A tensor approach to linear parameter varying system identification

Gunes, Bilal

DOI
10.4233/uuid:44dda417-a658-47d3-998b-48c082c9e989

Publication date
2018

Document Version
Final published version

Citation (APA)
Gunes, B. (2018). *A tensor approach to linear parameter varying system identification*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:44dda417-a658-47d3-998b-48c082c9e989

A TENSOR APPROACH TO LINEAR PARAMETER VARYING SYSTEM IDENTIFICATION

A TENSOR APPROACH TO LINEAR PARAMETER VARYING SYSTEM IDENTIFICATION

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. dr. ir. T. H. J. J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 8 mei 2018 om 12:30 uur

door

Bilal GUNES

werktuigbouwkundig ingenieur, TU Delft, Nederland,
geboren te Amsterdam, Nederland.

Dit proefschrift is goedgekeurd door de promotoren.

Samenstelling promotiecommissie bestaat uit:

Rector Magnificus,	voorzitter
prof. dr. ir. J.W. van Wingerden,	Technische Universiteit Delft, promotor
prof. dr. ir. M. H. G. Verhaegen,	Technische Universiteit Delft, promotor
Onafhankelijke leden:	
Prof. dr. M. Lovera	Politecnico di Milano, Italië
Prof. dr. ir. G. J. T. Leus	Technische Universiteit Delft
dr. ir. R. Tóth	Technische Universiteit Eindhoven
dr. I. Markovsky	Vrije Universiteit Brussel, België
dr. ir. A.C. Schouten	Technische Universiteit Delft
Prof. dr. ir. J. Hellendoorn	Technische Universiteit Delft, reservelid

Printed by: Gildeprint

Front & Back: B. Gunes. The illustration is based on a photograph owned by the Stedelijk Museum and has been used with explicit permission.

CONTENTS

1

INTRODUCTION

In this introductory chapter the use of Linear Parameter Varying state-space identification methods is motivated and the main problem, the curse-of-dimensionality, is discussed. To address this problem it is proposed to use tensor techniques, which will be the main topic of this thesis. Finally, the organization of the thesis is presented.

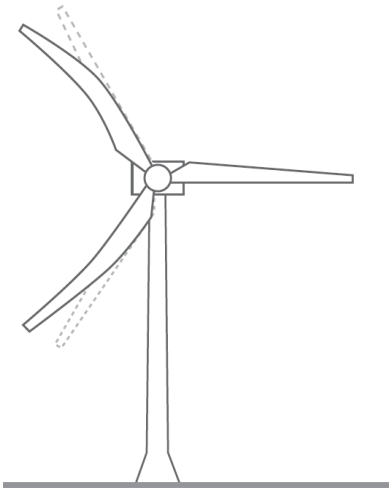
1.1. A BRIEF INTRODUCTION TO LPV SYSTEMS

Linear Parameter Varying (LPV) systems are an intermediate step between Linear Time Invariant (LTI) systems and non-linear systems. LPV systems are linear, but their dynamics depend on an external time-varying signal also called the scheduling sequence. This means LPV systems are time-varying strictly through their scheduling sequence. LPV systems can be accurate models for many industrial applications and there are powerful LPV control design methodologies available. This is especially interesting for applications for which LTI models do not suffice. Three such applications will now be discussed for illustration purposes.

As a first example, consider the whirling modes of a wind turbine (Gebraad et al., 2013, 2011a). The whirling modes describe a specific part of the movement of the rotor blades and are illustrated in Fig. 1.1a. Namely, any two of the blades can bend towards and away from each other in their plane of motion. It is shown in Hansen (2007) that the frequency of these modes equal $\omega \pm \Omega$, where ω is the natural bending frequency of the rotor blades, and Ω is the rotor speed. Hence the dynamics depend on the rotor speed which can be seen as a scheduling sequence. This is illustrated in Fig. 1.1c using measurement data from the wind turbine set-up shown in Fig. 1.1b. During operation, the wind causes the rotor speed to vary erratically over time. Because the rotor speed affects the dynamics, the dynamics also change considerably over time. This limits the application of LTI modelling techniques. Whereas whirling mode dynamics can be captured accurately using LPV models (Gebraad et al., 2013).

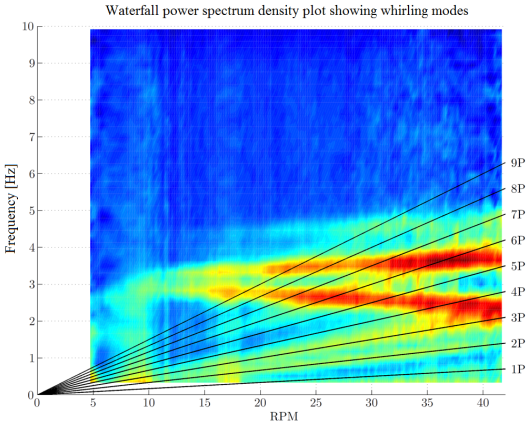
A second example of an application which can be accurately described by an LPV model, is an overhead crane system (Zavari et al., 2014) as illustrated in Fig. 1.2a. By using a varying cable length, containers can be raised and moved at the same time. The

1



(a) This figure shows a whirling mode of a wind turbine. This figure is from Gebraad et al. (2013).

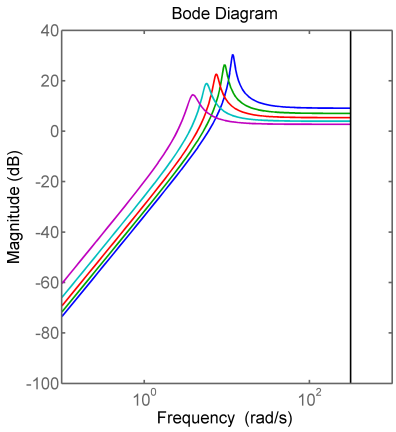
(b) This figure is showing the 600 kW Controls Advanced Research Turbine (CART) III wind turbine in Colorado, United States of America. Photo by Dennis Schroeder, NREL 37892.



(c) This figure shows a power spectrum density plot of the whirling modes of a wind turbine using a waterfall diagram. For different rotor speeds (on x-axis), the power spectrum density is plotted along the y-axis (for frequency). More red colors indicate higher power. Additionally, the harmonic frequencies of the rotor speed are plotted (1P, 2P, ...). The two diverging red lines represent the whirling modes.

Figure 1.1: Figures related to the example on whirling modes of a wind turbine.

(a) This is a photo of a real experimental set-up showing the load and the mechanism to vary the cable length.



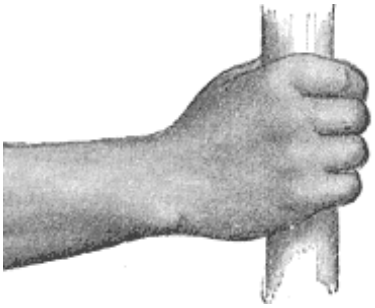
(b) This figure shows a bode magnitude plot of an overhead crane model for several cable lengths. A different color line is drawn per length. In more detail, the discrete-time frequency response from cart position to swing angle is shown.

Figure 1.2: Figures related to the example on an overhead crane.

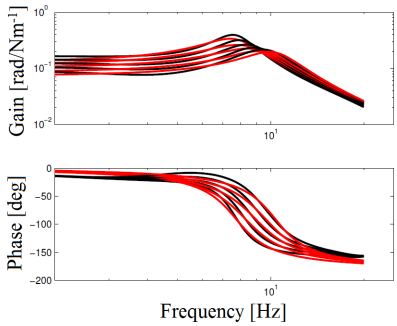
dynamics depend on the cable length and show LPV behaviour. Consider the relation between the position of the crane and the swing angle of the container. In Fig. 1.2b its discrete-time frequency response is shown for several cable lengths for a model. It is visible that the cable length has a major effect on the dynamics as it shifts the resonance frequency. Zavari et al. (2014) shows, using a experimental set-up, that an overhead crane can be very well described by an LPV model with the time-varying cable length as scheduling sequence.

A third example is from bio-mechanics: the joint impedance of a human wrist (van Eesbeek et al., 2013). An illustration of the human wrist is shown in Fig. 1.3a. The damping and stiffness of the human wrist joint are time-varying, because they depend on the voluntary (deliberate) torque. Therefore LTI models are not satisfactory. On the other hand, LPV models can be accurate for this time-varying problem, by using the voluntary torque as scheduling sequence. The LPV behaviour is clearly visible in Fig 1.3b, which shows a large variation of the frequency response at different values of the scheduling.

For these three examples and many other applications (see Mohammadpour and Scherer (2012) or Sename et al. (2013)), LPV systems play a key role for two reasons. Firstly, LPV models can capture time-varying behaviour allowing for higher accuracy than LTI models. As a result, the performance of subsequently designed (LPV) controllers can be higher (Tóth et al., 2011). Secondly, techniques are available (such as Scherer (2001)) to design LPV controllers which can guarantee robust performance. This guarantee does not require the scheduling sequence to be constant. In contrast, LTI and (the LPV precursor) classic gain-scheduling control (Rugh and Shamma, 2000; Leith and Leithead, 2000) generally give no guarantees on stability or performance,



(a) This figure shows an illustration of a human wrist. The human wrist can use its muscles to exert torque and has complex stiffness and damping. However, for small rotations these dynamics can be approximated by an LPV model (van Eesbeek et al., 2013). This figure is from Gray (1918).



(b) In black the Frequency Response Functions (FRF) of the LPV system for different constant values of the scheduling sequence and in red the fitted parametrized models (van Eesbeek et al., 2013) are shown. More specifically, the input is wrist angle and the output is torque. This figure is from (van Eesbeek et al., 2013).

Figure 1.3: Figures related to the example on the human wrist joint.

unless the scheduling sequence is constant. As a result there is no stability guarantee when the scheduling sequence is changing. However, for many applications such as the whirling mode example the scheduling sequence is constantly varying. Additionally, the technique of Scherer (2001) directly allows for Multiple Input Multiple Output (MIMO) LPV controller design. In many industrial applications, instead of a Multiple Input Multiple Output (MIMO) LPV controller, several Single Input Single Output (SISO) gain-scheduling controllers are still used (Bossanyi, 2000; Van Kuik and Peinke, 2016). However, the advantage of designing a single MIMO controller instead of multiple SISO controllers, is that any coupling between different input-output pairs can directly be taken into account and exploited (Skogestad and Postlethwaite, 2007).

While the model-based controller design for these LPV systems has been well developed, a *model* is required to design a controller. The accuracy of this model directly affects the performance of the controller. These models can be obtained in two ways. The first way is to use first principles modelling, which uses the laws of physics. However, this requires specialist knowledge as it is non-trivial which physics effects should be included (Mohammadpour and Scherer, 2012). Furthermore certain quantities, for example stiffness and damping, can be unknown and require dedicated experiments. This approach is often used as a starting point for the second modelling technique. The second modelling approach is system identification, which is the scope of this thesis and presented in the next section.

1.2. LPV SYSTEM IDENTIFICATION

In the previous section it was argued that for the design of LPV controllers an LPV model is needed and it was mentioned that this model can be obtained using system identifica-

tion. System identification is a mathematical framework which allows the user to obtain models of systems from their experimental input-output data. Whereas with first principles modelling it is non-trivial to determine which effects are dominant and should be included. In the next two subsections we present mathematical formulations of LPV models (model structures) and discuss the state-of-the-art identification methods.

1.2.1. MODEL STRUCTURE

Firstly, a description using state-space systems is presented. The LPV behaviour appears through the state-space matrices which are known functions of the scheduling sequence. This LPV model structure using discrete-time state-space representation is given by:

$$\begin{aligned} x_{k+1} &= A(\mu(k))x_k + B(\mu(k))u_k + w_k \\ y_k &= C(\mu(k))x_k + D(\mu(k))u_k + v_k, \end{aligned} \tag{1.1}$$

where the signals x_k, u_k, y_k, w_k and v_k are the state, input, output, process noise and measurement noise, and the matrices A, B, C and D are the time-varying state, input, output and feed-through matrices. Because this representation is in discrete-time, the sample number appears as the index k. Notice that the state-space matrices are functions of the scheduling sequence denoted by μ. This type of description is used by state-space methodologies.

In contrast, the type of description used by input-output methodologies is the input-output description, presented here without noise:

$$y_k = -\sum_{i=1}^{n_a} a^{(i)}(\mu(k))y_{k-i} + \sum_{j=1}^{n_b} b^{(j)}(\mu(k))u_{k-j}, \tag{1.2}$$

where n_a and n_b are model orders, $a^{(i)}$ and $b^{(j)}$ are coefficient matrices and a function of the time sample k.

The focus of this thesis will be on discrete-time LPV model structures with known scheduling sequence. This focus will be motivated in two parts. Firstly, LPV model structures exist for both discrete-time and continuous-time, and control design methodologies exist for both as well (Scherer, 2001; Dong and Wu, 2007)[1]. However, experimental data is always sampled data and thus discrete. Therefore the focus will be on discrete-time systems and models. Secondly, in many applications the scheduling sequence is measurable, such as the examples of the previous section. It is also possible to formulate an identification problem with unknown scheduling, which would lead to much more involved identification problems. In this thesis, the focus will be on exactly known scheduling sequences, which is common in LPV identification. Also, frequency-domain model structures and identification methods will not be discussed, because that category of methods is still in its early stage with only few publications in the literature (Goos et al., 2017). In the next subsection we present an overview of LPV identification methods.

[1] It has to be remarked that the transformation of continuous-time LPV systems to discrete ones is non-trivial and generally leads to more complicated functions of the scheduling sequences to appear (Tóth et al., 2008).

1

1.2.2. SYSTEM IDENTIFICATION METHODS

In this subsection we briefly review available LPV system identification methods.

GLOBAL AND LOCAL METHODS

One way to categorize methods is by division into global and local methods. The two categories of methods have different applications and purposes. Local methods, such as the ones described in (Tóth, 2010; De Caigny et al., 2009; Shamma, 2012), perform several LTI identification experiments. That is, at every experiment the scheduling sequence is kept at a constant value. Every experiment returns one LTI model, and these models can be combined into an LPV model. Obtaining LTI models is well-understood (van der Veen et al., 2013; Ljung, 1999), however the combination step requires special care. The first and common combination approach is by interpolation of the models. The second approach is to fit a parametrized global LPV model to the LTI models. This parametrization can exploit prior knowledge. This is the core of glocal methods, whose name is a contraction of 'global' and 'local'. In this thesis, glocal methods will be considered a sub-class of local methods. Glocal methods, such as the methods of Vizer et al. (2013); Petersson and Löfberg (2014); Mercere et al. (2011), avoid the problems of interpolation at the cost of a more difficult optimization problem. This approach is promising but arguably not yet mature (Sename et al., 2013). Regardless, the effectiveness of local methods depends on whether the scheduling sequences can be kept reasonably constant during the identification experiments (De Caigny et al., 2009). For applications where this is possible, such as for example high performance positioning devices (van der Maas et al., 2015; Tóth et al., 2011) and distillation columns (Bachnas et al., 2013), local methods can yield good results (De Caigny et al., 2009; Tóth, 2010; Shamma, 2012). However for a number of applications these experiments are not possible.

On the other hand, global methods work with data from experiments during which the scheduling sequence can vary and yield an LPV model. In contrast to local methods, data from a single experiment is used. This does mean the relevant dynamics have to be sufficiently excited during the experiment. There exist several global methods, both in input-output and state-space setting. Some examples are (van Wingerden and Verhaegen, 2009; Larimore and Buchholz, 2012; Golabi et al., 2017). These will be discussed in the next subsection. In the remainder of this thesis only global methods will be discussed.

Another way to divide methods is division into input-output and state-space methods. State-space methods can be further divided into subspace and state-space refinement methods. These three categories will be reviewed in the next three subsections.

INPUT-OUTPUT METHODS

Input-output methods are all methods which return input-output LPV models, and they have received considerable attention in literature (Tóth, 2010; Laurain et al., 2010; Butcher et al., 2008; Bamieh and Giarre, 2002). A few of them will be discussed to illustrate the vast literature. The (local) method of Tóth (2010) uses Orthonormal Basis Functions (OBF) to benefit from the well worked out theory of OBF's for LTI problems. These OBF are a set of user-chosen orthogonal and normalized functions which can be used to approximate a complex function, such as an impulse response. One of the benefits is that the obtained model simplifies subsequent control design. The methods

of Laurain et al. (2010); Abbas and Werner (2009) use an Instrumental Variable (IV) method to deal with more complex noise structures. Most identification methods assume and require some statistical properties of the input signal to produce unbiased estimates. If this assumption does not hold, one can construct and use an (instrumental) variable which does have these properties. There also exist input-output methods with Bayesian regularization, such as in Darwish et al. (2015); Golabi et al. (2017). Regularization is a way to introduce a bias-variance trade-off in estimates, where some small bias is introduced to reduce large variances. The intuitive extension of input-output models from LTI to LPV has also allowed specialized methods, such as for problems with spatially interconnected subsystems (Liu et al., 2016; Belforte et al., 2005).

However, the preferred model structure for mainstream LPV control design methodologies is state-space (Scherer, 2001), and transformation from input-output to those state-space models has been shown to be problematic in the LPV setting (Tóth et al., 2012). Namely, such a transformation can introduce very complex scheduling dependency or non-minimality. State-space methods have also received considerable attention (van Wingerden and Verhaegen, 2009; Larimore et al., 2015; Cox and Tóth, 2016a,b). The methods can be seen as extensions of different LTI approaches. Additionally, they produce state-space models and extend naturally to the MIMO case. In the next two subsections state-space methods are reviewed: first subspace then state-space refinement methods.

SUBSPACE METHODS

Subspace methods are state-space methods which use linear parametrization and involve convex optimization problems [2]. However, in the LPV setting, a linear parametrization can result in a huge number of parameters. More specifically, 'huge' refers to exponential explosion. This will be illustrated in the next section. This huge number of parameters causes problems with memory and computation costs and can lead to poorly conditioned problems. This problem appears as the number of parameters can vastly exceed the number of data points. In some special cases, the scheduling sequence can exhibit some structure which can simplify the problem. For example, tailored methods are available for periodic scheduling (Felici et al., 2007), noise scheduling (Favoreel et al., 1999) or piecewise constant scheduling (van Wingerden et al., 2007). However, generally the scheduling sequence is arbitrarily varying, such as for the presented wind turbine example. For these cases, several methods exist. The identification method of Cox and Tóth (2016a) uses correlation analysis: an approach based on the correlation of signals. In its current form, it does assume the noise and input to be white noise. The first two steps of the method of Cox and Tóth (2016b) can also be used to obtain an LPV state-space estimate in convex manner. However, the third non-convex refinement step is required to obtain some of the important properties of the total method. The method of Larimore and Buchholz (2012) (see also (Larimore et al., 2015)) uses canonical variate analysis, which estimates the states by first estimating some other, 'canonical' states. In Larimore and Buchholz (2012) it is claimed but not shown that this closed-loop method does not have an exponential explosion in the number of variables. That is, the

[2] Optimization problems can roughly be divided in two categories of difficulty: convex and non-convex. Non-convex problems can suffer from local minima.

algorithm is not publicly available. It is worth remarking that in Chiuso (2010) it is shown for the LTI case that this method is asymptotically [3] equivalent in terms of variance to the Predictor-Based Subspace IDentification (PBSID) method of Chiuso (2007). In this thesis the focus will be on the latter. More specifically, the methods of van Wingerden and Verhaegen (2009); Gebraad et al. (2011a) use a predictor-based approach (Chiuso, 2007), and will be discussed in detail in the next section. This approach is based on the assumption that the state-observer, which estimates the states of the system based on inputs and outputs, is asymptotically stable. This is quite common for LTI subspace methods (van der Veen et al., 2013) and allows dealing with closed-loop data. The huge parameter count is tackled by assuming that the solution of the underling regression problem is the minimum-norm solution (see Chapter 3 for a detailed discussion). This greatly reduces the parameter count, and provides means to perform computation efficiently. The ill-condition (and thus high variance) is further tackled using regularization. Several regularization techniques have been proposed in literature, but the underlying LPV structure is not exploited.

