & -0.0009 \end{bmatrix} \tag{2.114}$$

The vertexes of $\mathbf{S}_2(\mathbf{p}(t))$ are:

$$\mathbf{S}_{1,1} = 10^3 \cdot \begin{bmatrix} 0.0025 & -0.0107 & 0.0019 & 0.0011 \\ -0.0022 & 0.0007 & 0.0008 & 0.0010 \\ 0.0002 & 0.0005 & 1.2460 & -0.0005 \end{bmatrix} \tag{2.115}$$

$$\mathbf{S}_{2,1} = 10^3 \cdot \begin{bmatrix} 0.0025 & -0.0107 & 0.0030 & 0.0011 \\ -0.0022 & 0.0007 & 0.0008 & 0.0020 \\ 0.0002 & 0.0005 & 1.2460 & 0.0005 \end{bmatrix} \tag{2.116}$$

$$\mathbf{S}_{3,1} = 10^3 \cdot \begin{bmatrix} -0.0019 & -0.0107 & 0.0024 & 0.0011 \\ -0.0022 & 0.0007 & 0.0008 & 0.0015 \\ 0.0002 & 0.0005 & 1.2460 & 0.0000 \end{bmatrix} \tag{2.117}$$

$$\mathbf{S}_{1,2} = 10^3 \cdot \begin{bmatrix} 0.0025 & 0.0044 & -0.0014 & 0.0011 \\ -0.0023 & 0.0007 & 0.0008 & -0.0011 \\ 0.0002 & 0.0005 & 1.2460 & -0.0005 \end{bmatrix} \tag{2.118}$$

$$\mathbf{S}_{2,2} = 10^3 \cdot \begin{bmatrix} 0.0025 & 0.0044 & -0.0004 & 0.0011 \\ -0.0023 & 0.0007 & 0.0008 & -0.0001 \\ 0.0002 & 0.0005 & 1.2460 & 0.0005 \end{bmatrix} \tag{2.119}$$

$$\mathbf{S}_{3,2} = 10^3 \cdot \begin{bmatrix} -0.0019 & 0.0044 & -0.0009 & 0.0011 \\ -0.0023 & 0.0007 & 0.0008 & -0.0006 \\ 0.0002 & 0.0005 & 1.2460 & 0.0000 \end{bmatrix} \tag{2.120}$$

$$\mathbf{S}_{1,3} = 10^3 \cdot \begin{bmatrix} 0.0025 & -0.0026 & 0.0000 & 0.0011 \\ -0.0023 & 0.0007 & 0.0008 & -0.0002 \\ 0.0002 & 0.0005 & 1.2460 & -0.0005 \end{bmatrix} \tag{2.121}$$

$$\mathbf{S}_{2,3} = 10^3 \cdot \begin{bmatrix} 0.0025 & -0.0026 & 0.0010 & 0.0011 \\ -0.0023 & 0.0007 & 0.0008 & 0.0008 \\ 0.0002 & 0.0005 & 1.2460 & 0.0005 \end{bmatrix} \tag{2.122}$$

$$\mathbf{S}_{3,3} = 10^3 \cdot \begin{bmatrix} -0.0019 & -0.0026 & 0.0005 & 0.0011 \\ -0.0023 & 0.0007 & 0.0008 & 0.0003 \\ 0.0002 & 0.0005 & 1.2460 & 0.0000 \end{bmatrix} \tag{2.123}$$

$$\mathbf{S}_{1,4} = 10^3 \cdot \begin{bmatrix} 0.0025 & 0.0020 & -0.0004 & 0.0011 \\ 0.0017 & 0.0007 & 0.0008 & -0.0004 \\ 0.0002 & 0.0005 & 1.2460 & -0.0005 \end{bmatrix} \tag{2.124}$$

$$\mathbf{S}_{2,4} = 10^3 \cdot \begin{bmatrix} 0.0025 & 0.0020 & 0.0006 & 0.0011 \\ 0.0017 & 0.0007 & 0.0008 & 0.0006 \\ 0.0002 & 0.0005 & 1.2460 & 0.0005 \end{bmatrix} \tag{2.125}$$

$$\mathbf{S}_{3,4} = 10^3 \cdot \begin{bmatrix} -0.0019 & 0.0020 & 0.0001 & 0.0011 \\ 0.0017 & 0.0007 & 0.0008 & 0.0001 \\ 0.0002 & 0.0005 & 1.2460 & 0.0000 \end{bmatrix} \tag{2.126}$$