STATE-SPACE REFINEMENT METHODS

State-space refinement methods are methods which return state-space models, but involve a non-convex optimization problem which has to be solved. The non-convexity appears, because non-linear parametrizations are used to avoid huge parameter counts and problems with memory or computation. This does mean these methods require initialization by an initial estimate. Notice that there are many non-linear parametrizations and cost functions possible. The methods of Cox and Tóth (2016b); Verdult et al. (2003); Lee and Poolla (1999) directly (element-wise) parametrize the LPV state-space matrices. The method of Cox and Tóth (2016b) also deploys regularization, has local convergence and can deal with closed-loop data. However, notice that there do not exist any convex LPV subspace methods which directly parametrize the state-space matrices. As a result, the cost function of the convex method which produces the initial estimate is inherently different from the cost function of these subsequent refinement methods. This may give a possible disadvantage, since the initializing and refining methods are not in line in terms of cost functions. This raises the question whether refinement methods can be developed which do not have this possible disadvantage.

OPEN LOOP AND CLOSED LOOP

In this paragraph, the problem of closed-loop identification is discussed. Closed-loop identification is identification using 'closed-loop data': data obtained under closed-loop operation (with active controller). Closed-loop methods are methods which can deal with this type of data. Other methods can return biased estimates when given closed-loop data (Ljung, 1999). Inability to deal with closed-loop data is a problem, because open-loop data may not be available. For many industrial applications, such as wind turbines, the system under open-loop operation is either unstable or has poor performance, which makes open-loop experiments very costly. Also, for the human wrist example the data is inherently closed-loop data due to the control loops in the body. Additionally, "for

[3] (as the number of data points tend to infinity)

model-based control design, closed-loop identification gives better performance" (Hjalmarsson et al., 1996). Therefore the focus of this thesis is on LPV methods that can deal with (both open- and) closed-loop data.

In the remainder of this chapter, the focus will be on closed-loop predictor-based global subspace methods and state-space refinement methods. These predictor-based methods are reviewed in the next section.

1.3. PREDICTOR-BASED METHODS

In the previous section predictor-based methods have been motivated and in this section they will be presented in detail. Before presenting the bottleneck 'curse-of-dimensionality' and the challenge of this thesis, the predictor-based model structure and its key assumption will be reviewed.

1.3.1. MODEL STRUCTURE

In this subsection, the predictor-based model structure (Chiuso, 2007; van Wingerden and Verhaegen, 2009) is presented. For clarity, the model structure is first presented for the LTI case and afterwards for the LPV case. Furthermore, before presenting the predictor-based representation, the equivalent innovation form is presented because it has a more clear relation to the general LPV system representation of (1.1). Starting from that equation, the innovation representation is:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Ke_k \\ y_k &= Cx_k + Du_k + e_k, \end{aligned} \tag{1.3}$$

where e is the innovation signal and K is the innovation gain. This equations allows direct presentation of its LPV variant. It should be noted that for LPV systems, moving to this innovation representation removes the capability to model some of the more complex noise structures (Cox and Tóth, 2016c) like in (1.1). For the LPV variant of this equation, the state-space matrices become known functions of the scheduling sequence μ. Many methods, such as Cox and Tóth (2016b); van Wingerden and Verhaegen (2009), choose these functions as both affine and static functions of μ. Static dependency means that only the current sample of μ affects the time-varying state-space matrices at that sample: $A(\mu(k)) = A(\mu_k)$. Affine dependency means that the state-space matrices are affine functions of the scheduling sequence μ. This can also be regarded as a weighted sum of local models. The result is that the LPV state-space matrices take the following form:

$$A(\mu(k)) = \sum_{i=1}^{m} \mu_k^{(i)} A^{(i)}, \tag{1.4}$$

where m is the number of time-varying variables in the scheduling sequence, i is an index and $\mu_k^{(i)}$ is a scalar. A similar expression is used for the B, C and D matrices. This yields the following LPV system representation:

$$\begin{aligned} x_{k+1} &= \sum_{i=1}^{m} \mu_k^{(i)} A^{(i)} x_k + \mu_k^{(i)} B^{(i)} u_k + \mu_k^{(i)} K^{(i)} e_k \\ y_k &= \sum_{i=1}^{m} \mu_k^{(i)} C^{(i)} x_k + \mu_k^{(i)} D^{(i)} u_k + e_k \end{aligned} \tag{1.5}$$

1

Before moving to the predictor-based representation, the output equation will be simplified for the sake of presentation and simplicity of derivation, similar to what has been done in van Wingerden and Verhaegen (2009). This will not make the bottleneck of predictor-based methods trivial. This bottleneck will be presented in the next subsection. The simplification is that C will be parameter-invariant and D will be the zero matrix. Afterwards, the predictor-based equation can be obtained by substituting the bottom equation into the top equation to remove the innovation e from the top equation:

$$x_{k+1} = \sum_{i=1}^{m} \mu_k^{(i)} \left(\tilde{A}^{(i)} x_k + \bar{B}^{(i)} \begin{bmatrix} u_k \\ y_k \end{bmatrix} \right) \tag{1.6a}$$

$$y_k = C x_k + e_k, \tag{1.6b}$$

where $\tilde{A}^{(i)}$ is $A^{(i)} - K^{(i)}C$ and $\bar{B}^{(i)}$ is $[B^{(i)}, K^{(i)}]$. Notice that this predictor-based representation is very similar to a basic state-observer (which estimates the states x using the inputs and outputs). This form allows making the following assumption on state evolution to simplify identification.

1.3.2. KEY ASSUMPTION

In this subsection the key assumption of predictor-based methods is reviewed. This assumption relates to the evolution of states. The effect of an initial state on a future state is described by the time-varying state transition matrix ϕ (van Wingerden and Verhaegen, 2009):

$$\tilde{A}_k = \sum_{i=1}^{m} \mu_k^{(i)} \tilde{A}^{(i)} \tag{1.7a}$$

$$\phi_{j,k} = \tilde{A}_{k+j-1} \dots \tilde{A}_{k+1} \tilde{A}_k \tag{1.7b}$$

Notice that this matrix describes how an initial state would evolve by itself without inputs, outputs or noise. More specifically, it describes the relation between x_k and x_{k+j}. Predictor-based methods, both LTI and LPV, and several other subspace methods (see van der Veen et al. (2013)) assume this matrix to be exactly zero for large enough windows:

$$\phi_{j,k} \approx 0 \; \forall \; j \geq p, \tag{1.8}$$

where p is the past window. That is, without inputs, outputs or noise, any initial state is assumed to become approximately zero after some time steps. This is equivalent to assuming that an initial state does not affect states far enough in the future. This approximation has some favourable properties. If the predictor-based system (1.6) is (uniformly exponentially) stable [4], then the approximation error of (1.8) can be made arbitrarily small by increasing p (Knudsen, 2001). This approximation results in a bias in subsequent estimations, which disappears as p goes to infinity. But the effect is hard to quantify for finite p (Chiuso, 2007; Knudsen, 2001). Under this assumption, states are assumed not to affect future states which come p samples later. As a result, x_{k-p} has no

[4] The interested reader is referred to Verdult and Verhaegen (2002) for the detailed discussion and equation of this condition

effect on x_k and hence y_k. This in turn means the current output y_k can be expressed using current and only past inputs and outputs without the states. Hence this yields a simple relation between inputs and outputs, which can be estimated in a straight-forward manner. This estimate can then be used to obtain an estimate of the LPV state-space matrices (van Wingerden and Verhaegen, 2009). However, a bottleneck appears in the first estimation step: a 'curse-of-dimensionality'.

1.3.3. CURSE-OF-DIMENSIONALITY

Predictor-based identification methods have a bottleneck in their first estimation step: the 'curse-of-dimensionality'. That is, the number of parameters to estimate scales exponentially with the past window. Before specifically defining this effect and the problems it causes, first this effect is illustrated.

ILLUSTRATION

The parameters to be estimated are presented to illustrate the scale of the problem. Firstly, define these parameters as the (LPV sub-)Markov parameters. For brevity, the matrix K is fixed at zero in this subsection. Then, the parameters describe the relation between the current outputs and the current and past inputs. Hence for this case $\tilde{A}^{(*)} = A^{(*)}$ and $\tilde{B}^{(*)} = B^{(*)}$. For the LTI case, these parameters are just the elements of the matrices:

$$CB, \tag{1.9a}$$

$$CAB, \tag{1.9b}$$

$$CAAB, \ldots \tag{1.9c}$$

where these matrices and their relation to the past window p have been defined in the previous subsection. Notice that the number of parameters scales linearly with the past window p. That is, for $p = 1$ only CB is estimated, for $p = 2$ CB and CAB, etcetera. However in the LPV predictor-based case (1.6), there is not just one A matrix but m of them. As a result, every possible combination must be considered. For brevity, an LPV system is considered where only the matrix A is parameter dependent, and $m = 2$. Then the parameters to be estimated are the elements of:

$$CB, \tag{1.10a}$$

$$CA^{(1)}B,\ CA^{(2)}B, \tag{1.10b}$$

$$CA^{(1)}A^{(1)}B,\ CA^{(1)}A^{(2)}B,\ CA^{(2)}A^{(1)}B,\ CA^{(2)}A^{(2)}B, \ldots \tag{1.10c}$$

Notice that now for $p = 2$ three matrices, and for $p = 3$ seven matrices have to be estimated. This illustrates how the parameter count scales exponentially with the past window. In contrast, the scaling was linear for the LTI case. Notice that increasing the past window by one roughly doubles the number of parameters for the LPV case. For this example, the next steps would be adding 8, 16 and 32 new matrices to estimate respectively. This exponential increase is also illustrated in Fig. 1.4. The problems which this increase causes are described next.

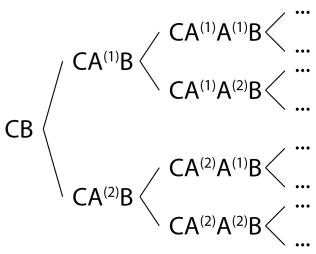


Figure 1.4: This figure shows how the parameters to estimate presented in (1.10) grow in number for increasing past windows. For $p = 1$ only the first column of parameters is estimated, for $p = 2$ the first and second columns, etcetera. Notice that every increment of p by one roughly doubles the amount of parameters.

DEFINITION AND PROBLEMS

Before presenting the specific problems, first the 'curse-of-dimensionality' is defined:

Remark 1.3.1 *In this chapter presence of 'curse-of-dimensionality' refers to that the number of parameters to estimate in the first estimation step scales exponentially with the past window.*

Since the past window can not be chosen too small as argued in the previous subsection, the 'curse-of-dimensionality' generally plays an important role. Namely, it gives three specific problems:

1. The memory cost scales exponentially with the past window.
2. The computational cost scales exponentially with the past window.
3. The parameters quickly become too numerous, possibly greatly exceeding the number of data points. This can make the first estimation problem ill-conditioned and increase the variance of the estimate.

These problems, combined with the fact that state-of-the-art methods don't exploit the underlying structure of the LPV sub-Markov parameters, lead to the challenge of this thesis.

1.3.4. CHALLENGE

The challenge of this thesis is built on the previously described three problems caused by the 'curse-of-dimensionality'. Regarding the first two problems, memory and computational cost, the challenge will require novel methods to be 'curse-of-dimensionality'-free in memory and computation. That is, these costs should scale slower than exponential, for example as a power. Regardless, it should be remarked that overly large past windows can increase variance. Regarding the third problem, ill-condition (or high variance), several approaches have been proposed in literature. This high-variance effect appears for all reasonable past windows. Hence, it is an important and open problem. Therefore, the challenge is:

Challenge: develop methods which are 'curse-of-dimensionality'-free in memory and computation and have reduced variance.

In this thesis the underlying structure of the LPV sub-Markov parameters will be exploited to solve this challenge in an elegant way. Before presenting the proposed approach, an overview of other approaches which try to solve the challenge mentioned above is presented. The different approaches can be divided in two groups: subspace and state-space refinement methods. Firstly, subspace methods greatly reduce the parameter count by assuming the solution of the regression problem is the minimum-norm solution. This allows 'curse-of-dimensionality'-free storage and computation[5]. The ill-condition problem is further tackled using regularization (van Wingerden and Verhaegen, 2009; Gebraad et al., 2011b). Regularization can be more effective if the underlying structure is exploited better. This is also the point where there is vast room for improvement compared to existing subspace methods. Secondly, state-space refinement methods use a non-linear parametrization [6] with few parameters which allows them to be 'curse-of-dimensionality'-free in memory and computation. Furthermore, this non-linear parametrization has less spurious freedom which can improve variance. This does come at the cost of having a non-convex optimization problem which requires initialization by an initial estimate. One approach is to circumvent the predictor-based approach and its 'curse-of-dimensionality' by directly parametrizing the state-space matrices Verdult et al. (2003); Lee and Poolla (1999); Cox and Tóth (2016b). However, as discussed in the previous section, this leads to a discrepancy between the cost functions of the initializing subspace method and subsequent refinement method which may hurt model quality. This raises the question whether the 'curse-of-dimensionality' can be tackled by a different direction of refinement methods, namely predictor-based ones. To summarize, this 'curse-of-dimensionality' is still an open problem.

One possible approach to tackle this 'curse-of-dimensionality' problem is to use *tensor techniques*. Tensor techniques can exploit multi-linear structure to break a 'curse-of-dimensionality'. The underlying LPV structure is such a multi-linear structure, as discussed in the previous subsection. Therefore in this thesis, tensor techniques will be used to tackle the stated challenge. Both novel tensor subspace and state-space refinement methods will be developed. In the next section tensors, tensor techniques and their relation to predictor-based identification are introduced.

1.4. TENSOR TECHNIQUES

In the previous section it was argued that the 'curse-of-dimensionality' of the LPV predictor-based identification problem can be tackled using tensor techniques. In this section, the connection between the two is discussed and tensors and tensor techniques are further reviewed.

[5] by using Kronecker algebra (van Wingerden and Verhaegen, 2009)

[6] That is, the model output is non-linear in the parameters of the parametrization.

1

1.4.1. TENSOR TECHNIQUES IN LITERATURE

Tensors and tensor techniques have been used successfully in many applications, such as structured data fusion (Sorber et al., 2015), blind signal separation (Cichocki et al., 2009), denoising (Signoretto et al., 2010), higher order statistics (De Lathauwer and Vandewalle, 2004) and chemometrics (Smilde et al., 2005). These tensor techniques are also deployed in TensorFlow (Abadi et al., 2016) by Google and even have specialized hardware for it. This is because tensors can represent multi-linear structure more intuitively and tensor techniques can be used to exploit this structure.

To illustrate how tensor techniques can be used to break 'curse-of-dimensionality', a problem from scientific computing (Vervliet et al., 2014; Khoromskij, 2012) is presented as an example. This problem inherently has a 'curse-of-dimensionality', because it involves discretization of a function over all its variables. For example, consider a function $y = f(a,b,c)$ and discretize it over a grid of different values of a, b and c. Suppose the user wishes to use a fine discretization, where every single variable is discretized at a large number of values. Let a, b and c be between zero and one and discretized in 100 points: this gives 100^3 grid points. Notice that as the number of variables increase, the number of grid points increase exponentially. For three variables there are one million grid points, but for four variables one hundred million and for six variables one trillion grid points. As a result, even modest variable counts yields unmanageably many grid points. This illustrates an inherent 'curse-of-dimensionality'.

This problem has been tackled in literature by reformulating it using tensors and then constructing tensor decompositions. Tensor decompositions are, like matrix decompositions, condensed and possibly approximate representations of tensors. That is, a huge tensor does not have to be stored element-by-element, but can also be stored in a less memory-consuming way. A thorough discussion will be presented in 1.4.3. The huge grid in three variables can be seen as a three-dimensional tensor. A tensor generalizes a matrix in that it can have more than two dimensions. For example, in addition to height and width it can also have depth. See Fig. 1.5 for an illustration. For this case, the dimensions link to a, b and c separately. For now, suppose the function to discretize is $y = abc$. Notice that this is a very basic form of multi-linear structure. Namely, a function is multi-linear if it is linear in every variable it has separately. Then, this multi-linear structure allows condensely storing the huge 100-by-100-by-100 grid or tensor. Namely, instead of element-wise storage a condense tensor decomposition can be stored. For example, the tensor can be described using three vectors of length 100 relating to the three variables. Each vector equals $[0.01,\ 0.02,\ \ldots,\ 1]$. Notice that this condensed representation reduces the number of elements to store from 100^3 to 300. Also notice that the condensed representation no longer has a 'curse-of-dimensionality', as it scales linearly with the number of variables. Additionally, decompositions can be obtained directly without storing original, huge tensors in memory by using randomized sub-block approaches (Vervliet and De Lathauwer, 2016). This is an example of how tensor techniques can be used to break a 'curse-of-dimensionality'.

The use of tensors and tensor techniques for LPV predictor-based identification is discussed in the next subsection.

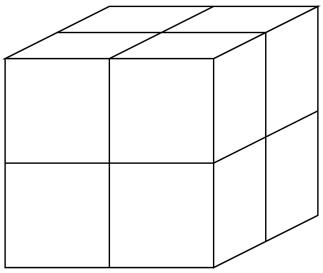


Figure 1.5: This figure shows a cube consisting of eight smaller cubes. Suppose every cube contains one value, then it can be seen as a three-dimensional tensor of size two-by-two-by-two.

1.4.2. RELATION BETWEEN TENSORS AND LPV PREDICTOR-BASED IDENTIFICATION

In this subsection the relation between LPV predictor-based identification and tensors are discussed.

Tensor techniques can exploit multi-linear structure. The LPV identification problem has this structure, and will now be presented explicitly. The parameters which suffer from the 'curse-of-dimensionality', the LPV sub-Markov parameters, are (1.10):

$$CB, \tag{1.11a}$$

$$CA^{(1)}B,\ CA^{(2)}B, \tag{1.11b}$$

$$CA^{(1)}A^{(1)}B,\ CA^{(1)}A^{(2)}B,\ CA^{(2)}A^{(1)}B,\ CA^{(2)}A^{(2)}B,\ldots \tag{1.11c}$$

Notice that the multi-linear structure of these parameters is clearly visible: each is a product of several matrices. More specifically, the LPV sub-Markov parameters are every possible combination of the $A^{(*)}$ of any product length, pre-multiplied by C and post-multiplied by B. This is an exact multi-linear structure. This structure allows natural representation using tensors, possibly with condense decompositions. More importantly, this also motivates use of tensor techniques, some of which can break the 'curse-of-dimensionality'. As a result, tensor algebra opens the possibility to develop 'curse-of-dimensionality'-free LPV state-space identification methods.

It is worth remarking that tensor decompositions have been used for LPV systems before (see Petres (2006) and the references therein), for the decomposition of entire LPV systems (and not LPV sub-Markov parameters) for purpose of control design.

In the next two subsection, detailed definitions on tensors are presented.

1.4.3. ADDITIONAL TENSOR DEFINITIONS

In this subsection, some key definitions are presented. The matrix variants are discussed first for clarity.

First consider the matrix case. Notice that for very large size matrices, storing every single element of the matrix may be costly in memory. In that case a decomposition can be used: a way of representing the matrix in a condensed manner. One such decomposition is matrix Singular Value Decomposition (SVD), which has received considerable

1

attention in literature (Deprettere, 1989; Vaccaro, 1991; Moonen and De Moor, 1995). For example, using the SVD, a matrix M can be decomposed into a set of singular values and the left- and right-'singular' vectors:

$$M = U \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots \end{bmatrix} V^T, \tag{1.12}$$

where the columns of U are the left-singular vectors, the columns of V are the right-singular vectors and σ_* are the singular values, whose properties are omitted for the sake of brevity. Notice that the SVD is a different way of representing the same matrix. The number of (non-zero) singular values is exactly the rank of the matrix. A rank one matrix of size 1000-by-1000, can be stored with just 2001 variables through the SVD. As the rank increases, the decomposition becomes less effective and finally defective. If a matrix has the largest possible rank for its size, then the matrix is called 'full-rank' and otherwise 'low-rank'. Regardless, for matrices with large ranks, the representation may not be as condensed as desired. For these cases, compression is an option. Compression removes the smallest singular values, reducing the number of variables of the decomposition, but introducing an approximation error.

These definitions allow defining some tensor notions. Like matrices, tensors can also be decomposed and compressed. This can be done using matrix compression techniques after flattening[7] the tensor, or directly using tensor compression techniques. Notice that tensor compression techniques are able to exploit multi-linear structure and break 'curse-of-dimensionality'. Some notions however, do not extend trivially to tensors. While the notion of matrix rank is well-understood and unique, tensors have several rank notions. Each corresponds to a different approach of tensor decomposition. The three tensor decompositions, which are the best understood and have the most attention received in literature, will be discussed in the next subsection.

1.4.4. TENSOR DECOMPOSITIONS

In this subsection tensor decompositions will be reviewed as they allow exploiting multi-linear structure and breaking 'curse-of-dimensionality'. Several tensor decomposition approaches exist, but only the three which are best understood and have received the most attention in literature are briefly reviewed. Their (low-)rank notions are explained, because low-rank properties of tensor decompositions play a key role in breaking 'curse-of-dimensionality'. Each of the three decompositions is reviewed in further detail in order in a subsequent chapter.

POLYADIC DECOMPOSITION

The polyadic (tensor) decomposition (Kolda and Bader, 2009; De Lathauwer, 2009) has been successfully applied in chemometrics (Smilde et al., 2005), neuroimaging (Zhou et al., 2013) and biomedical signals (De Vos et al., 2007), to name a few. This decomposition approach describes tensors as a sum of (polyadic) rank one tensors. Polyadic rank one tensors are tensors which are constructed as the (outer) product of vectors (Zhou

[7] flattening is the procedure of reorganizing the elements of a tensor into a matrix

et al., 2013). This allows describing a tensor by only storing a number of vectors. The memory usage depends on the polyadic rank: the number of polyadic rank one tensors summed. In this thesis a polyadic decomposition is said to be low-rank, if its polyadic rank is smaller than the maximally needed polyadic rank to describe every tensor of the same size[8].

MULTI-LINEAR SINGULAR VALUE DECOMPOSITION (MLSVD)

The Multi-Linear Singular Value Decomposition (De Lathauwer et al., 2000) has, amongst others, applications in harmonic retrieval (Papy et al., 2009) and image processing (Vasilescu and Terzopoulos, 2002). This tensor decomposition approach is a generalization of the matrix SVD to the tensor case. It decomposes a tensor into one 'core tensor' and several (orthogonal and normalized) matrices. If the core tensor is small, then a considerable memory usage reduction can be obtained. Since the size of this core tensor determines the effectiveness of this decomposition, it is defined as the multi-linear rank. Notice that hence the multi-linear rank is a tuple, for example it can be $(3,4,5)$. Therefore in this thesis an MLSVD is said to be low-rank if its multi-linear rank tuple is smaller than the original tensor size in all dimensions.

TENSOR NETWORKS

Tensor networks (Batselier et al., 2017; Chen et al., 2016; Oseledets, 2011) [9] can be numerically efficient and have applications in for example molecular dynamics (Schollwöck, 2005) and Volterra systems (Batselier et al., 2017; Chen et al., 2016). This decomposition decomposes a tensor into a series of three-dimensional tensors. These tensors are 'multiplied' with each other (see Chapter 4 for details) to obtain the original tensor. The sizes of the dimensions across which 'multiplication' finds place, determine the effectiveness of the decomposition and are defined as the 'tensor network ranks'. Finally, a tensor network is low-rank if its rank tuple is (in any element) less than what is maximally needed to describe every tensor of the same size. This value can be computed (Oseledets and Tyrtyshnikov, 2010).

Using the novel insights, on how the underlying structure of the LPV sub-Markov parameters can be exploited using tensor techniques, and the stated challenge for LPV identification the goal of this thesis can now be formulated.

1.5. GOAL OF THIS THESIS

In Section 1.3, it was argued that the development of LPV identification techniques is hampered by the 'curse-of-dimensionality'. Advances in this field would greatly benefit several applications, such as wind turbines and bio-mechanics. This will involve exploiting the underlying structure better. In the previous section, it was shown that this structure can be seen as (exact) multi-linear structure. This multi-linear structure, through the use of tensor techniques, allows breaking the 'curse-of-dimensionality'. Therefore, in this thesis our goal is as follows.

[8] It has to be remarked that this value is generally only known up to some bounds (Alexeev et al., 2011). Therefore the conservative, lower bound (Alexeev et al., 2011) will be used.

[9] In this thesis 'tensor networks' only refer to the ones of Oseledets (2011); Batselier et al. (2017); Chen et al. (2016).

1

Thesis goal: develop LPV identification techniques which are 'curse-of-dimensionality'-free in memory and computation and have improved variance by exploiting the tensor structure.

More specifically, in this thesis improved variance is defined as higher Variance Accounted For (VAF) than the LPV-PBSID$_{\text{opt}}$ method with regularization and kernels of van Wingerden and Verhaegen (2009) which is taken as the 'base-line' method.

The goal of this thesis can be achieved if the following research question is answered:

Research question: do exact, low-rank

- polyadic,
- multi-linear singular value,
- tensor network,

decompositions of the LPV sub-Markov parameters exist, and if so, what are they and how can they be exploited to obtain methods which are 'curse-of-dimensionality'-free in memory and computation and have improved variance?

Notice that this research question is focused on three tensor decomposition approaches which have received the most attention in literature as discussed in the previous section.

1.6. ORGANIZATION OF THIS THESIS

In this thesis, three novel tensor methods for LPV state-space identification are derived. Each method uses one of the three previously motivated tensor decomposition approaches in order. Since these algorithms are completely different from each other, they are presented in separate chapters which can be read independently. This does mean there is some overlap between the chapters. A concise overview of each chapter is provided at the start of each chapter in their abstract. The references are provided per chapter. The backbone of each chapter is based on one journal paper. There is a slight difference in notation between the chapters and for Chapter 3, an extended version of its paper is presented. This version provides the tools to perform the proposed method in 'curse-of-dimensionality'-free manner. A short outline of the chapters is given below:

- In Chapter 2, a novel refinement LPV state-space identification method based on polyadic decompositions is presented. It also contains a thorough introduction of moving from the matrix perspective on LPV identification to a tensor perspective and is therefore placed first.

  This chapter has been published in:

Gunes, B., van Wingerden, J.-W., and Verhaegen, M. Predictor-based tensor regression (PBTR) for LPV subspace identification. Automatica, 79:235 – 243, 2017a. ISSN 0005-1098

and is also based on:

Gunes, B., van Wingerden, J.-W., and Verhaegen, M. Tensor regression for LTI subspace identification. In American Control Conference (ACC), 2015, pages 1131–1136. IEEE, 2015a,

Gunes, B., van Wingerden, J.-W., and Verhaegen, M. Tensor regression for LTI subspace identification: free parametrizations. SYmposium on System IDentification, IFAC-PapersOnLine, 48(28):909–914, 2015b

Gunes, B., van Wingerden, J.-W., and Verhaegen, M. Tensor regression for LPV subspace identification. SYmposium on System IDentification, IFAC-PapersOnLine, 48 (28):421–426, 2015c

- In Chapter 3, a novel LPV subspace identification method based on the MLSVD is presented. This includes several results which are interesting to a wide system identification audience. Firstly, it is shown how to form and exploit (exactly) multilinear low-rank tensors from the (LPV sub-Markov) parameters. Secondly, for this problem tools are provided to perform the (regularized) optimization in 'curse-of-dimensionality'-free manner.

  This chapter is an extension [10] of:

  Gunes, B., van Wingerden, J.-W., and Verhaegen, M. Tensor nuclear norm LPV subspace identification. IEEE Transactions on Automatic Control, 2018.

- In Chapter 4, a novel refinement LPV state-space identification method based on tensor networks is presented. This chapter provides several contributions to mature the use of tensor techniques for LPV refinement methods, and can be seen as a successor to Gunes et al. (2017). The ranks of these tensor networks are exactly the number of states. This allows making educated guesses of the tensor network ranks. Additionally, tensor network optimization tools are well-understood in literature and have nice properties: they are well-posed for fixed ranks (even with incorrectly fixed ranks) and enjoy local linear convergence under mild conditions. Furthermore, the entire method is 'curse-of-dimensionality'-free in storage and computation.

  This chapter has been published in:

  Gunes, B., van Wingerden, J.-W., and Verhaegen, M. Tensor networks for MIMO LPV system identification. Submitted to International Journal of Control, 2017b

Finally in Chapter 5, the conclusions and recommendations are presented. A visualization of the thesis outline is presented in Fig. 1.6.

[10] This extension makes the method 'curse-of-dimensionality'-free in memory and computation.

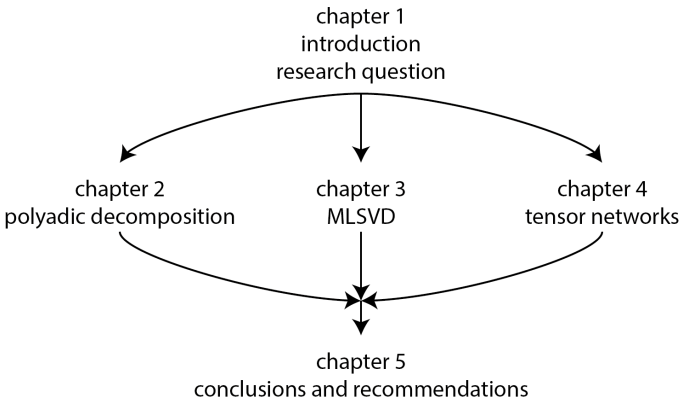


Figure 1.6: This figure shows an illustration of the thesis outline. In this chapter the research question and its sub-questions have been introduced. The next three chapters each tackle one sub-question. Finally, the sub-answers are combined in the final conclusion chapter.

BIBLIOGRAPHY

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Abbas, H. and Werner, H. An instrumental variable technique for open-loop and closed-loop identification of input-output LPV models. In *European Control Conference (ECC), 2009*, pages 2646–2651. IEEE, 2009.

Alexeev, B., Forbes, M. A., and Tsimerman, J. Tensor rank: Some lower and upper bounds. In *Computational Complexity (CCC), 2011 IEEE 26th Annual Conference on*, pages 283–291. IEEE, 2011.

Bachnas, A., Tóth, R., Mesbah, A., and Ludlage, J. Perspectives of data-driven LPV modeling of high-purity distillation columns. In *European Control Conference (ECC), 2013*, pages 3776–3783. IEEE, 2013.

Bamieh, B. and Giarre, L. Identification of linear parameter varying models. *International journal of robust and nonlinear control*, 12(9):841–853, 2002.

Batselier, K., Chen, Z., and Wong, N. Tensor network alternating linear scheme for MIMO volterra system identification. *Automatica*, 84:26–35, 2017.

Belforte, G., Dabbene, F., and Gay, P. LPV approximation of distributed parameter systems in environmental modelling. *Environmental Modelling & Software*, 20(8):1063–1070, 2005.

Bossanyi, E. A. The design of closed loop controllers for wind turbines. *Wind energy*, 3 (3):149–163, 2000.

Butcher, M., Karimi, A., and Longchamp, R. On the consistency of certain identification methods for linear parameter varying systems. *IFAC Proceedings Volumes*, 41(2):4018–4023, 2008.

Chen, Z., Batselier, K., Suykens, J. A., and Wong, N. Parallelized tensor train learning for polynomial pattern classification. *arXiv preprint arXiv:1612.06505*, 2016.

Chiuso, A. The role of vector autoregressive modeling in predictor-based subspace identification. *Automatica*, 43(6):1034–1048, 2007.

Chiuso, A. On the asymptotic properties of closed-loop CCA-type subspace algorithms: equivalence results and role of the future horizon. *IEEE Transactions on automatic control*, 55(3):634–649, 2010.

Cichocki, A., Zdunek, R., Phan, A. H., and Amari, S.-i. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

Cox, P. and Tóth, R. Alternative form of predictor based identification of LPV-SS models with innovation noise. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 1223–1228. IEEE, 2016a.

Cox, P. B. and Tóth, R. LPV state-space model identification in the bayesian setting: A 3-step procedure. In *American Control Conference (ACC), 2016*, pages 4604–4610. IEEE, 2016b.

Cox, P. B. and Tóth, R. On the connection between different noise structures for LPV-SS models. *arXiv preprint arXiv:1610.09173*, 2016c.

Darwish, M., Cox, P., Pillonetto, G., and Tóth, R. Bayesian identification of LPV box-jenkins models. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 66–71, Dec 2015.

De Caigny, J., Camino, J. F., and Swevers, J. Interpolating model identification for SISO linear parameter-varying systems. *Mechanical Systems and Signal Processing*, 23(8): 2395–2417, 2009.

De Lathauwer, L. A survey of tensor methods. In *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, pages 2773–2776. IEEE, 2009.

De Lathauwer, L. and Vandewalle, J. Dimensionality reduction in higher-order signal processing and rank-(r1, r2,..., rn) reduction in multilinear algebra. *Linear Algebra and its Applications*, 391:31–55, 2004.

De Lathauwer, L., De Moor, B., and Vandewalle, J. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

De Vos, M., Vergult, A., De Lathauwer, L., De Clercq, W., Van Huffel, S., Dupont, P., Palmini, A., and Van Paesschen, W. Canonical decomposition of ictal scalp EEG reliably detects the seizure onset zone. *NeuroImage*, 37(3):844–854, 2007.

Deprettere, E. F. *SVD and signal processing: algorithms, applications and architectures*. North-Holland Publishing Co., 1989.

Dong, K. and Wu, F. Robust and gain-scheduling control of LFT systems through duality and conjugate lyapunov functions. *International Journal of Control*, 80(4):555–568, 2007.

Favoreel, W., De Moor, B., and Van Overschee, P. Subspace identification of bilinear systems subject to white inputs. *Automatic Control, IEEE Transactions on*, 44(6):1157–1165, 1999.

Felici, F., Van Wingerden, J.-W., and Verhaegen, M. Subspace identification of MIMO LPV systems using a periodic scheduling sequence. *Automatica*, 43(10):1684–1697, 2007.

Gebraad, P. M., van Wingerden, J.-W., Fleming, P. A., and Wright, A. D. LPV subspace identification of the edgewise vibrational dynamics of a wind turbine rotor. In *Control Applications (CCA), 2011 IEEE International Conference on*, pages 37–42. IEEE, 2011a.

Gebraad, P. M., van Wingerden, J.-W., van der Veen, G. J., and Verhaegen, M. LPV subspace identification using a novel nuclear norm regularization method. In *American Control Conference (ACC)*, 2011b.

Gebraad, P. M., van Wingerden, J.-W., Fleming, P. A., and Wright, A. D. LPV identification of wind turbine rotor vibrational dynamics using periodic disturbance basis functions. *Control Systems Technology, IEEE Transactions on*, 21(4):1183–1190, 2013.

Golabi, A., Meskin, N., Tóth, R., and Mohammadpour, J. A bayesian approach for LPV model identification and its application to complex processes. *IEEE Transactions on Control Systems Technology*, 2017.

Goos, J., Lataire, J., Louarroudi, E., and Pintelon, R. Frequency domain weighted nonlinear least squares estimation of parameter-varying differential equations. *Automatica*, 75:191–199, 2017.

Gray, H. *Anatomy of the human body*. Lea & Febiger, 1918.

Gunes, B., van Wingerden, J.-W., and Verhaegen, M. Predictor-based tensor regression (PBTR) for LPV subspace identification. *Automatica*, 79:235 – 243, 2017. ISSN 0005-1098.

Hansen, M. H. Aeroelastic instability problems for wind turbines. *Wind Energy*, 10(6): 551–577, 2007.

Hjalmarsson, H., Gevers, M., and De Bruyne, F. For model-based control design, closed-loop identification gives better performance. *Automatica*, 32(12):1659–1673, 1996.

Khoromskij, B. N. Tensors-structured numerical methods in scientific computing: Survey on recent advances. *Chemometrics and Intelligent Laboratory Systems*, 110(1):1–19, 2012.

Knudsen, T. Consistency analysis of subspace identification methods based on a linear regression approach. *Automatica*, 37(1):81–89, 2001.

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

Larimore, W. E., Cox, P. B., and Tóth, R. CVA identification of nonlinear systems with LPV state-space models of affine dependence. In *2015 American Control Conference (ACC)*, pages 831–837, July 2015.

Larimore, W. E. and Buchholz, M. ADAPT-LPV software for identification of nonlinear parameter-varying systems. *IFAC Proceedings Volumes*, 45(16):1820–1825, 2012.

Laurain, V., Gilson, M., Tóth, R., and Garnier, H. Refined instrumental variable methods for identification of LPV box-jenkins models. *Automatica*, 46(6):959–967, 2010.

Lee, L. H. and Poolla, K. Identification of linear parameter-varying systems using nonlinear programming. *Journal of dynamic systems, measurement, and control*, 121(1): 71–78, 1999.

Leith, D. J. and Leithead, W. E. Survey of gain-scheduling analysis and design. *International journal of control*, 73(11):1001–1025, 2000.

Liu, Q., Mohammadpour, J., Tóth, R., and Meskin, N. Non-parametric identification of linear parameter-varying spatially-interconnected systems using an LS-SVM approach. In *2016 American Control Conference (ACC)*, pages 4592–4597, July 2016.

Ljung, L. *System identification (2nd ed.): theory for the user*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999. ISBN 0-13-656695-2.

Mercere, G., Lovera, M., and Laroche, E. Identification of a flexible robot manipulator using a linear parameter-varying descriptor state-space structure. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 818–823. IEEE, 2011.