$$\mathbf{S}_{1,5} = 10^3 \cdot \begin{bmatrix} 0.0025 & 0.0016 & -0.0014 & 0.0011 \\ 0.0015 & 0.0007 & 0.0008 & -0.0011 \\ 0.0002 & 0.0005 & 1.2460 & -0.0005 \end{bmatrix} \tag{2.127}$$

$$\mathbf{S}_{2,5} = 10^3 \cdot \begin{bmatrix} 0.0025 & 0.0016 & -0.0004 & 0.0011 \\ 0.0015 & 0.0007 & 0.0008 & -0.0001 \\ 0.0002 & 0.0005 & 1.2460 & 0.0005 \end{bmatrix} \tag{2.128}$$

$$\mathbf{S}_{3,5} = 10^3 \cdot \begin{bmatrix} -0.0019 & 0.0016 & -0.0009 & 0.0011 \\ 0.0015 & 0.0007 & 0.0008 & -0.0006 \\ 0.0002 & 0.0005 & 1.2460 & 0.0000 \end{bmatrix} \tag{2.129}$$

2.8 Generalized TP Model Transformation

This section provides a summary of the above sections and describes the summarized algorithm of the TP model transformation.

Let us assume a set of given functions $f_l(\mathbf{x}) \in \mathbb{R}^{O_{l,1}\times\cdots\times O_{l,K_l}}$, $\mathbf{x} \in \Omega \subset \mathbb{R}^N$, $l \in \mathrm{L}$. The goal is to find and further manipulate the TP model representations of the functions in a given Ω. It is irrelevant whether these functions are given using closed formulae or soft-computing techniques (e.g., fuzzy logic, neural network based methods, etc.); the only requirement is that their discretized variant $\mathcal{F}_l^{D(\Omega,G)}$ should be available. If there are no exact TP model representations for any of the functions, then the goal is to find the best TP model representations of those functions by achieving a trade-off between approximation accuracy and the complexity of the core tensor.

It can be further assumed that a set of predefined weighting functions $\mathbf{w}_d(x_d)$ are given for dimensions $d \in \mathrm{D} \subseteq \mathrm{N}$, or that predefined characteristics defined by the specific points of the functions $\mathbf{w}_h(x_h)$ expected for dimensions $h \in \mathrm{H} \subseteq \mathrm{N}$ (obviously $\mathrm{D} \cap \mathrm{H} = 0$) are given in the form $\mathfrak{W}_h^{D(\omega_h,G_h)}$. For such cases, the following transformation is proposed:

Algorithm 8 (Summarized TP Model Transformation). *We assume that* $\mathcal{Y} = f_l(\mathbf{x}) \in \mathbb{R}^{O_{l,1}\times\cdots\times O_{l,K_l}}$, $\mathbf{x} \in \Omega \subset \mathbb{R}^N$, $\mathbf{w}_d(x_d)$, $d \in \mathrm{D} \subseteq \mathrm{N}$, $\mathfrak{W}_h^{D(\omega_h,G_h)}$, $h \in \mathrm{H} \subseteq \mathrm{N}$, $\mathrm{D} \cap \mathrm{H} = 0$ *and* Ω *are given and* $\forall l : \mathcal{F}_l^{D(\Omega,G)}$ *exist. The transformation results in* $f_l(\mathbf{x}) = \mathcal{S}_l \underset{n\in\mathrm{N}}{\boxtimes} \mathbf{w}_n(x_n)$*:*

- ***STEP 1: Discretization over G:***
  - *Determine* $\mathcal{F}_l^{D(\Omega,G)} \in \mathbb{R}^{G_1\times\cdots\times G_N\times O_{l,1}\times\cdots\times O_{l,K_l}}$ *and* $\mathcal{W}_d^{D(\omega_d,G_d)} \in \mathbb{R}^{G_d\times I_d}$.
  - *Rearrange the* $O_{l,1} \times \cdots \times O_{l,K_l}$ *sized elements of tensor* $\mathcal{F}_l^{D(\Omega,G)}$ *into vectors of tensor* $\mathbf{H} \in \mathbb{R}^{G_1\times\cdots\times G_N\times Z}$, $Z = \sum_{l=1}^{L} \Pi_{k=1}^{K_l} O_{l,k_l}$.
- ***STEP 2: Determination of the TP structure***
  - *Incorporate the predefined weighting functions or characteristics as*

$$\mathcal{S}' = \left(\mathcal{H} \underset{d\in D}{\boxtimes} \left(\mathfrak{W}_d^{D(\omega_d,G_d)}\right)^+\right) \underset{h\in H}{\boxtimes} \left(\mathfrak{W}_h^{D(\omega_h,G_h)}\right)^+ \tag{2.130}$$

  - *Execute CHOSVD, specifically by discarding all zero singular values, in dimensions* $n \in N, n \notin D \cup H$ *of* $\mathcal{S}'$*:*

$$\mathcal{S}' = \mathcal{S}'' \underset{n\in N, n\notin D\cup H}{\boxtimes} \mathbf{U}_n. \tag{2.131}$$

  - *Let* $\mathfrak{W}_n^{D(\omega_n,G_n)} = \mathbf{U}_n$ *(*$n \in N, n \notin D \cup H$*), in which the fully discretized structure of the TP model can be expressed as:*

$$\mathcal{H} = \mathcal{S}'' \underset{n\in N}{\boxtimes} \mathfrak{W}_n^{D(\omega_n,G_n)}. \tag{2.132}$$

– *$\mathcal{S}''$ can be partitioned in dimension $N+1$ according Step 1. Storing these elements in tensor $\mathcal{S}_l$ in a fashion opposite to Step 1, tensor $\mathcal{F}_l^{D(\Omega,G)}$ is obtained:*

$$\mathcal{F}_l^{D(\Omega,G)} = \mathcal{S}_l \underset{n\in N}{\boxtimes} \mathfrak{W}_n^{D(\omega_n,G_n)}. \tag{2.133}$$

– *When a complexity trade-off is necessary, RHOSVD is performed by discarding nonzero singular values and the corresponding weighting functions. This leads to an approximation of the discretized tensor, which implies that the resulting TP model will only be an approximation. Obviously, the transformation is also not exact if the rank of any $\mathfrak{W}_n^{D(\omega_n,G_n)}$, $n \in \mathrm{D} \cup \mathrm{H}$ is less than the d-mode rank of $\mathcal{H}$.*

- ***STEP 3: Determination of the weighting functions***
  This step is the same as in the original TP model transformation. It determines all points of the weighting functions of $\mathbf{w}_n(x_n)$ or the piece-wise linear variant $\bar{\mathbf{w}}_n(x_n)$ from the discretized variants $\mathfrak{W}_n^{D(\omega_n,G_n)}$.
- ***STEP +1 and +2: Error check of the resulting TP function and the weighting functions*** *The numerical computation of the TP model transformation, the reduction of the number of singular values, the recalculation of the continuous weighting functions, and the use of piece-wise weighting functions all introduce errors into the resulting TP model. These errors can be theoretically bounded based on the discarded singular values, but can also be numerically approximated by measurement over a large number of random points in Ω. The predefined weighting functions in dimensions $d \in D$ can be recalculated in the third step and checked for accuracy. Such a step can serve as a kind of evaluation of the transformation.*

2.9 Interpolation of the Weighting Functions

Since the type of convexity of the TP model influences the LMI based design (see later), it naturally follows that the control performance can be optimized through various manipulations of the TP model. Typically, controller design benefits from the use of a tight convex hull (cf. [6]) that is able to decrease the conservativeness of the solution. However, it is not very well known in the control literature that the effectiveness of the observer and the resulting control performance can also be improved in cases by loosening the convex hull. Therefore, an optimal convex hull exists between these two opposing directions. In the following, a simple TP model manipulation technique is proposed for interpolation between two convex TP models, which captures the transition of the convex hull, for instance, between tight and loose forms. The key idea behind the interpolation is based on the interpolation

of the weighting functions. Assuming that two different TP model representations of a given model are available:

$$\mathbf{S}(\mathbf{p}) = \mathcal{A} \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n^A(p_n) = \mathcal{B} \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n^B(p_n). \tag{2.134}$$