Mohammadpour, J. and Scherer, C. W. *Control of linear parameter varying systems with applications*. Springer Science & Business Media, 2012.

Moonen, M. and De Moor, B. *SVD and Signal Processing, III: Algorithms, Architectures and Applications*. Elsevier, 1995.

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

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

Papy, J.-M., De Lathauwer, L., and Van Huffel, S. Exponential data fitting using multilinear algebra: The decimative case. *Journal of Chemometrics*, 23(7-8):341–351, 2009.

Petersson, D. and Löfberg, J. Optimisation-based modelling of LPV systems using an objective. *International Journal of Control*, 87(8):1536–1548, 2014.

Petres, Z. Polytopic decomposition of linear parameter-varying models by tensor-product model transformation. 2006.

Rugh, W. J. and Shamma, J. S. Research on gain scheduling. *Automatica*, 36(10):1401–1425, 2000.

Scherer, C. W. LPV control and full block multipliers. *Automatica*, 37(3):361–375, 2001.

Schollwöck, U. The density-matrix renormalization group. *Reviews of modern physics*, 77(1):259, 2005.

Sename, O., Gaspar, P., and Bokor, J. *Robust control and linear parameter varying approaches: application to vehicle dynamics*, volume 437. Springer, 2013.

Shamma, J. S. An overview of LPV systems. In *Control of linear parameter varying systems with applications*, pages 3–26. Springer, 2012.

Signoretto, M., De Lathauwer, L., and Suykens, J. A. Nuclear norms for tensors and their use for convex multilinear estimation. *Submitted to Linear Algebra and Its Applications*, 43, 2010.

Skogestad, S. and Postlethwaite, I. *Multivariable feedback control: analysis and design*, volume 2. Wiley New York, 2007.

Smilde, A., Bro, R., and Geladi, P. *Multi-way analysis: applications in the chemical sciences*. John Wiley & Sons, 2005.

Sorber, L., Van Barel, M., and De Lathauwer, L. Structured data fusion. *IEEE Journal of Selected Topics in Signal Processing*, 9(4):586–600, 2015.

Tóth, R. *Modeling and identification of linear parameter-varying systems*, volume 403. Springer, 2010.

Tóth, R., Felici, F., Heuberger, P., and Van den Hof, P. Crucial aspects of zero-order hold LPV state-space system discretization. *IFAC Proceedings Volumes*, 41(2):4952–4957, 2008.

Tóth, R., van de Wal, M., Heuberger, P. S., and Van den Hof, P. M. LPV identification of high performance positioning devices. In *American Control Conference (ACC), 2011*, pages 151–158. IEEE, 2011.

Tóth, R., Abbas, H. S., and Werner, H. On the state-space realization of LPV input-output models: Practical approaches. *Control Systems Technology, IEEE Transactions on*, 20 (1):139–153, 2012.

Vaccaro, R. J. *SVD and Signal Processing II: Algorithms, analysis and applications*. Elsevier Science Inc., 1991.

van der Maas, R., van der Maas, A., and Oomen, T. Accurate frequency response function identification of LPV systems: A 2D local parametric modeling approach. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pages 1465–1470. IEEE, 2015.

van der Veen, G., van Wingerden, J.-W., Bergamasco, M., Lovera, M., and Verhaegen, M. Closed-loop subspace identification methods: an overview. *IET Control Theory & Applications*, 7(10):1339–1358, July 2013. ISSN 1751-8652.

van Eesbeek, S., van der Helm, F., Verhaegen, M., and de Vlugt, E. LPV subspace identification of time-variant joint impedance. In *2013 6th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 343–346, Nov 2013.

Van Kuik, G. and Peinke, J. *Long-term Research Challenges in Wind Energy-A Research Agenda by the European Academy of Wind Energy*, volume 6. Springer, 2016.

van Wingerden, J.-W. and Verhaegen, M. Subspace identification of bilinear and LPV systems for open- and closed-loop data. *Automatica*, 45(2):372 – 381, 2009. ISSN 0005-1098.

van Wingerden, J.-W., Felici, F., and Verhaegen, M. Subspace identification of MIMO LPV systems using a piecewise constant scheduling sequence with hard/soft switching. In *European Control Conference (ECC), 2007*, pages 927–934. IEEE, 2007.

Vasilescu, M. A. O. and Terzopoulos, D. Multilinear analysis of image ensembles: Tensorfaces. In *Computer Vision ECCV 2002*, pages 447–460. Springer, 2002.

Verdult, V. and Verhaegen, M. Subspace identification of multivariable linear parameter-varying systems. *Automatica*, 38(5):805 – 814, 2002. ISSN 0005-1098.

Verdult, V., Bergboer, N., and Verhaegen, M. Identification of fully parameterized linear and nonlinear state-space systems by projected gradient search. In *Proceedings of the 13th IFAC Symposium on System Identification, Rotterdam*, 2003.

Vervliet, N. and De Lathauwer, L. A randomized block sampling approach to canonical polyadic decomposition of large-scale tensors. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):284–295, 2016.

Vervliet, N., Debals, O., Sorber, L., and De Lathauwer, L. Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis. *IEEE Signal Processing Magazine*, 31(5):71–79, 2014.

Vizer, D., Mercere, G., Prot, O., Laroche, E., and Lovera, M. Linear fractional LPV model identification from local experiments: an H_∞-based optimization technique. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 4559–4564. IEEE, 2013.

Zavari, K., Pipeleers, G., and Swevers, J. Gain-scheduled controller design: illustration on an overhead crane. *IEEE Transactions on Industrial Electronics*, 61(7):3713–3718, 2014.

Zhou, H., Li, L., and Zhu, H. Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association*, 108(502):540–552, 2013.

2

PREDICTOR-BASED TENSOR REGRESSION (PBTR) FOR LPV SUBSPACE IDENTIFICATION

The major bottleneck in state-of-the-art Linear Parameter Varying (LPV) subspace methods is the curse-of-dimensionality during the first regression step. In this paper, the origin of the curse-of-dimensionality is pinpointed and subsequently a novel method is proposed which does not suffer from this bottleneck. The problem is related to the LPV sub-Markov parameters. These have inherent structure and are dependent on each other. But state-of-the-art LPV subspace methods parametrize the LPV sub-Markov parameters independently. This means the inherent structure is not preserved in the parametrization. In turn this leads to a superfluous parametrization with the curse-of-dimensionality. The solution lies in using parametrizations which preserve the inherent structure sufficiently to avoid the curse-of-dimensionality. In this paper a novel method based on tensor regression is proposed. This novel method is named the Predictor-Based Tensor Regression method (PBTR). This method preserves the inherent structure sufficiently to avoid the curse-of-dimensionality. Simulation results show that PBTR has superior performance with respect to both state-of-the-art LPV subspace techniques and also non-convex techniques.

2.1. INTRODUCTION

Identification problems can be seen as inverse problems. Given some observations and a model structure, they try to infer the values of the parameters characterizing the system. Better results can be obtained both by better observations and richer model structures. One way to obtain richer model structures, is to incorporate more structure of the underlying problem. For some systems this can be achieved by starting to use Linear Parameter Varying (LPV) model structures (Giarré et al., 2006). In this paper, we develop novel methods for Linear Parameter Varying (LPV) subspace identification.

This chapter is a reprint of the paper (Gunes et al., 2017) published in Automatica.

LPV systems are a very useful subclass of non-linear systems. They are time-varying systems, but their dependence on time is strictly through a scheduling sequence. This system description is very useful for applications for which the scheduling sequence is known. Some application examples are wind turbines (Bianchi et al., 2005; Gebraad et al., 2013), aircraft applications (Balas, 2002), batteries (Remmlinger et al., 2013) and compressors (Giarré et al., 2006). Unlike descriptions of systems that are completely non-linear, there are control methodologies available for LPV systems which can guarantee stability and performance in the face of uncertainties (Scherer, 2001). These control methodologies of course require models of the system, which can be obtained from first principles approaches or from identification.

Our focus is on the development of novel LPV identification methods. More specifically, we allow for arbitrarily varying scheduling sequence. This class of systems also encompasses bilinear systems, where the scheduling sequence equals the inputs. Models can be obtained from experimental data by using identification methods. LPV Identification can be divided into global and local approaches. Global approaches perform only one identification experiment, while local approaches perform several experiments at fixed scheduling parameters and then interpolate. Therefore, they perform differently depending on the application (De Caigny et al., 2009) , (Lovera and Mercere, 2007) , (Shamma, 2012). In this paper only global approaches will be discussed. There are two major approaches to (global) LPV identification: the subspace approach and the Prediction Error (PE) approach. Both have received considerable attention in literature (Bamieh and Giarre, 2002; Tóth et al., 2012; van Wingerden and Verhaegen, 2009). The advantage of subspace methods is that they produce state-space models which can be directly used by the mainstream LPV control design methodologies (Scherer, 2001). This is advantageous, because transforming between input-output and state-space models in the LPV setting is non-trivial (Tóth et al., 2012). Another advantage is that subspace methods can extend naturally to Multiple Input Multiple Output (MIMO) and closed-loop systems. But they also have a major disadvantage: they suffer from the curse-of-dimensionality and yield unwieldy parameter counts (van Wingerden and Verhaegen, 2009). There are several solutions proposed in literature. Some solutions are based on regularization, such as Tikhonov or Nuclear Norm regularization (van Wingerden and Verhaegen, 2009; Gebraad et al., 2011). Some other solutions are tailored towards scheduling sequences which are periodic (Felici et al., 2007), white noise (Favoreel et al., 1999) or piecewise constant (van Wingerden et al., 2007). However these solutions either only partially alleviate this bottleneck or only work for specific cases.

In this paper the origin of the curse-of-dimensionality of LPV subspace methods is pinpointed. It is shown that the curse-of-dimensionality appears when structure of the LPV sub-Markov parameters is not preserved in the parametrization. More specifically, not all the structure of the LPV sub-Markov parameters need to be preserved. This will be made clearly visible by reformulating the LPV data equation using tensors. Such a reformulation will be presented using the inner product of a structured LPV sub-Markov parameter tensor and a corresponding data tensor. Based on this insight, a novel method based on *tensor regression* (Zhou et al., 2013; Lu et al., 2011; Guo et al., 2012) is proposed. Tensor regression is generally used in order to deal with curse-of-dimensionality, such as Magnetic Resonance Imaging (MRI) data (Zhou et al., 2013). Tensor arise naturally in

several more applications such as facial recognition (Vasilescu and Terzopoulos, 2002) and gait recognition (Lu et al., 2008), and preserving that structure can be highly beneficial (Signoretto et al., 2010). The novel method preserves structure just sufficiently to avoid the curse-of-dimensionality. This method is named the Predictor-Based Tensor Regression method (PBTR). Simulation results show that this method has higher performance than both state-of-the-art (LPV subspace) methods and also other non-convex methods in the sense of variance accounted for.

In previous work, we presented variants of PBTR for both for LTI (Gunes et al., 2015a,b) and LPV (Gunes et al., 2015c) systems. The novel LPV parametrization presented here, has been presented before in Gunes et al. (2015c) using unnecessarily complicated matrices. In this paper we now present everything explicitly using tensor forms in order to greatly improve clarity on the true system, the parametrizations of several methods and the cause of the curse-of-dimensionality. These tensor forms are the inherent form for tensor regression. This also allows for a clear discussion of the design choices made with PBTR. Furthermore, we compare PBTR with existing non-convex methods, and present simulation results which show that PBTR can outperforms them in terms of variance. We do remark that the formal proof to generalize these simulation results remains an open issue. This paper provides a complete and concise investigation of tensor regression for LPV subspace identification.

The outline of this paper is as follows. The basics of LPV subspace identification are discussed in the next section. Afterwards in Section 2.3, PBTR is presented together with its motivations. Simulations results are presented in Section 2.4. Finally the conclusions are presented.

2.2. LPV SUBSPACE IDENTIFICATION

In this section, LPV subspace identification is reviewed. The focus will be on the work of van Wingerden and Verhaegen (2009) and Verdult et al. (2003).

An LPV system can be described by a discrete LPV state-space equation:

$$x_{k+1} = A(\mu_k)x_k + B(\mu_k)u_k + w_k \tag{2.1a}$$

$$y_k = C(\mu_k)x_k + D(\mu_k)u_k + v_k, \tag{2.1b}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^r$ and $y \in \mathbb{R}^l$ are the state, input and output vector variables. This description takes into account both process noise w and measurement noise v. The subscript k indicates the sample number. The matrices A, B, C and D are the appropriately dimensioned state-space matrices. The scheduling sequence μ_k is time-varying and affects the state-space matrices. We assume that the relation of the scheduling sequence to the state-space matrices is affine:

$$A(\mu_k) = \sum_{i=1}^{m} \mu_k^{(i)} A^{(i)}, \tag{2.2}$$

and similarly for the other state-space matrices. The scalar $\mu_k^{(i)}$ is defined as the k-th sample of the i-th scheduling parameter, and $\mu_k^{(1)} = 1$. Additionally we assume in this paper that:

Assumption 2.2.1 *The scheduling sequence μ_k is known.*

For presentation reasons we also restrict ourselves to LPV systems whose output equations are independent of the scheduling sequence. We also omit D for presentation purposes. The extension to include D is straight-forward.

For identification purposes, the innovation representation is commonly used (Ljung, 1999). This representation uses the innovation term e to describe the system. For our LPV system, the resulting expression becomes:

$$x_{k+1} = \sum_{i=1}^{m} \mu_k^{(i)} \left(A^{(i)} x_k + B^{(i)} u_k + K^{(i)} e_k \right) \tag{2.3a}$$

$$y_k = C x_k + e_k \tag{2.3b}$$

This representation uses the properties of the Kalman filter.

If the system is closed-loop, then the inputs and noise are correlated due to the feedback. This causes open-loop identification methods to produce biased estimates. The state-of-the-art (LPV subspace) method presented in this section is a closed-loop method, and deals with the correlation between the input and noise by using a predictor-based representation of (2.3):

$$\hat{x}_{k+1} = \sum_{i=1}^{m} \mu_k^{(i)} \left(\tilde{A}^{(i)} x_k + \bar{B}^{(i)} \begin{bmatrix} u_k \\ y_k \end{bmatrix} \right) \tag{2.4a}$$

$$y_k = C x_k + e_k, \tag{2.4b}$$

where $\tilde{A}^{(i)} = A^{(i)} - K^{(i)} C$ and $\bar{B}^{(i)} = [B^{(i)}, K^{(i)}]$. This representation has two nice properties. First, notice that now the innovation only appears at the output equation. Second, the equations now describe the observer error states and use the corresponding observer error dynamics $\tilde{A}^{(i)} = A^{(i)} - K^{(i)} C$. The observer error dynamics can be assumed to be uniformly exponentially stable (van Wingerden and Verhaegen, 2009; Chiuso, 2007; Jansson, 2005), and hence the influence of the states at time k will decay with time. This can be exploited. It appears that if the LPV description (2.4) is (uniformly) exponentially stable, then it can be approximated arbitrarily well under the assumption that the effect of an initial state is exactly zero after some p time steps (Knudsen, 2001). In other words: the current state can be arbitrarily well approximated by using the p past inputs and outputs without any (initial) states:

$$\hat{x}_{k+p} \approx \mathcal{K} \mathcal{Z}_{k+p}, \tag{2.5}$$

where $\mathcal{K}$ contains the (LPV sub-Markov) parameters and $\mathcal{Z}_k$ the effective (past input and output) data. In this factorization (van Wingerden and Verhaegen, 2009), the scheduling sequence is absorbed into $\mathcal{Z}$ and $\mathcal{K}$ is independent of the scheduling sequence. These two matrices will be defined explicitly later in this section. The output equation follows directly:

$$y_{k+p} \approx C \mathcal{K} \mathcal{Z}_{k+p} + e_{k+p}, \tag{2.6}$$

which is very useful for the first identification step because it directly allows for linear regression. Next, for completeness we present the parameter matrix $C\mathcal{K}$ and effective data matrix $\mathcal{Z}_k$.

The matrix $C\mathcal{K}$ contains the sub-Markov parameters and is independent of the scheduling sequence. Recall that the scheduling sequence is absorbed into $\mathcal{Z}_k$. The matrix $C\mathcal{K}$ is a function of the predictor-based state-space matrices:

$$C\mathcal{K} = [\mathcal{L}_p, \ldots, \mathcal{L}_1], \tag{2.7}$$

where $\mathcal{L}_j$ contains all possible routes from inputs to outputs of length $(j+1)$:

$$\mathcal{C}_1 = [\bar{B}^{(1)}, \ldots, \bar{B}^{(m)}] \tag{2.8a}$$

$$\mathcal{C}_2 = [\tilde{A}^{(1)}\mathcal{C}_1, \ldots, \tilde{A}^{(m)}\mathcal{C}_1] \tag{2.8b}$$

$$\mathcal{C}_{i+1} = [\tilde{A}^{(1)}\mathcal{C}_i, \ldots, \tilde{A}^{(m)}\mathcal{C}_i] \tag{2.8c}$$

$$\mathcal{L}_j = C\mathcal{C}_j \tag{2.8d}$$

Notice that this definition is slightly different from van Wingerden and Verhaegen (2009) in the sense that we absorb the C matrix into $\mathcal{L}$.

During the first regression step of predictor-based methods, the matrix $C\mathcal{K}$ has to be estimated. But unlike in the LTI case, $C\mathcal{K}$ now has a very large number of elements:

$$q = l(l+r)\sum_{j=1}^{p} m^j \tag{2.9}$$

This creates a problem for linear regression because linear regression uses as much parameters as there are elements in $C\mathcal{K}$, namely q. More specifically, state-of-the-art LPV subspace methods suffer from the *curse-of-dimensionality*:

Definition 2.2.1 *Identification methods suffer from the curse-of-dimensionality if their number of parameters scales exponentially with the past window p.*

The main contribution of this paper is a novel method which does not suffer from the curse-of-dimensionality and has good numeric properties.

The effective data matrix $\mathcal{Z}_k$ is:

$$\mathcal{Z}_k = N_{k-p}^{p} Z_k, \tag{2.10}$$

where N_{k-p}^{p} contains the scheduling sequence and Z_k the input-output data relevant to y_k. The matrix N_k^p is:

$$N_k^p = \begin{bmatrix} P_{p|k} & 0 & \cdots & 0 \\ 0 & P_{p-1|k+1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & P_{1|k+p-1} \end{bmatrix}, \tag{2.11a}$$

$$P_{p|k} = \mu_{k+p-1} \otimes \cdots \otimes \mu_k \otimes I_{r+l}, \tag{2.11b}$$

where $\otimes$ is defined as the Kronecker product (Brewer, 1978). The matrix Z_k is:

$$Z_k = \begin{bmatrix} z_{k-p} \\ \vdots \\ z_{k-1} \end{bmatrix}, z_k = \begin{bmatrix} u_k \\ y_k \end{bmatrix} \tag{2.12}$$

The general estimation procedure is as follows. First, the matrix $C\mathcal{K}$ is estimated using the data-equation (2.6). Afterwards the estimate of $C\mathcal{K}$ is used in order to choose a model order and obtain an estimate of the state sequence. Together with that estimate, the state-space matrices can be readily estimated (van Wingerden and Verhaegen, 2009).