A linear interpolation characterized by a parameter $\lambda \in [0, 1]$ can be applied as follows:

$$\mathbf{w}_n^\lambda(p_n) = \lambda \mathbf{w}_n^A(p_n) + (1 - \lambda)\mathbf{w}_n^B(p_n). \tag{2.135}$$

Executing the TP$^+$ model transformation on the given model with the predefined weighting functions $\mathbf{w}_n^\lambda(p_n)$, the following form is obtained:

$$\mathbf{S}(\mathbf{p}) = \mathcal{V} \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n^\lambda(p_n) = \mathcal{A} \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n^A(p_n) = \mathcal{B} \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n^B(p_n). \tag{2.136}$$

Note that this technique interpolates the weighting functions and, hence, the vertexes of the interpolated TP model are not the linear interpolation of the vertexes of the given two TP models.

2.9.1 Numerical Example

Let us examine the following simple example. We assume the following TP model $\mathbf{S}(p(t)) = \sum_{r=1}^{3} w_r(p(t))\mathbf{S}_r$ given in HOSVD based canonical form, where $p(t) \in \Omega = [0, 0.04]$ and

$$\mathbf{S}_1 = \begin{bmatrix} -11480.824 & -11929.594 & -8.6483351 \\ 11489.012 & 11921.288 & 8.6422240 \end{bmatrix}$$

$$\mathbf{S}_2 = \begin{bmatrix} -132.28962 & 127.48303 & 2.1133932 \\ 130.26178 & -125.36528 & -2.1102187 \end{bmatrix}$$

$$\mathbf{S}_3 = \begin{bmatrix} 0.2018141 & -0.1981534 & -0.4225654 \\ 0.2117763 & -0.2086436 & 0.4221598 \end{bmatrix}.$$

The weighting functions are depicted in Fig. 2.24.

Let us execute the convex TP model transformation on the given TP model with SN and NN transformation. This leads to a TP model where the vertexes define a convex hull around the given qLPV model. The weighting functions are given in Fig. 2.25. The vertexes are:

$$\mathbf{S}_1^{\mathrm{snnn}} = \begin{bmatrix} 935.16726 & 1064.7034 & 2.0443571 \\ -937.13030 & -1062.7079 & -2.0421822 \end{bmatrix}$$

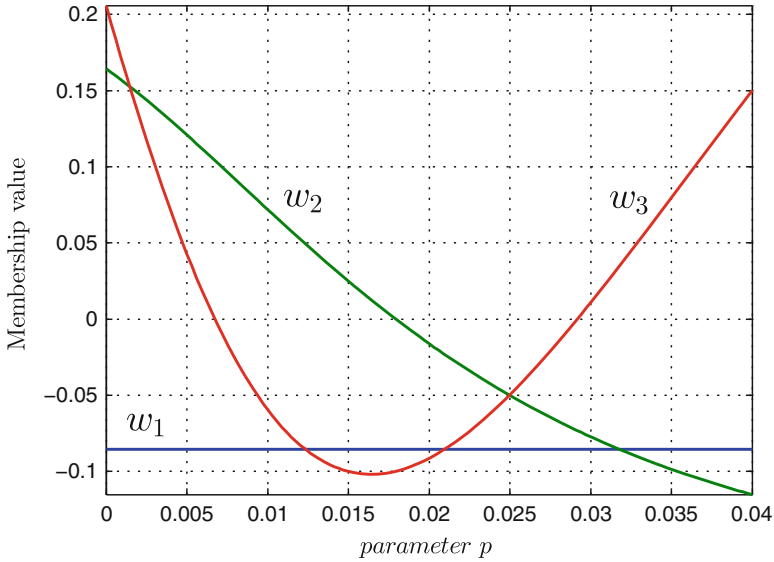

Fig. 2.24 Weighting function system of the HOSVD canonical form

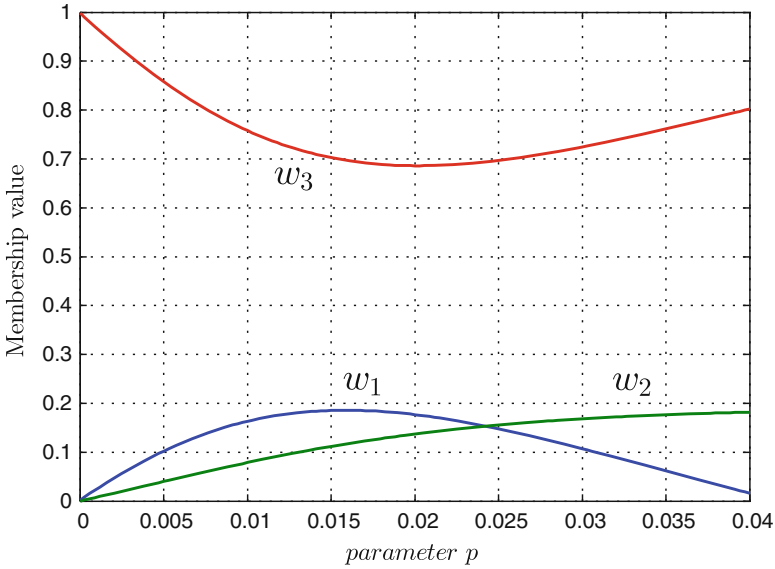

Fig. 2.25 SNNN type weighting function system

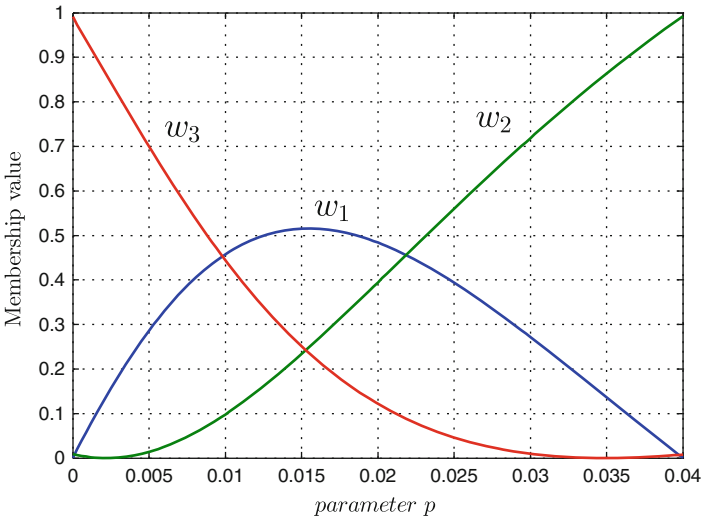

Fig. 2.26 CNO type weighting function system

$$\mathbf{S}_2^{\text{snnn}} = \begin{bmatrix} 1159.1814 & 836.83060 & -2.2077200 \\ -1157.0535 & -839.07308 & 2.2038245 \end{bmatrix}$$

$$\mathbf{S}_3^{\text{snnn}} = \begin{bmatrix} 959.93584 & 1040.8290 & 0.9979833 \\ -960.88224 & -1039.8563 & -0.9970246 \end{bmatrix}$$

We also execute the TP model transformation with CNO transformation that leads to a weighting functions system such that the vertexes form a tight convex hull around the given model,

$$\mathbf{S}(p(t)) = \sum_{r=1}^{3} w_r^{\text{SNN}}(p(t)) S_r^{\text{SNN}} = \sum_{r=1}^{3} w_r^{\text{CNO}}(p(t)) S_r^{\text{CNO}}. \tag{2.137}$$

The weighting functions are given in Fig. 2.26. The vertexes are:

$$\mathbf{S}_1^{\text{cno}} = \begin{bmatrix} 978.02837 & 1021.7823 & 0.9348855 \\ -978.91888 & -1020.8823 & -0.9341519 \end{bmatrix}$$

$$\mathbf{S}_2^{\text{cno}} = \begin{bmatrix} 996.04859 & 1003.8310 & 0.4273445 \\ -996.44795 & -1003.4311 & -0.4272558 \end{bmatrix}$$

$$\mathbf{S}_3^{\text{cno}} = \begin{bmatrix} 959.58489 & 1041.1860 & 1.0046280 \\ -960.53769 & -1040.2067 & -1.0036599 \end{bmatrix}$$