The state-of-the-art (LPV subspace) method (van Wingerden and Verhaegen, 2009) follows these same steps, but additionally deploys a dual (or kernel) approach with Tikhonov regularization in the first step. Its regularization parameter is chosen using Generalized Cross Validation (Golub et al., 1979). This reduces computational complexity and improves the quality of the estimate in most cases.

Alternatively, non-convex methods exist such as Lee and Poolla (1999) and Verdult et al. (2003). These methods directly parametrize the state-space matrices (2.3). The resulting parametrizations are polynomial and very sensitive to local minima. In order to somewhat ease this issue, most of these methods assume $K = 0$. In this paper PBTR is compared among others with the output-error method of Verdult et al. (2003).

In the next section, the novel representation and PBTR are presented.

2.3. PREDICTOR-BASED TENSOR REGRESSION

The novel method, Predictor-based Tensor Regression, is presented in this section. First some general (tensor regression) expressions are presented. Afterwards we show that the LPV subspace identification problem contains structure that can be exploited. This can be done using tensor regression in order to avoid the curse-of-dimensionality. We show how the LPV sub-Markov parameters indeed form a parameter tensor. Then the parametrization of PBTR and its algorithm are presented.

2.3.1. GENERAL TENSOR REGRESSION EXPRESSIONS

We first present some general (tensor regression) expressions.

Define $[M]_{i,j}$ as the entry of M at row i and column j. Let $[M]_{:,j}$ and $[M]_{i,:}$ respectively be a column and row vector. For a two-by-two matrix M:

$$M = \begin{bmatrix} [M]_{1,1} & [M]_{1,2} \\ [M]_{2,1} & [M]_{2,2} \end{bmatrix} = \begin{bmatrix} [M]_{1,:} \\ [M]_{2,:} \end{bmatrix} = \begin{bmatrix} [M]_{:,1} & [M]_{:,2} \end{bmatrix} \tag{2.13}$$

For both row and column vectors, define $[v]_i$ as the i-th entry of v.

We define an operator to form tensors from a set of vectors, like in Zhou et al. (2013). Let β_d represent a vector with size d_i-by-1. Then the outer product $\beta_1 \circ \beta_2 \circ \cdots \circ \beta_D$ is a tensor of size $\mathbb{R}^{d_1,d_2,\ldots,d_D}$ with entries:

$$[\beta_1 \circ \cdots \circ \beta_D]_{i_1,\ldots,i_D} = \prod_{d=1}^{D} [\beta_d]_{i_d} \tag{2.14}$$

A tensor can be represented in several forms. For the use of tensor regression, the most natural form is the rank-R decomposition (Zhou et al., 2013) (or CANonical DEComposition/PARAllel FACtors in psychometrics (Kolda and Bader, 2009)). This decomposition decomposes a tensor into the sum of exactly R outer products. For example consider a tensor $\mathcal{T}$ with D dimensions. This tensor can then be rank-R decomposed

into:

$$\mathcal{T} = \sum_{r=1}^{R} \beta_1^{(r)} \circ \beta_2^{(r)} \circ \cdots \circ \beta_D^{(r)}, \tag{2.15}$$

where the $\beta_*^{(*)}$ represent vectors and $\circ$ is the outer product (Zhou et al., 2013) which turns vectors into a tensor. The subscript indicates the vector group, and the superscript indicates which individual vector to take. There are in total R times D vectors.

In the succeeding subsections, the structure of the LPV subspace problem and PBTR are presented explicitly and in tensor form.

2.3.2. THE HIGHLY-STRUCTURED PARAMETER TENSOR

In this subsection we use the LPV sub-Markov parameters to build a highly-structured parameter tensor. We present this tensor in rank-R decomposition form, such that tensor regression can be directly applied. This will allow for a clear view on the parametrization of PBTR and why it is sufficient to avoid the curse-of-dimensionality.

We present the structure of the LPV sub-Markov parameters explicitly in tensor form. For this purpose, consider the LPV sub-Markov parameters:

$$C\mathcal{K} = [\mathcal{L}_p, \mathcal{L}_{p-1}, \ldots, \mathcal{L}_1] \tag{2.16}$$

For presentation purposes, consider the part:

$$\mathcal{L}_2 = C[\tilde{A}^{(1)}\bar{B}^{(1)}, \tilde{A}^{(1)}\bar{B}^{(2)}, \tilde{A}^{(2)}\bar{B}^{(1)}, \cdots \tilde{A}^{(m)}\bar{B}^{(m)}]$$

Notice that there is structure present. This structure becomes more apparent if we reorganize the parameters to:

$$\tilde{\mathcal{L}}_2 = \begin{bmatrix} C\tilde{A}^{(1)}\bar{B}^{(1)} & C\tilde{A}^{(1)}\bar{B}^{(2)} & \ldots & C\tilde{A}^{(1)}\bar{B}^{(m)} \\ \vdots & \vdots & \ddots & \vdots \\ C\tilde{A}^{(m)}\bar{B}^{(1)} & C\tilde{A}^{(m)}\bar{B}^{(2)} & \ldots & C\tilde{A}^{(m)}\bar{B}^{(m)} \end{bmatrix},$$

or even more clear:

$$\tilde{\mathcal{L}}_2 = \begin{bmatrix} C\tilde{A}^{(1)} \\ C\tilde{A}^{(2)} \\ \vdots \\ C\tilde{A}^{(m)} \end{bmatrix} [\bar{B}^{(1)}, \bar{B}^{(2)}, \ldots, \bar{B}^{(m)}], \tag{2.17}$$

where the reorganization changed the size from l-by-$(l+r)m^2$ to lm-by-$(l+r)m$.

However, this description using matrix products is not directly suitable for tensor regression. For that reason we now move towards a rank-R decomposition form, which is a sum of outer products of vectors. First we solve this problem for one output C_1 and one effective input $\bar{B}_1$. Equation (2.17) then becomes:

$$\tilde{\mathcal{L}}_2^{1,1} = \begin{bmatrix} C_1\tilde{A}^{(1)} \\ C_1\tilde{A}^{(2)} \\ \vdots \\ C_1\tilde{A}^{(m)} \end{bmatrix} [\bar{B}_1^{(1)}, \ldots, \bar{B}_1^{(m)}], \tag{2.18}$$

Then we can rewrite the equation above by splitting out every single summation inside the matrix multiplications:

$$\bar{\mathcal{L}}_2^{1,1} = \sum_{i=1}^{n}\sum_{j=1}^{n} \begin{bmatrix} [C_1]_i[\tilde{A}^{(1)}]_{i,j} \\ [C_1]_i[\tilde{A}^{(2)}]_{i,j} \\ \vdots \\ [C_1]_i[\tilde{A}^{(m)}]_{i,j} \end{bmatrix} [[\bar{B}_1^{(1)}]_j, \ldots, [\bar{B}_1^{(m)}]_j],$$

or using the fact that $[C_1]_i$ is scalar:

$$\bar{\mathcal{L}}_2^{1,1} = \sum_{i=1}^{n}\sum_{j=1}^{n} [C_1]_i \begin{bmatrix} [\tilde{A}^{(1)}]_{i,j} \\ [\tilde{A}^{(2)}]_{i,j} \\ \vdots \\ [\tilde{A}^{(m)}]_{i,j} \end{bmatrix} [[\bar{B}_1^{(1)}]_j, \ldots, [\bar{B}_1^{(m)}]_j]$$

Now there are just products of vectors instead of products of matrices. Hence, we can set up the following rank-R decomposition:

$$\bar{\mathcal{L}}_2^{1,1} = \sum_{i=1}^{n}\sum_{j=1}^{n} v_1^{(i)} \circ v_2^{(i,j)} \circ v_3^{(j)} \tag{2.21}$$

where we have:

$$\tilde{v}_1^{(i)} = [C]_{1,i}, v_2^{(i,j)} = \begin{bmatrix} [\tilde{A}^{(1)}]_{i,j} \\ \vdots \\ [\tilde{A}^{(m)}]_{i,j} \end{bmatrix}, \tilde{v}_3^{(j)} = \begin{bmatrix} [\bar{B}_1^{(1)}]_j \\ \vdots \\ [\bar{B}_1^{(m)}]_j \end{bmatrix},$$

where tildes have been used to indicate definitions which are only valid for the single output and single effective input case (2.18).

Now we can extend to the case of a full C matrix. This is done by redefining $v_1^{(i)}$. While previously it was a scalar, we now turn it into a vector. The resulting outer product becomes three-dimensional. Simply define:

$$v_1^{(i)} = [C]_{:,i} \tag{2.22}$$

Now we can extend to the case of a full $\bar{B}$ matrix as well. This is more involved, because we already had a vector group devoted to $\bar{B}_1$. The solution is to devote two vector groups to $\bar{B}$. First, we define v_3 to incorporate the entire $\bar{B}$ matrix. This requires another superscript index $\bar{\kappa}$:

$$v_3^{(j,\bar{\kappa})} = \begin{bmatrix} [\bar{B}^{(1)}]_{j,\bar{\kappa}} \\ \vdots \\ [\bar{B}^{(m)}]_{j,\bar{\kappa}} \end{bmatrix} \tag{2.23}$$

Notice that previously $\bar{\kappa}$ was fixed at one. Also notice that j cycles the states, and $\bar{\kappa}$ cycles the width of $\bar{B}^{(\#)}$. We want to map this added complexity on a new dimension. For that purpose, we define v_4 as:

$$v_4^{(\bar{\kappa})} = \begin{bmatrix} \bar{0}_{\bar{\kappa}-1} \\ 1 \\ \bar{0}_{(l+r)-\bar{\kappa}} \end{bmatrix} \tag{2.24}$$

If we then use:

$$\bar{\mathcal{L}}_2 = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{\bar{\kappa}=1}^{l+r} v_1^{(i)} \circ v_2^{(i,j)} \circ v_3^{(j,\bar{\kappa})} \circ v_4^{(\bar{\kappa})}, \tag{2.25}$$

we obtain a $\mathbb{R}^{l,m,m,l+r}$ tensor where the last dimension cycles over the width of $\bar{B}^{(\#)}$. That is, $[\bar{\mathcal{L}}_2]_{:,:,:,i}$ corresponds to the i-th column of all $\bar{B}^{(\#)}$. Notice that $\bar{\mathcal{L}}_2$ and $\mathcal{L}_2$ have the same entries, but in different positions. Basically, we reorganized $\mathcal{L}_2$ into $\bar{\mathcal{L}}_2$ in order to make it suitable for rank-R decomposition and tensor regression.

The resulting expression (2.25) is valid for any LPV system, but describes only the LPV sub-Markov parameters in $\mathcal{L}_2$. Hence, we need to extend this formulation to capture all the LPV sub-Markov parameters. Before searching for an expression containing all the LPV sub-Markov parameters, we first investigate an expression containing the LPV sub-Markov parameters of $\mathcal{L}_p$. Its added complexity is the appearance of multiple $\tilde{A}$ within every product. We accommodate this complexity by using multiple v_2, one for each appearance of $\tilde{A}$ within every product:

$$\begin{aligned}
\bar{\mathcal{L}}_2 &= \sum_{i,j}^{n}\sum_{\bar{\kappa}=1}^{l+r} v_1^{(i)} \circ v_2^{(i,j)} \circ v_3^{(j,\bar{\kappa})} \circ v_4^{(\bar{\kappa})} \\
\bar{\mathcal{L}}_p &= \sum_{i,j,\delta_1,\ldots,\delta_{p-2}}^{n}\sum_{\bar{\kappa}=1}^{l+r} v_1^{(i)} \circ v_2^{(i,\delta_1)} \circ v_2^{(\delta_1,\delta_2)} \circ \ldots \\
&\quad \circ v_2^{(\delta_{p-2},j)} \circ v_3^{(j,\bar{\kappa})} \circ v_4^{(\bar{\kappa})},
\end{aligned}$$

where $\bar{\mathcal{L}}_p$ is a $\mathbb{R}^{l,m,m,\ldots,m,l+r}$ tensor. Notice that the superscripts continue to form a chain link (i,δ_1), (δ_1,δ_2) etc. This is a result of the underlying matrix multiplications, as was explained for (2.20).

The following step is to define a single tensor which contains all the LPV sub-Markov parameters. This requires the stacking of all $\bar{\mathcal{L}}_*$. These will be stacked over a new dimension, which runs from 1 to p and has index $\bar{p}$. So $\bar{\mathcal{L}}_1$ will be at the first index and so forth. However, the tensors $\bar{\mathcal{L}}_*$ do not have equal size. The solution is to make all these tensors the same size. For this purpose, consider the first product of the first $\mathcal{L}_*$:

$$\mathcal{L}_1 = [C\bar{B}^{(1)}, \ldots] \tag{2.27a}$$
$$\mathcal{L}_2 = [C\tilde{A}^{(1)}\bar{B}^{(1)}, \ldots] \tag{2.27b}$$
$$\mathcal{L}_3 = [C\tilde{A}^{(1)}\tilde{A}^{(1)}\bar{B}^{(1)}, \ldots] \tag{2.27c}$$
$$\vdots$$

The dimension mismatch of different $\bar{\mathcal{L}}_\#$ appears because they have a different number of terms in their products. This can be easily solved by adding identity matrices:

$$\mathcal{L}_1 = [CI_nI_n\bar{B}^{(1)}, \ldots] \tag{2.28a}$$
$$\mathcal{L}_2 = [C\tilde{A}^{(1)}I_n\bar{B}^{(1)}, \ldots] \tag{2.28b}$$
$$\mathcal{L}_3 = [C\tilde{A}^{(1)}\tilde{A}^{(1)}\bar{B}^{(1)}, \ldots] \tag{2.28c}$$
$$\vdots$$

Now every $\mathcal{L}_*$ has the same number of terms inside their products, and we can redefine:

Definition 2.3.1 *The tensors $\bar{\mathcal{L}}_j$*
Redefine the tensors $\bar{\mathcal{L}}_j$ as:

$$\begin{aligned}
\bar{\mathcal{L}}_p &= \sum_{i,j,\delta_1,\ldots,\delta_{p-2}}^{n} \sum_{\bar{\kappa}=1}^{l+r} v_1^{(i)} \circ v_2^{(i,\delta_1)} \circ v_2^{(\delta_1,\delta_2)} \circ \ldots \\
&\quad \circ v_2^{(\delta_{p-2},j)} \circ v_3^{(j,\bar{\kappa})} \circ v_4^{(\bar{\kappa})} \\
\bar{\mathcal{L}}_{p-1} &= \sum_{i,j,\delta_1,\ldots,\delta_{p-2}}^{n} \sum_{\bar{\kappa}=1}^{l+r} v_1^{(i)} \circ v_2^{(i,\delta_1)} \circ v_2^{(\delta_1,\delta_2)} \circ \ldots \\
&\quad \circ [I]_{\delta_{p-2},j} \bar{1}_m \circ v_3^{(j,\bar{\kappa})} \circ v_4^{(\bar{\kappa})} \\
&\quad \vdots \\
\bar{\mathcal{L}}_2 &= \sum_{i,j,\delta_1,\ldots,\delta_{p-2}}^{n} \sum_{\bar{\kappa}=1}^{l+r} v_1^{(i)} \circ v_2^{(i,\delta_1)} \circ [I]_{\delta_1,\delta_1} \bar{1}_m \circ \ldots \\
&\quad \circ [I]_{\delta_{p-2},j} \bar{1}_m \circ v_3^{(j,\bar{\kappa})} \circ v_4^{(\bar{\kappa})} \\
\bar{\mathcal{L}}_1 &= \sum_{i,j,\delta_1,\ldots,\delta_{p-2}}^{n} \sum_{\bar{\kappa}=1}^{l+r} v_1^{(i)} \circ [I]_{i,\delta_1} \bar{1}_m \circ [I]_{\delta_1,\delta_2} \bar{1}_m \circ \ldots \\
&\quad \circ [I]_{\delta_{p-2},j} \bar{1}_m \circ v_3^{(j,\bar{\kappa})} \circ v_4^{(\bar{\kappa})},
\end{aligned}$$

such that all $\bar{\mathcal{L}}_i$ have the same dimensions which is $\mathbb{R}^{l,m,m,\ldots,m,m,l+r}$. The tensor $\bar{\mathcal{L}}_i$ contains all entries of $\mathcal{L}_i$ and their transformation back and forth is one-to-one. Basically the smaller tensors are padded with themselves until they have the appropriate size.

Now we can safely stack $\bar{\mathcal{L}}_*$ over a new dimension to obtain the parameter tensor. Define:

Definition 2.3.2 *Define the tensor $\bar{\mathcal{L}} \in \mathbb{R}^{l,m,\ldots,m,l+r,p}$ as:*

$$\bar{\mathcal{L}} = \sum_{\bar{p}}^{p} \bar{\mathcal{L}}_{\bar{p}} \circ \begin{bmatrix} \bar{0}_{\bar{p}-1} \\ 1 \\ \bar{0}_{p-\bar{p}} \end{bmatrix}, \tag{2.30}$$

using Definition 2.3.1, (2.22), (2.22), (2.23) *and* (2.24). *Notice that $\bar{\mathcal{L}}$ and $C\mathcal{K}$ have the same entries (with some duplicates), but in different positions. Basically, we reorganized $C\mathcal{K}$ into $\bar{\mathcal{L}}$ in order to make it suitable for rank-R decomposition and tensor regression.*

We have finished deriving the highly-structured parameter tensor $\bar{\mathcal{L}}$. Its expression is different from the parameter matrix $C\mathcal{K}$, but contains the same LPV sub-Markov parameters. The new expression will be useful for clarifying which pieces of structure are discarded in state-of-the-art methods and why the curse-of-dimensionality appears. This relates strongly to the chosen parametrizations. In the next section, we present the parametrizations of both the state-of-the-art method and PBTR and investigate the resulting parameter counts.

2.3.3. PARAMETRIZATIONS

In this subsection, we present the parametrization of PBTR and compare it to the parametrization of state-of-the-art methods. We show which pieces of structure are ignored where, and what causes the curse-of-dimensionality to appear.

Consider the LPV predictor-based data equation:

$$y_k \approx C\mathcal{K}\mathcal{Z}_k + e_k \tag{2.31}$$

This equation is parametrized by state-of-the-art LPV subspace methods *element-wise* as:

$$\hat{y}_k(\theta) = [C\mathcal{K}](\theta)\mathcal{Z}_k \tag{2.32}$$

As a result, the parameter count of state-of-the-art LPV subspace methods is equal to the number of entries in $C\mathcal{K}$. Because this number scales exponentially with the past window p, so these methods suffer from the curse-of-dimensionality.

The PBTR is a tensor regression method, therefore we present a novel rewritten LPV data equation which is more suitable for tensor regression. This data equation uses the inner product of the parameter tensor $\bar{\mathcal{L}}$ (Definition 2.3.2) and and appropriate data tensor $\bar{\mathcal{Z}}_k$. This appropriate data tensor is a reorganization of $\mathcal{Z}_k$ which matches the reorganization of $C\mathcal{K}$ into $\bar{\mathcal{L}}$, where the data corresponding to duplicate parameters are scaled down. This tensor-form LPV data equation is:

$$y_k \approx <\bar{\mathcal{L}}, \bar{\mathcal{Z}}_k> + e_k, \tag{2.33}$$

where the inner product is redefined in order to deal with multiple outputs:

Definition 2.3.3 *Consider the inner product of two tensors:* $<\mathcal{T},\mathcal{U}>$. *Normally this requires* $\mathcal{T}$ *and* $\mathcal{U}$ *to have equal size. But in order to deal with multiple outputs, we extend the definition of this operator as follows. Let* $\mathcal{T} \in \mathbb{R}^{l,d_1,\ldots,d_N}$ *and* $\mathcal{U} \in \mathbb{R}^{d_1,\ldots,d_N}$. *Then their inner product exists and equals:*

$$<\mathcal{T},\mathcal{U}>= \begin{bmatrix} <\mathcal{T}_1,\mathcal{U}> \\ \vdots \\ <\mathcal{T}_l,\mathcal{U}> \end{bmatrix}, \tag{2.34}$$

where $\mathcal{T}_i \in \mathbb{R}^{d_1,\ldots,d_N}$ *is an appropriate part of* $\mathcal{T}$.

The parametrization of PBTR is a tensor regression parametrization, and as a result multi-linear in nature. Additionally, the parametrization revolves around the predictor-

based state-space matrices. It can then be written as:

$$\bar{\mathcal{L}}(\theta) = \sum_{\bar{p}}^{p} \bar{\mathcal{L}}_{\bar{p}}(\theta) \circ \begin{bmatrix} \bar{0}_{\bar{p}-1} \\ 1 \\ \bar{0}_{p-\bar{p}} \end{bmatrix} \tag{2.35a}$$

$$\bar{\mathcal{L}}_p(\theta) = \sum_{i,j,\delta_1,\ldots,\delta_{p-2}}^{n} \sum_{\bar{\kappa}=1}^{l+r} v_1^{(i)}(\theta_1) \circ v_2^{(i,\delta_1)}(\theta_{2,1}) \circ \ldots \circ v_2^{(\delta_{p-2},j)}(\theta_{2,p-1}) \circ v_3^{(j,\bar{\kappa})}(\theta_3) \circ v_4^{(\bar{\kappa})} \tag{2.35b}$$

$$\bar{\mathcal{L}}_{p-1}(\theta) = \sum_{i,j,\delta_1,\ldots,\delta_{p-2}}^{n} \sum_{\bar{\kappa}=1}^{l+r} v_1^{(i)}(\theta_1) \circ v_2^{(i,\delta_1)}(\theta_{2,1}) \circ \ldots \circ [I]_{\delta_{p-2},j} \bar{1}_m \circ v_3^{(j,\bar{\kappa})}(\theta_3) \circ v_4^{(\bar{\kappa})} \tag{2.35c}$$

where each individual vector is parametrized element-wise. The sizes of the parameter group are as follows. The θ_1 has ln parameters, $\theta_{2,i}$ has n^2m parameters for all $i \in \{1,\ldots,p-1\}$, and θ_3 has $(l+r)nm$ parameters. Notice that this parametrization is not a direct parametrization in the LPV predictor-based state-space matrices (2.4), because the $\tilde{A}^{(i)}$ are spuriously parametrized in order to obtain a multi-linear parametrization. Notice that PBTR exploits more structure than the state-of-the-art LPV subspace methods do and does not suffer from the curse-of-dimensionality.

It is also possible to use the polynomial non-convex method (Verdult et al., 2003), which enforces available structure by directly parametrizing the regular state-space matrices. Notice that this method does not have a second estimation step. This method also does not suffer from the curse-of-dimensionality, and the surplus enforced structure slightly further reduces the parameter count. Notice that both PBTR and the polynomial non-convex method have a non-convex parametrization. Therefore they require an initial estimate (including a model order). This initial estimate can be obtained from state-of-the-art (LPV subspace) methods. This places PBTR and the polynomial non-convex method as refinement methods for LPV subspace methods.

The parameter counts are summarized in Table 2.3.3. Notice that the parameter counts of the evaluated methods scale differently. The state-of-the-art LPV subspace methods suffer from the curse-of-dimensionality, because their parameter count scales exponentially with p. Usage of kernels (dual approaches) changes the parameter count to scale with data instead, but have the disadvantage that they result in ill-conditioned problems. Regularization can only partially solve this problem (van Wingerden and Verhaegen, 2009). Both PBTR and the polynomial method do not suffer from the curse-of-dimensionality.

This concludes the evaluation of the parameter counts of PBTR and state-of-the-art methods. In the next subsection, the full PBTR algorithm is presented.

2.3.4. ALGORITHM

Algorithm 2.3.1 *The PBTR*

Method	Parameter count
LPV-PBSID$_{opt}$ (primal)	$l(r+l)\sum_{j=1}^{p} m^j$
LPV-PBSID$_{opt}$ (dual)	$l(N-2p)$
PBTR (with free parametrization)	$nl+n(l+r)m+n^2m(p-1)$
Polynomial method	$nl+nrm+n^2m$

Table 2.1: Comparison of the parameter counts for the (first) estimation step

Define the cost function of PBTR as:

$$V_N(\theta) = \sum_{k=1}^{N} (y_k - \hat{y}_k(\theta))^T (y_k - \hat{y}_k(\theta)), \tag{2.36a}$$

$$\hat{y}_k(\theta) = < \bar{\mathcal{L}}(\theta), \bar{\mathcal{Z}}_k > \tag{2.36b}$$

where $\hat{y}_k$ is the model output, $\bar{\mathcal{L}}(\theta)$ is defined in (2.35) *and $\bar{\mathcal{Z}}_k$ is defined in the previous subsection. This is a multi-linear parametrization in the predictor-based state-space matrices with additional structure. It is possible to obtain a consistent estimate of all θ using multi-linear optimization (Zhou et al., 2013), for cases where equation* (2.6) *is not an approximation but an equality. This can be done using Alternating Least Squares (Zhou et al., 2013) or MATLAB's 'fmin' command. Notice that the only indeterminacy is modulo global state-coordinate transformation, which is common. After obtaining an estimate of $\bar{\mathcal{L}}$, an estimate of $C\mathcal{K}$ can be directly and one-to-one constructed. The succeeding steps follow the same methodology as other predictor-based subspace methods as presented in Section 2.2.*

This concludes the section on PBTR. In the next section, the simulation results are presented.

2.4. SIMULATIONS

In this section simulation results are presented in order to compare PBTR with state-of-the-art methods for several cases in terms of bias, variance and parameter count.

2.4.1. SIMULATION SETTINGS

In this subsection, the general simulation settings and some definitions are presented.

The results presented in this paper are based on 100 Monte Carlo simulations. For every Monte Carlo simulation different realizations of both the input and the innovation vector were used. During every Monte Carlo simulation, first estimates of the state-space matrices are obtained from both the unregularized and the regularized variant of the LPV subspace method of van Wingerden and Verhaegen (2009). Then the estimate of the regularized variant is used as an initial estimate for the non-convex methods: PBTR and the

method of Verdult et al. (2003). It is worth noting that we do not consider the prediction error variant of the method of Verdult et al. (2003), because the authors indicated that that variant performs badly. The variant that we do use, fixes the parameters of K to zero in order to somewhat relieve its involved parametrization. All methods are provided with the system order n, which is assumed to be known, and the information that $D = 0$ and C is LTI.

We also present the following settings which are the same for every case. The matrix $K^{(i)}$ for $i = \{1,\ldots,m\}$ is obtained from the Discrete Algebraic Ricatti Equation (DARE) with $A^{(i)}$, C and identity covariance of the concatenated process and measurement noise. Every signal of the input vector u_k and innovation vector e_k is white noise. The data size N is chosen as 200, and both the past window p and the future window f are 6.

The quality of the estimates is evaluated by investigating the Variance Accounted For (VAF) on a validation data set *different* from the one used for identification, in the sense that different realizations of both the input and the innovation vector are used. The VAF for single-output systems is defined as (van Wingerden and Verhaegen, 2009):

$$VAF(\bar{y}_k, \hat{y}_k) = max\left\{1 - \frac{var(\bar{y}_k - \hat{y}_k)}{var(\bar{y}_k)}, 0\right\} * 100\%$$

Notice that the noise-free simulated output of the system can be used when evaluating the VAF, because the data is obtained from simulations. This allows the VAF to reach 100% when the model is equal to the true system modulo global state-coordinate transformations, such that the analysis becomes more clear. The noise-free (simulated) output of the system is denoted as $\bar{y}_k$. Similarly, $\hat{y}_k$ is used for the noise-free (simulated) model output. The $var(*)$ operator denotes the variance.

In the case that a non-convex method produces an estimate with an identification data VAF less than half the identification data VAF of the initial estimate, the refined estimate is rejected and substituted directly by the initial estimate. This is possible, because the identification data is available during estimation. Notice that this does not prevent local minima, but merely serves to reject poor optimization results.

The cases and their results are presented in the following subsection. A parameter count investigation is performed in the last subsection.

2.4.2. SIMULATION RESULTS CASE 1

This case uses the following LPV state-space system (2.3):

$$[A^{(1)}, A^{(2)}] = \left[\begin{array}{cc|cc} \frac{4}{15} & \frac{1}{15} & \frac{3}{20} & -\frac{1}{60} \\ -\frac{1}{6} & \frac{1}{30} & -\frac{1}{60} & \frac{3}{20} \end{array}\right],$$

$$[B^{(1)}, B^{(2)}] = \left[\begin{array}{c|c} 1 & 0.2 \\ 0 & 0.2 \end{array}\right], C = \begin{bmatrix} 1 & 0 \end{bmatrix},$$

and the Signal-to-Noise Ratio (SNR) is 1. The remaining settings are as described in Subsection 2.4.1. The system is evaluated at two different affine scheduling sequences with:

$$\mu_k^{(2)} = cos(2\pi k \frac{\Pi}{N})/2 + 0.2,$$

where $\Pi = 20$ for the first and $\Pi = 4$ for the second scheduling sequence.

Scheduling	Method	VAF
$\Pi = 20$	LPV-PBSID$_{opt}$ (kernel)	95.8
	Reg. LPV-PBSID$_{opt}$ (kernel)	96.6
	PBTR	98.0
	Polynomial non-convex method	96.5
$\Pi = 4$	LPV-PBSID$_{opt}$ (kernel)	23
	Reg. LPV-PBSID$_{opt}$ (kernel)	97.0
	PBTR	98.1
	Polynomial non-convex method	96.7

Table 2.2: Mean VAF for different methods for Case 1

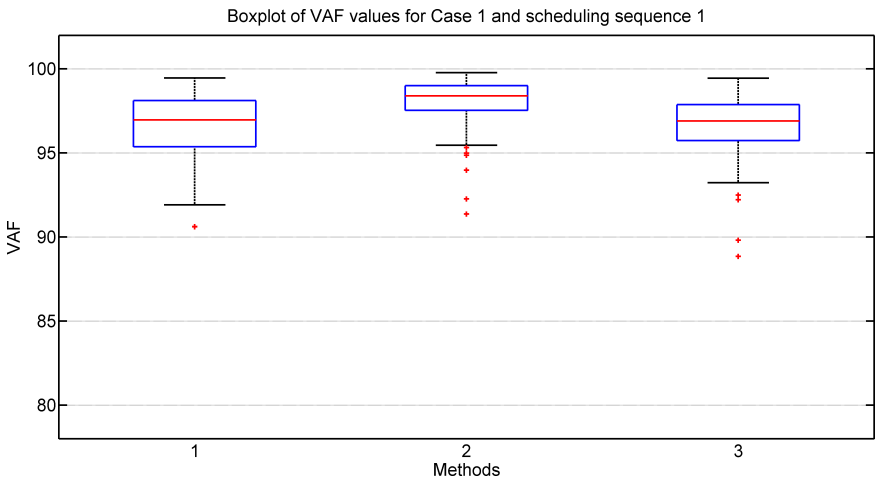


Figure 2.1: Boxplots of the VAF results of Case 1 for the evaluated methods at scheduling sequence 1. The methods are: 1. Reg. LPV-PBSID$_{opt}$, 2. PBTR, 3. Polynomial method.

From the results of Table 2.2 it can be seen that PBTR has superior VAF in comparison to the other methods. The other (polynomial) non-convex method has comparable VAF to state-of-the-art LPV subspace method. Furthermore, the VAF of the unregularized variant of the state-of-the-art LPV subspace method appears to vary considerably with the scheduling sequence. These results are further supported by Fig. 2.1 and 2.2.

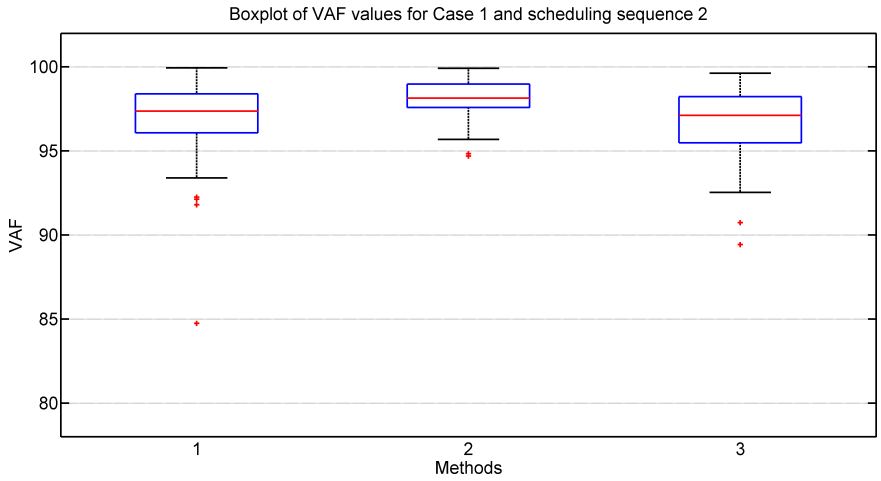


Figure 2.2: Boxplots of the VAF results of Case 1 for the evaluated methods at the scheduling sequence 2. The methods are: 1. Reg. LPV-PBSID$_{opt}$, 2. PBTR, 3. Polynomial method.

2.4.3. SIMULATION RESULTS CASE 2

This case uses the following LPV state-space system (2.3):

$$[A^{(1)}, A^{(2)}, A^{(3)}] = \left[\begin{array}{cc|cc|cc} \frac{4}{15} & \frac{1}{15} & \frac{3}{20} & -\frac{1}{60} & \frac{29}{405} & \frac{2}{81} \\ -\frac{1}{6} & \frac{1}{30} & -\frac{1}{60} & \frac{3}{20} & \frac{1}{81} & \frac{52}{405} \end{array}\right],$$

$$[B^{(1)}, B^{(2)}, B^{(3)}] = \left[\begin{array}{c|c|c} 1 & 0.2 & 0.2 \\ 0 & 0.2 & -0.2 \end{array}\right], C = \begin{bmatrix} 1 & 0 \end{bmatrix},$$

and the Signal-to-Noise Ratio (SNR) is 2. The remaining settings are as described in Subsection 2.4.1. The system is evaluated at two different affine scheduling sequences with:

$$\mu_k^{(2)} = cos(2\pi k \frac{\Pi}{N})/2 + 0.2$$
$$\mu_k^{(3)} = cos(2\pi k \frac{\Pi}{2.5N} + 0.5\pi)/3,$$

where $\Pi = 20$ for the first and $\Pi = 4$ for the second scheduling sequence.

From the results of Table 2.3 it can be seen that PBTR has superior VAF in comparison to the other methods. The other (polynomial) non-convex method has comparable VAF to state-of-the-art LPV subspace method. Furthermore, the VAF of the unregularized variant of the state-of-the-art LPV subspace method appears to vary considerably with the scheduling sequence. These results are further supported by Fig. 2.3 and 2.4.

Scheduling	Method	VAF
$\Pi = 20$	LPV-PBSID$_{opt}$ (kernel)	81
	Reg. LPV-PBSID$_{opt}$ (kernel)	97.4
	PBTR	98.4
	Polynomial non-convex method	97.6
$\Pi = 4$	LPV-PBSID$_{opt}$ (kernel)	7.3
	Reg. LPV-PBSID$_{opt}$ (kernel)	97.7
	PBTR	98.4
	Polynomial non-convex method	97.6

Table 2.3: Mean VAF for different methods for Case 2

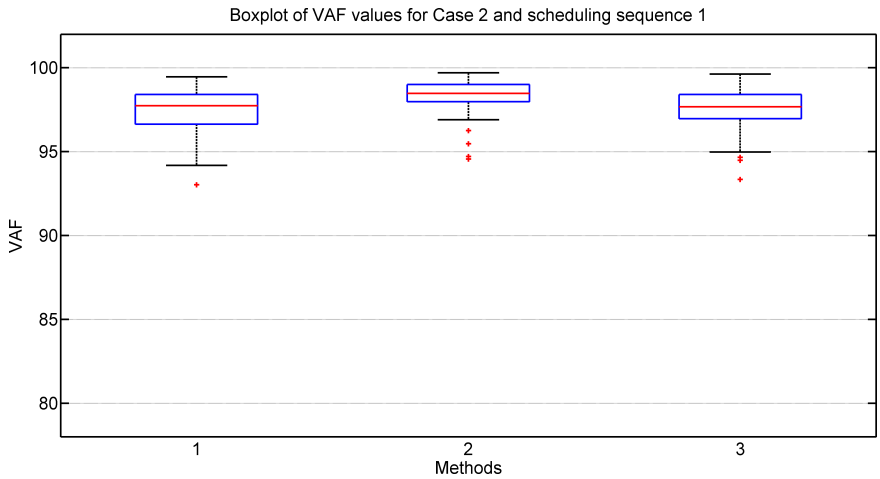


Figure 2.3: Boxplots of the VAF results of Case 2 for the evaluated methods at scheduling sequence 1. The methods are: 1. Reg. LPV-PBSID$_{opt}$, 2. PBTR, 3. Polynomial method.