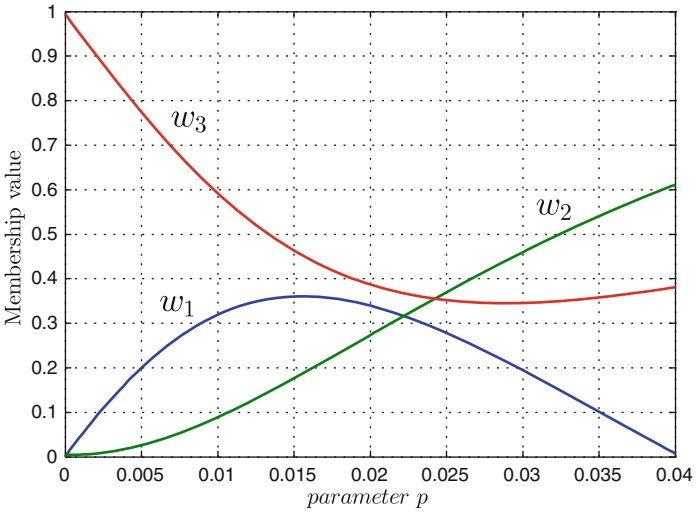

Fig. 2.27 Interpolated weighting functions at $\lambda = 0.53$

We define the linear interpolation between the weighting function systems for all $p(t) \in \Omega$ such that:

$$\mathbf{w}_n^{\lambda}(p(t)) = \lambda \mathbf{w}_n^{\text{CNO}}(p(t)) + (1-\lambda)\mathbf{w}_n^{\text{SNN}}(p(t)), \tag{2.138}$$

where $\lambda \in [0, 1]$. This means that we are tightening the convex hull with λ. Then, using the TP$^+$ model transformation, we determine the interpolated and exact TP model such that:

$$\mathbf{S}(\mathbf{p}(t)) = \sum_{r=1}^{3} w_r^{\lambda}(p(t))S_r^{\lambda}. \tag{2.139}$$

Figure 2.27 shows the case when $\lambda = 0.53$.
The vertexes are:

$$\mathbf{S}_1^{\lambda} = \begin{bmatrix} 980.74020 & 1018.7800 & 0.9897259 \\ -981.68530 & -1017.8291 & -0.9889653 \end{bmatrix}$$

$$\mathbf{S}_2^{\lambda} = \begin{bmatrix} 1018.4915 & 980.84818 & 0.06826419 \\ -1018.5465 & -980.80845 & -0.06872040 \end{bmatrix}$$

$$\mathbf{S}_3^{\lambda} = \begin{bmatrix} 959.61303 & 1041.1578 & 1.0039480 \\ -960.56517 & -1040.1791 & -1.0029807 \end{bmatrix}$$

2.10 Unifying the Weighting Functions

Let us assume that a set of TP functions are given in the form:

$$\mathcal{Y}_l = \mathcal{S}_l \underset{n\in \mathrm{N}}{\boxtimes} \mathbf{w}_{l,n}(x_n), \tag{2.140}$$

$l \in \mathrm{L}$. The goal is to find TP model representations in Ω:

$$\mathcal{Y}_l = \mathcal{T}_l \underset{n\in \mathrm{N}}{\boxtimes} \mathbf{w}^U_n(x_n), \tag{2.141}$$

where the TP models have the same weighting functions systems (superscript "U" means unified). Thus the goal is to ensure:

$$\forall l : \mathcal{S}_l \underset{n\in \mathrm{N}}{\boxtimes} \mathbf{w}_{l,n}(x_n) = \mathcal{T}_l \underset{n\in \mathrm{N}}{\boxtimes} \mathbf{w}^U_n(x_n), \tag{2.142}$$

One obvious way to fulfill these criteria is to execute the Multi-TP model transformation. However, such an approach would result in a significant computational load if HOSVD is executed on a large-sized discretized tensor constructed from all of the individual TP functions. We may have a simplified way here as we know the TP structure of the given TP functions; hence, we can easily determine $\mathfrak{W}^{D(\omega_n,G_n)}_{l,n}$. In order to find the unified set of weighting functions, the matrices of the discretized weighting functions can be stored in the form:

$$\mathbf{H}_n = \left(\mathfrak{W}^{D(\omega_n,G_n)}_{1,n} \cdots \mathfrak{W}^{D(\omega_n,G_n)}_{L,n}\right), \tag{2.143}$$

and compact SVD (or reduced, if complexity reduction is on purpose) can be executed on $\mathbf{H}$ as

$$\mathbf{H}_n = \mathbf{U}_n \mathbf{D}_n \mathbf{V}^T_n = \mathbf{U}_n \mathbf{L}_n. \tag{2.144}$$

If needed, the manipulation of the type of the unified weighting functions (defining SN, NN, CNO, etc. type unified functions) can be integrated at this point with the execution of SVD to obtain such a $\mathbf{U}_n$ that leads to the desired weighting functions. The discretized unified weighting functions, then, have the following form:

$$\mathfrak{W}^{U,D(\omega_n,G_n)}_n = \mathbf{U}_n, \tag{2.145}$$

Given these weighting functions, there are two ways to proceed. The first approach consists in executing the TP^+ model transformation on the given set of TP functions using the predefined weighting functions $\mathbf{w}^U_n(x_n)$ available in the discretized variant $\mathfrak{W}^{U,D(\omega_n,G_n)}_n$. An alternative approach further relaxes the computational requirements. Since we have all the transformation matrices in $\mathbf{L}_n$

we can directly derive the core tensors to the unified weighting functions. Thus, we define the partitions of the matrix $\mathbf{L}_n$ according to the number of the columns of blocks $\mathfrak{W}_{l,n}^{D(\omega_n,G_n)}$ in $\mathbf{H}_n$ as follows:

$$\mathbf{L}_n = \left(\mathbf{V}_{n,1} \cdots \mathbf{V}_{n,L}\right). \tag{2.146}$$

Thus

$$\mathbf{H}_n = \left(\mathfrak{W}_{1,n}^{D(\omega_n,G_n)} \cdots \mathfrak{W}_{L,n}^{D(\omega_n,G_n)}\right) = \mathbf{U}_n \left(\mathbf{V}_{n,1} \cdots \mathbf{V}_{n,L}\right) \tag{2.147}$$

that means

$$\mathfrak{W}_{l,n}^{D(\omega_n,G_n)} = \mathbf{U}_n \mathbf{V}_{n,l}. \tag{2.148}$$

The core tensors can be derived using these transformation matrices as:

$$\mathcal{S}_l \underset{n\in\mathrm{N}}{\boxtimes} \mathfrak{W}_{l,n}^{D(\omega_n,G_n)} = \mathcal{S}_l \underset{n\in\mathrm{N}}{\boxtimes} \mathbf{U}_n \mathbf{V}_{n,l} = \tag{2.149}$$

$$= \left(\mathcal{S}_l \underset{n\in\mathrm{N}}{\boxtimes} \mathbf{V}_{n,l}\right) \underset{n\in\mathrm{N}}{\boxtimes} \mathbf{U}_n = \mathcal{T}_l \underset{n\in\mathrm{N}}{\boxtimes} \mathbf{U}_n = \tag{2.150}$$

$$= \mathcal{T}_l \underset{n\in\mathrm{N}}{\boxtimes} \mathfrak{W}_n^{U,D(\omega_n,G_n)}. \tag{2.151}$$

Thus

$$\mathcal{T}_l = \mathcal{S}_l \underset{n\in\mathrm{N}}{\boxtimes} \mathbf{V}_{n,l}. \tag{2.152}$$

Finally, the third step of the generalized TP model transformation can be executed to determine the continuous weighting functions to one of the pairs of $f_l(\mathbf{x})$ and $\mathcal{S}_l$, since we have unified weighting functions.

2.11 Operations Between TP Functions

Once a set of unified weighting functions are obtained, the addition of TP functions can be performed easily by adding together the corresponding core tensors:

$$\mathcal{S} \underset{n\in\mathrm{N}}{\boxtimes} \mathbf{w}_n(x_n) = \sum_{l=1}^{L} \left(\mathcal{A}_l \underset{n\in\mathrm{N}}{\boxtimes} \mathbf{w}_n(x_n)\right), \tag{2.153}$$

thus

$$\mathcal{S} = \sum_{l=1}^{L} \mathcal{A}_l. \tag{2.154}$$

Further operations are defined between TP models in the following form:

$$\mathbf{S}(\mathbf{x}) = f(\mathbf{A}(\mathbf{x}), \mathbf{B}(\mathbf{x})) = f(\mathcal{A} \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n^A(x_n), \mathcal{B} \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n^B(x_n)), \tag{2.155}$$

can also be numerically reconstructed:

$$\mathbf{S}(\mathbf{x}) = \mathcal{S} \underset{n \in \mathrm{N}}{\boxtimes} \mathbf{w}_n(x_n), \tag{2.156}$$

by executing the TP model transformation on $f(\mathbf{A}(\mathbf{x}), \mathbf{B}(\mathbf{x}))$. Thus, the TP model transformation is executed on the whole function. The result will be exact if it is contained within the TP functions, namely, if the rank of $\mathbf{S}(\mathbf{x})$ is bounded by dimensions. For instance, adding TP functions can easily be derived using the TP model transformation, even in cases when these functions are given using different soft-computing representations (analytical operations between fuzzy and neural network based representations would be very hard as if not impossible). While these claims are quite trivial, they are not applied nearly as much in the literature as their significance would suggest.