2.4.4. SIMULATION RESULTS CASE 3

This case uses the following LPV state-space system (2.3) with:

$$[A^{(1)}, A^{(2)}] = \left[\begin{array}{cccc|cccc} \frac{-1}{300} & \frac{1}{30} & \frac{11}{75} & \frac{8}{75} & \frac{-3}{10} & \frac{1}{6} & \frac{11}{30} & \frac{11}{30} \\ \frac{3}{20} & \frac{1}{20} & \frac{-3}{20} & \frac{-3}{20} & \frac{1}{20} & \frac{7}{60} & \frac{-1}{20} & \frac{-1}{20} \\ \frac{-29}{100} & \frac{1}{10} & \frac{32}{75} & \frac{7}{25} & \frac{-9}{20} & \frac{3}{20} & \frac{31}{60} & \frac{9}{20} \\ \frac{11}{300} & \frac{-1}{60} & \frac{-3}{100} & \frac{23}{300} & \frac{-1}{30} & \frac{1}{30} & \frac{1}{30} & \frac{1}{10} \end{array}\right],$$

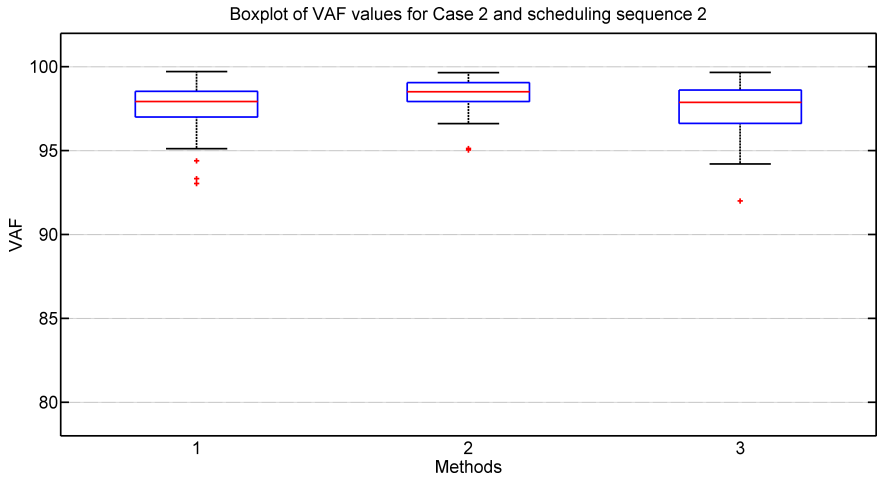


Figure 2.4: Boxplots of the VAF results of Case 2 for the evaluated methods at the scheduling sequence 2. The methods are: 1. Reg. LPV-PBSID$_{opt}$, 2. PBTR, 3. Polynomial method.

and

$$[B^{(1)}, B^{(2)}] = \left[\begin{array}{c|c} 1 & 0.2 \\ 0 & 0.2 \\ 0 & 0.2 \\ 0 & 0.2 \end{array}\right], C = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix},$$

and the Signal-to-Noise Ratio (SNR) is 0.5. The remaining settings are as described in Subsection 2.4.1. The system is evaluated at the affine scheduling sequence with:

$$\mu_k^{(2)} = cos(2\pi k \frac{20}{N})/2 + 0.2,$$

Method	VAF
LPV-PBSID$_{opt}$ (kernel)	81
Reg. LPV-PBSID$_{opt}$ (kernel)	85.1
PBTR	90.8
Polynomial non-convex method	80.9

Table 2.4: Mean VAF for different methods for Case 3

From the results of Table 2.4 it can be seen that PBTR has superior VAF in comparison to the other methods. The other (polynomial) non-convex method however fails to

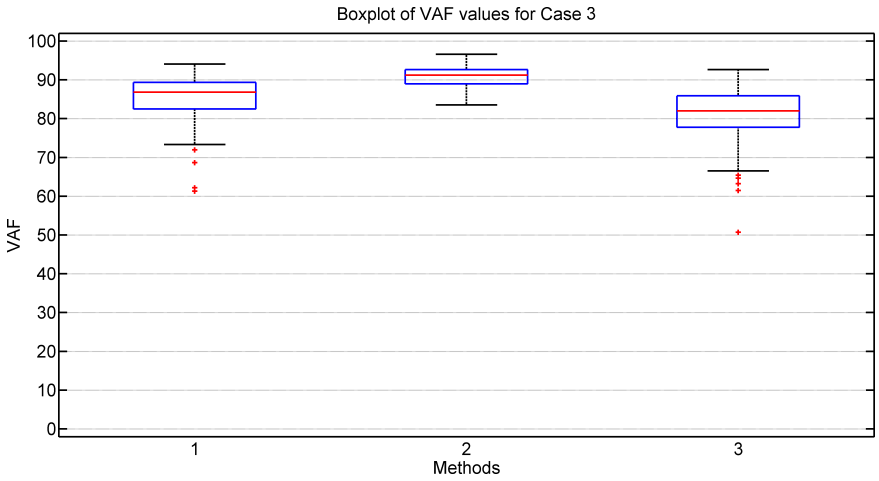


Figure 2.5: Boxplots of the VAF results of Case 3 for the evaluated methods. The methods are: 1. Reg. LPV-PBSID$_{opt}$, 2. PBTR, 3. Polynomial method.

refine the initial estimates supplied by the state-of-the-art LPV subspace method. These results are supported by Fig. 2.5.

2.4.5. PARAMETER COUNTS

The parameter counts of the evaluated methods for Cases 1 and 2 are presented in Table 2.5. It is visible that the PBTR has a parameter count roughly in between the state-of-the-art LPV subspace methods and the polynomial non-convex method. The PBTR does not suffer from the curse-of-dimensionality, while also having a superior performance in terms of VAF as shown in the previous subsections.

Method	**Case 1**	**Case 2**
LPV-PBSID$_{opt}$ (primal)	252	2184
LPV-PBSID$_{opt}$ (dual)	188	188
PBTR (with free parametrization)	50	74
Polynomial method	18	26

Table 2.5: Comparison of the parameter counts for the (first) estimation step for some cases

2.5. CONCLUSIONS

In this paper a novel method for LPV identification was presented, which is named PBTR. The benefit of PBTR over state-of-the-art LPV subspace methods is that it does not suf-

fer from the curse-of-dimensionality. This was achieved by first pinpointing the origin of the curse-of-dimensionality, which appeared to be the ignoring of inherent structure of the LPV sub-Markov parameters, and then using tensor regression to prevent it. This does make PBTR a non-convex method. The difference of PBTR with other non-convex methods is that it uses tensor regression to exploit only the structure necessary to avoid the curse-of-dimensionality. Though the formal proof remains an open issue, simulation results show that PBTR has better performance with respect to state-of-the-art LPV subspace techniques and non-convex techniques by looking at the variance.

BIBLIOGRAPHY

Balas, G. J. Linear, parameter-varying control and its application to aerospace systems. In *ICAS Congress Proceedings*, 2002.

Bamieh, B. and Giarre, L. Identification of linear parameter varying models. *International journal of robust and nonlinear control*, 12(9):841–853, 2002.

Bianchi, F. D., Mantz, R. J., and Christiansen, C. F. Gain scheduling control of variable-speed wind energy conversion systems using quasi-LPV models. *Control Engineering Practice*, 13(2):247–255, 2005.

Brewer, J. Kronecker products and matrix calculus in system theory. *IEEE Transactions on circuits and systems*, 25(9):772–781, 1978.

Chiuso, A. The role of vector autoregressive modeling in predictor-based subspace identification. *Automatica*, 43(6):1034–1048, 2007.

De Caigny, J., Camino, J. F., and Swevers, J. Interpolating model identification for SISO linear parameter-varying systems. *Mechanical Systems and Signal Processing*, 23(8): 2395–2417, 2009.

Favoreel, W., De Moor, B., and Van Overschee, P. Subspace identification of bilinear systems subject to white inputs. *Automatic Control, IEEE Transactions on*, 44(6):1157–1165, 1999.

Felici, F., Van Wingerden, J.-W., and Verhaegen, M. Subspace identification of MIMO LPV systems using a periodic scheduling sequence. *Automatica*, 43(10):1684–1697, 2007.

Gebraad, P. M., van Wingerden, J.-W., van der Veen, G. J., and Verhaegen, M. LPV subspace identification using a novel nuclear norm regularization method. In *American Control Conference (ACC)*, 2011.

Gebraad, P. M., van Wingerden, J.-W., Fleming, P. A., and Wright, A. D. LPV identification of wind turbine rotor vibrational dynamics using periodic disturbance basis functions. *Control Systems Technology, IEEE Transactions on*, 21(4):1183–1190, 2013.

Giarré, L., Bauso, D., Falugi, P., and Bamieh, B. LPV model identification for gain scheduling control: An application to rotating stall and surge control problem. *Control Engineering Practice*, 14(4):351–361, 2006.

Golub, G. H., Heath, M., and Wahba, G. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.

Gunes, B., van Wingerden, J.-W., and Verhaegen, M. Tensor regression for LTI subspace identification. In *American Control Conference (ACC), 2015*, pages 1131–1136. IEEE, 2015a.

Gunes, B., van Wingerden, J.-W., and Verhaegen, M. Tensor regression for LTI subspace identification: free parametrizations. *SYmposium on System IDentification, IFAC-PapersOnLine*, 48(28):909–914, 2015b.

Gunes, B., van Wingerden, J.-W., and Verhaegen, M. Tensor regression for LPV subspace identification. *SYmposium on System IDentification, IFAC-PapersOnLine*, 48(28):421–426, 2015c.

Gunes, B., van Wingerden, J.-W., and Verhaegen, M. Predictor-based tensor regression (PBTR) for LPV subspace identification. *Automatica*, 79:235 – 243, 2017. ISSN 0005-1098.

Guo, W., Kotsia, I., and Patras, I. Tensor learning for regression. *Image Processing, IEEE Transactions on*, 21(2):816–827, 2012.

Jansson, M. A new subspace identification method for open and closed loop data. *IFAC Proceedings Volumes*, 38(1):500–505, 2005.

Knudsen, T. Consistency analysis of subspace identification methods based on a linear regression approach. *Automatica*, 37(1):81–89, 2001.

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

Lee, L. H. and Poolla, K. Identification of linear parameter-varying systems using nonlinear programming. *Journal of dynamic systems, measurement, and control*, 121(1): 71–78, 1999.

Ljung, L. *System identification (2nd ed.): theory for the user*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999. ISBN 0-13-656695-2.

Lovera, M. and Mercere, G. Identification for gain-scheduling: a balanced subspace approach. In *American Control Conference 2007, ACC'07*, page CDROM, 2007.

Lu, H., Plataniotis, K. N., and Venetsanopoulos, A. N. MPCA: Multilinear principal component analysis of tensor objects. *Neural Networks, IEEE Transactions on*, 19(1):18–39, 2008.

Lu, H., Plataniotis, K. N., and Venetsanopoulos, A. N. A survey of multilinear subspace learning for tensor data. *Pattern Recognition*, 44(7):1540–1551, 2011.

Remmlinger, J., Buchholz, M., and Dietmayer, K. Identification of a bilinear and parameter-varying model for lithium-ion batteries by subspace methods. In *American Control Conference (ACC), 2013*, pages 2268–2273. IEEE, 2013.

Scherer, C. W. LPV control and full block multipliers. *Automatica*, 37(3):361–375, 2001.

Shamma, J. S. An overview of LPV systems. In *Control of linear parameter varying systems with applications*, pages 3–26. Springer, 2012.

Signoretto, M., De Lathauwer, L., and Suykens, J. A. Nuclear norms for tensors and their use for convex multilinear estimation. *Submitted to Linear Algebra and Its Applications*, 43, 2010.

Tóth, R., Abbas, H. S., and Werner, H. On the state-space realization of LPV input-output models: Practical approaches. *Control Systems Technology, IEEE Transactions on*, 20 (1):139–153, 2012.

van Wingerden, J.-W. and Verhaegen, M. Subspace identification of bilinear and LPV systems for open- and closed-loop data. *Automatica*, 45(2):372 – 381, 2009. ISSN 0005-1098.

van Wingerden, J.-W., Felici, F., and Verhaegen, M. Subspace identification of MIMO LPV systems using a piecewise constant scheduling sequence with hard/soft switching. In *European Control Conference (ECC), 2007*, pages 927–934. IEEE, 2007.

Vasilescu, M. A. O. and Terzopoulos, D. Multilinear analysis of image ensembles: Tensorfaces. In *Computer Vision ECCV 2002*, pages 447–460. Springer, 2002.

Verdult, V., Bergboer, N., and Verhaegen, M. Identification of fully parameterized linear and nonlinear state-space systems by projected gradient search. In *Proceedings of the 13th IFAC Symposium on System Identification, Rotterdam*, 2003.

Zhou, H., Li, L., and Zhu, H. Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association*, 108(502):540–552, 2013.

3

TENSOR NUCLEAR NORM LPV SUBSPACE IDENTIFICATION

Linear Parameter Varying (LPV) subspace identification methods suffer from an exponential growth in number of parameters to estimate. This results in problems with ill-conditioning, memory and computation costs. In literature, attempts have been made to address the ill-conditioning by using regularization and curse-of-dimensionality by using dual problems. The effectiveness of regularization hinges on suitable a priori knowledge. In this paper we propose using an alternative regularization. That is, we first show that the LPV sub-Markov parameters can be organized into several tensors which are multi-linear low-rank *by construction. Namely, their matricization along any mode is a low-rank matrix. Then we propose a novel convex method with tensor nuclear norm regularization which exploits this low-rank property. Simulation results show that the novel method has higher performance than the regularized LPV-PBSID$_{opt}$ technique in terms of bias and variance.*

3.1. INTRODUCTION

Linear Parameter Varying (LPV) systems have been used in many applications. Some examples are wind turbines (Bianchi et al., 2005; Gebraad et al., 2011a), aircraft applications (Balas, 2002), batteries (Remmlinger et al., 2013), compressors (Giarré et al., 2006) and wafer stages (Wassink et al., 2005; van der Maas et al., 2015). These LPV systems are linear systems whose dynamics vary with a known time-varying parameter vector. They are suitable for describing many applications in greater detail than LTI systems can. Just like for LTI systems, there also exists a powerful control design framework for LPV systems (Scherer, 2001), which can guarantee stability, performance and robustness. This is generally not the case for non-linear systems. Most control design frameworks do require an LPV state-space model of the system.

This chapter is an extended version of the paper (Gunes et al., 2018) published in IEEE Transactions on Automatic Control.

This model can be obtained from measurement data using system identification. There are many LPV identification methods which can be classified as global and local methods (Tóth, 2010; De Caigny et al., 2009; Shamma, 2012). Local methods hinge on the property that for constant scheduling parameters the LPV system behaves as an LTI system. They identify LTI models at several fixed constant scheduling parameter conditions and then use interpolation techniques to obtain an LPV model. For applications where these experiments are possible, this can yield good results (De Caigny et al., 2009; Tóth, 2010; Leith and Leithead, 2000; Rugh and Shamma, 2000; Shamma, 2012). Global methods on the other hand do not have this requirement and only use one experiment. In this paper we focus on global LPV identification methods.

These methods can be further divided into input-output and subspace methods. Input-output methods yield input-output LPV models and have received considerable attention in the literature (Tóth, 2010; Laurain et al., 2010; Butcher et al., 2008). But the preferred model structure for mainstream LPV control design methodologies is state-space, and transformation from input-output to state-space models is problematic in the LPV case (Tóth et al., 2012). Subspace methods have also received considerable attention (van Wingerden and Verhaegen, 2009; Felici et al., 2007; Favoreel et al., 1999; van Wingerden et al., 2007; Gebraad et al., 2011a; Cox and Tóth, 2016; Larimore and Buchholz, 2012). They directly produce state-space models, and can deal naturally with multiple input multiple output and closed-loop systems. In this paper we focus on subspace methods. If the scheduling sequence has some special structure, for example, is periodic (Felici et al., 2007), white noise (Favoreel et al., 1999) or piecewise constant (van Wingerden et al., 2007), then tailored methods can be used. However, this is not true for all applications. There exist several LPV subspace methods for this case (van Wingerden and Verhaegen, 2009; Gebraad et al., 2011b; Cox and Tóth, 2016), and they suffer from the 'curse-of-dimensionality' during their first regression step. This means that the number of LPV sub-Markov parameters to be estimated can quickly, vastly exceed the number of data points. This results in ill-conditioned problems, memory and computational cost issues. The memory and computational costs can be reduced by solving dual problems (van Wingerden and Verhaegen, 2009). One way to tackle ill-condition of the problem is to use regularization. Methods with Tikhonov regularization (van Wingerden and Verhaegen, 2009) and matrix nuclear norms (Gebraad et al., 2011a) have been proposed. The former penalizes the magnitude of estimate parameters, and the latter exploits the approximate low-rank property of the state-revealing matrix. In this paper we propose using an alternative regularization, which arguably exploits more structure. Namely, we propose using tensor nuclear norms (Signoretto et al., 2010), in order to exploit the exact multi-linear low-rank property of the parameter tensor.

Our proposed method can also be seen as an extension of the method presented in Hjalmarsson et al. (2012). That is, our method extends it to the LPV Multiple Input Multiple Output (MIMO) with tensors case. Furthermore we provide numeric efficiency contributions, in order to perform the LPV case computations 'curse-of-dimensionality' free.

A tensor is a multi-dimensional generalization of a matrix. That is, it can have more than two dimensions. Then this multi-dimensional structure can be exploited using techniques from the tensor framework such as tensor nuclear norms (Signoretto et al.,

2010). Tensor nuclear norms and nuclear norms in general have received considerable attention in literature (Recht et al., 2010; Verhaegen and Hansson, 2014; Signoretto et al., 2010; He et al., 2005; Vasilescu and Terzopoulos, 2002). Matrix nuclear norms can be used to exploit knowledge that some matrices are low-rank. Tensor nuclear norms can be used to exploit knowledge that some tensors have low-rank matricizations.

In this paper we propose a novel convex LPV subspace identification method which uses these tensor nuclear norms as regularization. Namely, as the major contribution we show and exploit that the LPV sub-Markov parameters can be organized into several tensors whose matricizations are all low-rank. Additionally, we present three minor contributions to make the optimization 'curse-of-dimensionality' free. Firstly, we show how these regularization terms can be expressed in the dual problem similar to van Wingerden and Verhaegen (2009). Secondly, we present how the arguments of these tensor nuclear norms can be replaced by 'curse-of-dimensionality' free arguments while producing the exact same norm result. Thirdly, we show how these norm results can be computed 'curse-of-dimensionality' free by using tensor trains (Oseledets, 2011). In short, we propose a novel method with tensor nuclear norms together with means to perform its optimization 'curse-of-dimensionality' free.

This paper relates to previous work as follows. In Gunes et al. (2017a,b), the 'curse-of-dimensionality' of the LPV sub-Markov parameters was also tackled by using tensor techniques. However, in Gunes et al. (2017a) the LPV sub-Markov parameters were organized in a single padded tensor and then a non-linear (polyadic) decomposition and parametrization was used. In Gunes et al. (2017b), a tensor train decomposition and parametrization is used. Both methods do not use multi-linear low-rank properties or nuclear norms or dual problems and are non-convex refinement methods. The full LPV sub-Markov tensors and data tensors used in this paper have been generalized from the tensors presented before in Gunes et al. (2017b). Additionally, the tensor train decomposition of the data tensors have also been generalized from the decomposition in Gunes et al. (2017b). It is worth remarking that the proposed method is a convex method.

Before presenting the novel method, first essential background material is provided in Section 3.2. This section covers LPV subspace identification, tensor decompositions, matrix and tensor nuclear norms. Afterwards the LPV sub-Markov parameter tensors are presented explicitly in Section 3.3 and proven to be of multi-linear low-rank. Then in Section 3.5 it is shown how these tensor nuclear norms can be computed 'curse-of-dimensionality'-free. This requires the tensor train decompositions of the data tensors in Section 3.4. Finally the novel method is presented in Section 3.6 and a comparison with the regularized LPV-PBSID$_{opt}$ method of van Wingerden and Verhaegen (2009) through simulation results are presented in Section 3.7.

3.2. BACKGROUND

Prior to presenting the novel method, several related topics are reviewed in this section. These topics are LPV subspace identification, tensor decompositions and matrix and tensor nuclear norms.