2.12 Towards Approximation in Case of Non-TP Functions

The TP model transformation works even in cases where the entire TP model structure of the given function or model are hidden. The only requirement of the presented algorithms is that the model at hand should be discretizable over G. In the case of functions which have a TP model structure (with bounded number of components), once we find all the ranks through the TP model transformation, then irrespective of how many extra gridpoints we add to the discretization, the number of the nonzero singular values will not increase upon the execution of HOSVD. If we have a function that has no TP model representation (with bounded number of components), then the rank of the discretized tensor will increase (at least in one dimension) with the density of G, such that the rank will always be G_n. Since the computational power available limits G_n, it becomes irrelevant in engineering applications whether the given function is a TP function with a higher rank than G_n, or if it is a function that does not have an exact TP function representation. We are faced with the same uncertainty when we have a limitation on the number of resulting weighting functions and we have to execute RHOSVD in any case. If we find that the given function and the resulting TP function or model are equivalent in a numerical sense, then we may suppose that we have found all the

ranks. Therefore, it should be kept in mind that in a mathematical sense, we are always dealing with approximations unless we perform further analysis; however, from an engineering perspective, the possible cases will be numerically equivalent (limitations are imposed only by the available computational resources).

References

1. P. Baranyi, TP model transformation as a way to LMI-based controller design. IEEE Trans Ind. Electron. **51**(2), 387–400 (2004)
2. P. Baranyi, Output feedback control of two-dimensional aeroelastic system. J. Guid. Control. Dyn. **29**(3), 762–767 (2006)
3. P. Baranyi, D. Tikk, Y. Yam, R.J. Patton, From differential equations to PDC controller design via numerical transformation. Comput. Ind. **51**(3), 281–297 (2003)
4. P. Baranyi, Z. Petres, P. Korondi, Y. Yam, H. Hashimoto, Complexity relaxation of the tensor product model transformation for higher dimensional problems. Asian J. Control **9**(2), 195–200 (2007)
5. P. Baranyi, Z. Petres, Sz. Nagy, TPtool — tensor product MATLAB toolbox. Website (2007). http://tp-control.hu/
6. P. Baranyi, Y. Yam, P. Varlaki, *Tensor Product Model Transformation in Polytopic Model-Based Control* (CRC/Taylor & Francis Group, Boca Raton/London, 2013)
7. Sz. Nagy, Z. Petres, P. Baranyi, H. Hashimoto, Computational relaxed TP model transformation: restricting the computation to subspaces of the dynamic model. Asian J. Control **11**(5), 461–475 (2009)
8. L. Szeidl, P. Várlaki, HOSVD based canonical form for polytopic models of dynamic systems. J. Adv. Comput. Intell. Intell. Infor. **13**(1), 52–60 (2009)
9. D. Tikk, P. Baranyi, R.J. Patton, Approximation properties of TP model forms and its consequences to TPDC design framework. Asian J. Control **9**(3), 221–231 (2007)
10. P. Várkonyi, D. Tikk, P. Korondi, P. Baranyi, A new algorithm for RNO-INO type tensor product model representation, in *Proceedings of the IEEE 9th International Conference on Intelligent Engineering Systems* (2005), pp. 263–266
11. Y. Yam, P. Baranyi, C.T. Yang, Reduction of fuzzy rule base via singular value decomposition. IEEE Trans. Fuzzy Syst. **7**(2), 120–132 (1999)

http://www.springer.com/978-3-319-19604-6

TP-Model Transformation-Based-Control Design Frameworks

Baranyi, P.

2016, XXVI, 230 p. 120 illus., 52 illus. in color.,
Hardcover
ISBN: 978-3-319-19604-6