3.2.1. FIRST REGRESSION STEP OF LPV SUBSPACE IDENTIFICATION

In this subsection we review LPV subspace identification van Wingerden and Verhaegen (2009). Firstly, define the signals relevant to the LPV state-space description:

$$x_k \in \mathbb{R}^{\hat{n}},\ u_k \in \mathbb{R}^r,\ y_k \in \mathbb{R}^l,\ \mu_k \in \mathbb{R}^m, \tag{3.1}$$

as the state, input, output and scheduling sequence vector at sample number k. The number $\hat{n}$ is the system order. Additionally, define the column vector:

$$\mu_k = \begin{bmatrix} \mu_k^{(1)} & \dots & \mu_k^{(m)} \end{bmatrix}^T \tag{3.2}$$

Predictor-based methods (Chiuso, 2007; van Wingerden and Verhaegen, 2009) make use of the innovation representation and predictor-based representation. Therefore the assumed data-generating system is directly presented as:

$$x_{k+1} = \sum_{i=1}^{m} \mu_k^{(i)} \left(A^{(i)} x_k + B^{(i)} u_k + K^{(i)} e_k \right) \tag{3.3a}$$

$$y_k = C x_k + e_k, \tag{3.3b}$$

where the matrices $A^{(i)}$, $B^{(i)}$, C and $K^{(i)}$ are appropriately dimensioned state, input, output and observer matrices and e_k is the innovation signal at sample k. Here, an LTI C matrix and zero matrix feed-through term D is used as in van Wingerden and Verhaegen (2009). This is an assumption which is made for presentation and simplicity of derivation. However, this does not trivialize the bottleneck 'curse-of-dimensionality'. Furthermore, we assume μ_k is known and this state-space system has affine dependency on μ_k (van Wingerden and Verhaegen, 2009). That is, $\mu_k^{(1)} = 1\ \forall k$. By substituting (3.3b) in (3.3a), the predictor-based representation can be obtained as:

$$x_{k+1} = \sum_{i=1}^{m} \mu_k^{(i)} \left(\tilde{A}^{(i)} x_k + \bar{B}^{(i)} \begin{bmatrix} u_k \\ y_k \end{bmatrix} \right) \tag{3.4a}$$

$$y_k = C x_k + e_k, \tag{3.4b}$$

where $\tilde{A}^{(i)}$ is $A^{(i)} - K^{(i)} C$ and $\bar{B}^{(i)}$ is $[B^{(i)}, K^{(i)}]$. Notice that these states are the observer states. Also, we define p as a past window. Additionally, define the discrete-time time-varying transition matrix (van Wingerden and Verhaegen, 2009):

$$\tilde{A}_k = \sum_{i=1}^{m} \mu_k^{(i)} \tilde{A}^{(i)} \tag{3.5a}$$

$$\phi_{j,k} = \tilde{A}_{k+j-1} \dots \tilde{A}_{k+1} \tilde{A}_k \tag{3.5b}$$

The key assumption of predictor-based methods is that this matrix is exactly zero when j is greater than or equal to the past window p (van Wingerden and Verhaegen, 2009; Chiuso, 2007):

$$\phi_{j,k} \approx 0\ \forall\ j \geq p \tag{3.6}$$

This approximation is also used in several LTI methods (van der Veen et al., 2013). The resulting approximation error can be made arbitrarily small by increasing p if the predictor-based system is uniformly exponentially stable (Knudsen, 2001). Furthermore, the introduced bias also disappears as p goes to infinity (but remains hard to quantify for finite p (Chiuso, 2007)). This assumption is used to obtain the predictor-based data equation.

But before presenting this data equation, some definitions are given. Define for brevity the integer q:

$$q = (l+r)\sum_{j=1}^{p} m^j, \tag{3.7}$$

The predictor-based data equation involves the extended LPV controllability matrix (van Wingerden and Verhaegen, 2009), which is defined as:

$$\mathcal{K} = \begin{bmatrix} L_p & \dots & L_2 & \bar{B} \end{bmatrix} \qquad \in \mathbb{R}^{\hat{n}\times q}, \tag{3.8a}$$

$$\bar{B} = \begin{bmatrix} \bar{B}^{(1)} & \dots & \bar{B}^{(m)} \end{bmatrix} \qquad \in \mathbb{R}^{\hat{n}\times (l+r)m} \tag{3.8b}$$

$$L_2 = \begin{bmatrix} \tilde{A}^{(1)}\bar{B} & \dots & \tilde{A}^{(m)}\bar{B} \end{bmatrix} \qquad \in \mathbb{R}^{\hat{n}\times (l+r)m^2} \tag{3.8c}$$

$$L_{i+1} = \begin{bmatrix} \tilde{A}^{(1)}L_i & \dots & \tilde{A}^{(m)}L_i \end{bmatrix} \qquad \in \mathbb{R}^{\hat{n}\times (l+r)m^{i+1}}, 2 \geq i \geq p-1 \tag{3.8d}$$

This data equation also involves the 'effective' data matrix (van Wingerden and Verhaegen, 2009):

$$\mathcal{Z}_{k+p} = N_k^p Z_{k+p} \in \mathbb{R}^q, \tag{3.9}$$

where Z_k is:

$$z_{k+p} = \begin{bmatrix} u_{k+p} \\ y_{k+p} \end{bmatrix}, \tag{3.10a}$$

$$Z_{k+p} = \begin{bmatrix} z_k \\ z_{k+1} \\ \vdots \\ z_{k+p-1} \end{bmatrix} \tag{3.10b}$$

and N_k^p is:

$$N_k^p = \begin{bmatrix} \tilde{P}_{p|k} & 0 & \cdots & 0 \\ 0 & \tilde{P}_{p-1|k+1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{P}_{1|k+p-1} \end{bmatrix}, \tag{3.11a}$$

$$\tilde{P}_{p|k} = \mu_{k+p-1} \otimes \cdots \otimes \mu_k \otimes I_{r+l}, \tag{3.11b}$$

where μ_k is a column vector and the operator $\otimes$ is the Kronecker product (Brewer, 1978). One detail is that $\mathcal{Z}_{k+p}$ involves samples from k to $k+p-1$. These definitions and equation (3.4) allow describing the states as:

$$x_{k+p} = \phi_{p,k} x_k + \mathcal{K}\mathcal{Z}_{k+p} \tag{3.12}$$

The effect of the initial state x_k can be neglected using assumption (3.6) to obtain:

$$x_{k+p} \approx \mathcal{K}\mathcal{Z}_{k+p}, \tag{3.13}$$

which directly results in the predictor-based data equation:

$$y_{k+p} \approx C\mathcal{K}\mathcal{Z}_{k+p} + e_{k+p} \tag{3.14}$$

For brevity, we name the entries of $C\mathcal{K}$ as the LPV sub-Markov parameters.

However, notice that the size of $C\mathcal{K}$ and $\mathcal{Z}_{k+p}$ require special care. These matrices and their sizes are inherent to LPV predictor-based methods. Namely, their number of elements scale exponentially with the past window. Additionally, as argued earlier in this section the past window affects the approximation error and bias. In the remainder of this paper this size increase will be referred to as 'the curse-of-dimensionality'. Additionally, we refer to the absence of 'the curse-of-dimensionality' as 'curse-of-dimensionality free'.

With these results, an estimate of the LPV state-space matrices can be obtained as follows. Firstly, stack the data over all relevant samples as:

$$\mathcal{Z} = \begin{bmatrix} \mathcal{Z}_{p+1} & \dots & \mathcal{Z}_N \end{bmatrix} \tag{3.15a}$$

$$Y = \begin{bmatrix} y_{p+1} & \dots & y_N \end{bmatrix} \tag{3.15b}$$

Notice that the width of $\mathcal{Z}$ is $N-p$ because every $\mathcal{Z}_k$ involves samples from $k-p$ to $k-1$. Thus when using all samples, this yields $N-p$ many $\mathcal{Z}_k$. Also, assume for persistence of excitation that $[\mu_1 \ \dots \ \mu_{N-p+1}]$ has rank m and $N-p+1 > m$. With these definitions, the matrix $C\mathcal{K}$ is estimated in the first regression step, for example by solving:

$$\min_{C\mathcal{K}} ||Y - C\mathcal{K}\mathcal{Z}||_F^2 \tag{3.16}$$

In this paper we are mainly interested in the first regression step as it suffers from the 'curse-of-dimensionality'. Afterwards, this estimate is used to form a rank-revealing matrix. This allows choosing a model order with the assistance of a Singular Value Decomposition (SVD). Then the state sequence is reconstructed. With this state sequence the state-space matrices can be readily estimated, see for example van Wingerden and Verhaegen (2009).

The size of $C\mathcal{K}$ does require special care. In the method of van Wingerden and Verhaegen (2009), a dual problem is used to reduce the parameter count in the first regression step. Using Verdult and Verhaegen (2005), it is shown that the minimum norm solution of (3.16) is equal to the solution of

$$\min_{\alpha} ||\alpha||_F^2 \text{ with } Y - \alpha\mathcal{Z}^T\mathcal{Z} = 0 \tag{3.17}$$

where α are the Lagrange multipliers or dual parameters, provided $\mathcal{Z}$ has full column rank. Notice that $\mathcal{Z} \in \mathbb{R}^{q \times N-p}$ is generally tall and its columns are $\mathcal{Z}_k$ for different k, so it having full column rank is a weak assumption. This equivalence motivates the assumption that:

$$C\mathcal{K} = \alpha \mathcal{Z}^T \tag{3.18}$$

In this paper we will refer to expressions with the LPV sub-Markov parameters as the primal form and to expressions with the Lagrange multipliers α as the dual form. The benefit of this dual approach is as follows. Firstly, the dual parameters α and the related data matrix $\mathcal{Z}^T\mathcal{Z}$ are both 'curse-of-dimensionality' free in size. Secondly, $\mathcal{Z}^T\mathcal{Z}$ can be computed 'curse-of-dimensionality' free in memory and computation. In fact, the entire method of van Wingerden and Verhaegen (2009) can be performed 'curse-of-dimensionality' free in memory and computation. For completeness, we present the data equation:

$$Y \approx \alpha \mathcal{Z}^T \mathcal{Z} + E, \tag{3.19}$$

This ill-condition problem has been tackled using Tikhonov regularization (van Wingerden and Verhaegen, 2009) and matrix nuclear norms (Gebraad et al., 2011b) in literature.

In the next subsection we introduce several tensor-related definitions.

3.2.2. GENERAL TENSOR-RELATED DEFINITIONS

In this subsection some general tensor-related definitions are presented. These definitions are needed in the next section in order to present a tensor perspective on the LPV subspace identification problem.

Firstly, we formally define a tensor and its sizes.

Definition 3.2.1 *Consider a tensor* $\mathbf{T}$. *Let D be the number of dimensions. Denote the size of the i-th dimension as* J_i. *Then the tensor is in* $\mathbb{R}^{J_1 \times \cdots \times J_D}$. *Or in other words, the size of the tensor is* J_1*-by-…-by-*J_D *and contains real values. In this paper we use bold upper case characters to denote tensors.*

Secondly, we define how we access entries from vectors and matrices.

Definition 3.2.2 *Consider a matrix M and a vector v. Define* $[M]_{i,j}$ *as the entry of M at row i and column j. Let* $[M]_{:,j}$ *and* $[M]_{i,:}$ *respectively be the j-th column and i-th row vector. For a two-by-two matrix M this means:*

$$M = \begin{bmatrix} [M]_{1,1} & [M]_{1,2} \\ [M]_{2,1} & [M]_{2,2} \end{bmatrix} = \begin{bmatrix} [M]_{1,:} \\ [M]_{2,:} \end{bmatrix} = \begin{bmatrix} [M]_{:,1} & [M]_{:,2} \end{bmatrix} \tag{3.20}$$

For both row and column vectors, define $[v]_i$ *as the i-th entry of v.*

In the next subsection tensor nuclear norms are reviewed.

3.2.3. THE MATRIX NUCLEAR NORM AND THE TENSOR NUCLEAR NORM

In this subsection the matrix nuclear norm (Recht et al., 2010) and the tensor nuclear norm (Signoretto et al., 2010) are reviewed. They can be used to exploit a priori knowledge on low-rank properties of matrices and matricizations of tensors. These norms will play a key role in the method proposed in this paper.

First we review the matrix nuclear norm. The matrix nuclear norm has received considerable attention in the literature (Recht et al., 2010; Verhaegen and Hansson, 2014; Yu and Verhaegen, 2015). It can be used in a regularization term in order to exploit low-rank properties in a convex manner (Recht et al., 2010). The matrix nuclear norm of a matrix is the largest convex lower bound of the rank of that matrix (Recht et al., 2010) as:

$$||M||_* \leq \text{rank}(M),\ ||M||_2 \leq 1, \tag{3.21}$$

where $||M||_*$ is the nuclear norm of M, and M has been normalized. The matrix nuclear norm itself is defined as the sum of singular values.

Next we review the tensor nuclear norm. The tensor nuclear norm of a tensor relates to the multi-linear rank of a tensor (Signoretto et al., 2010). The multi-linear rank of a tensor is a tuple of numbers. Each number is the matrix rank of a different matricization of the tensor. We formally define the n-mode matricization of a tensor as follows:

Definition 3.2.3 *Consider a tensor* $\mathbf{T}$ *(Definition 3.2.1) of size* $\mathbb{R}^{J_1 \times J_2 \times \cdots \times J_d}$. *This tensor is of* d*-th order. Let* n *be a integer* $\in \{1, \ldots, d\}$ *and represent the mode number at hand. Then the* n*-mode matricization of* $\mathbf{T}$ *is denoted* $\mathbf{T}_{<n>}$ *and can be constructed as follows. First rearrange the dimensions of the tensor in the ordering* $[n, n+1, \ldots, d, 1, 2, \ldots, n-1]$. *The entry of* $\mathbf{T}$ *at position* $(i_1, i_2, \ldots, i_D)$ *is now put on position* $(i_n, i_{n+1}, \ldots, i_{n-1})$. *Then reshape the result into a matrix with* J_n *rows. The resulting matrix is the* n*-mode matricization or tensor unfolding (with forward cycling). This* n*-mode matricization can be transformed back into the original tensor by performing the two operations in reverse.*

Define also the n-mode product:

Definition 3.2.4 *The* n*-mode product of a tensor* $\mathbf{T}$ *with a matrix* M *is* $\mathbf{T} \bullet_n M$ *and can be computed using:*

$$[\mathbf{T} \bullet_n M]_{<n>} = M\mathbf{T}_{<n>}, \tag{3.22}$$

where the right hand side requires a matricization and a matrix multiplication. Returning to $\mathbf{T} \bullet_n M$ *can be done using Definition 3.2.3. This and other tensor operations can also be computed using the TensorLab toolbox (Vervliet et al., 2016).*

Also define:

Definition 3.2.5 *The* n*-rank of a tensor is defined as the rank of the* n*-mode matricization of that tensor:*

$$rank_n(\mathbf{T}) = rank(\mathbf{T}_{<n>}) \tag{3.23}$$

Now it is possible to define the multi-linear rank explicitly:

Definition 3.2.6 *The multi-linear rank of a tensor is the* d*-tuple of all* n*-ranks of that tensor.*

The multi-linear rank notion is computationally attractive because it is a tuple of matrix ranks. For contrast, the polyadic rank is NP-hard to determine (Håstad, 1990). This multi-linear rank relates to the tensor nuclear norm as defined in Signoretto et al. (2010). In this paper we use the definition of Signoretto et al. (2010) and not the different one

of Friedland and Lim (2014), because the former is computationally much cheaper to compute. Both have received considerable attention in the literature (He et al., 2005; Vasilescu and Terzopoulos, 2002; Friedland and Lim, 2014). The tensor nuclear norm is defined as:

$$||\mathbf{T}||_* = \frac{1}{d}\sum_{n=1}^{d} ||\mathbf{T}_{<n>}||_*, \tag{3.24}$$

and has the property:

$$||\mathbf{T}||_* \leq \frac{1}{d}\sum_{n=1}^{d} \text{rank}_n(\mathbf{T}),\ ||\mathbf{T}_{<n>}||_2 \leq 1\ \forall n, \tag{3.25}$$

however it is not proven to be the best convex heuristic. Just like the matrix nuclear norm, the tensor nuclear norm can be used as a regularization term.

In the next subsection the multi-linear SVD of a tensor is reviewed.

3.2.4. MULTI-LINEAR SINGULAR VALUE DECOMPOSITION (MLSVD)

In this subsection we review the SVD for matrices and the MLSVD for tensors, in order to be able to prove the multi-linear low-rank property of the parameter tensors in the next section.

First we present the more simple matrix SVD for illustration. The SVD (Golub et al., 1979) of a real matrix can be seen as a decomposition with special properties:

$$M = U\Sigma V^T, \tag{3.26}$$

where U and V are unitary and Σ contains the (non-negative) singular values of M along its diagonal in descending order. What is interesting for this paper is that the matrix M is low-rank if and only if some singular values are zero. Suppose only $\bar{r}$ singular values are non-zero, such that the rank is $\bar{r}$. This allows truncating without error to:

$$M = U\Sigma V^T = \bar{U}\bar{\Sigma}\bar{V}^T, \tag{3.27}$$

where $\bar{U}$ and $\bar{V}$ are the first $\bar{r}$ columns of U and V and $\bar{\Sigma}$ is the top-left $\bar{r}$-by-$\bar{r}$ sub-matrix of Σ. Notice that $\bar{\Sigma}$ is smaller than M for low-rank M.

The MLSVD (De Lathauwer et al., 2000) can be seen as an extension of the SVD to tensors. It decomposes a tensor as follows:

$$\mathbf{T} = \mathcal{S} \bullet_1 U^{(1)} \bullet_2 U^{(2)} \bullet_3 \cdots \bullet_D U^{(D)}, \tag{3.28}$$

where the n-mode product $\bullet_n$ is defined in Definition 3.2.4, $\mathcal{S}$ is the all-orthogonal core tensor with ordered multi-linear singular values and the matrices $U^{(*)}$ have orthonormal columns. The conditions for multi-linear low-rankness of a tensor are:

Definition 3.2.7 *A tensor is multi-linear low-rank if and only if all its n-mode matricizations are low-rank.*

or alternatively:

Definition 3.2.8 *A tensor is multi-linear low-rank if the core tensor of its MLSVD* (3.28) *is smaller than the full tensor in all dimensions. More specifically, the multi-linear rank of a tensor is the minimal size of its core tensor.*

In fact, there is no need for an MLSVD to prove multi-linear low-rankness. A *multi-linear decomposition* is sufficient to prove some tensor to be multi-linear low-rank. This decomposition is defined as:

$$\mathbf{T} = \mathcal{A} \bullet_1 M^{(1)} \bullet_2 M^{(2)} \bullet_3 \cdots \bullet_D M^{(D)}, \tag{3.29}$$

where $\mathcal{A}$ is a D-th order tensor and M are matrices with corresponding sizes. With this decomposition the multi-linear rank of the tensor is entry-wise bounded from above by the size of $\mathcal{A}$. This is because $\mathcal{A}$ and $M^{(*)}$ can be normalized to obtain an MLSVD with the same sizes (Vervliet et al., 2016). In the next section it will be proven that the parameter tensor is multi-linear low-rank, by providing a multi-linear decomposition with a non-trivially sized $\mathcal{A}$. But first, in the next subsection tensor trains are reviewed for later use in the numeric efficiency results.

3.2.5. TENSOR TRAINS AND NETWORKS

In this subsection we review the tensor train framework (Oseledets, 2011), because tensor trains will be used for presenting the data tensors and for the numeric efficiency results.

Firstly, we define a tensor and how we access any of its elements:

Definition 3.2.9 *Consider a tensor* $\mathbf{T}$ *(Definition 3.2.1) with D dimensions. Let* i_1 *to* i_D *be the D indices of the tensor, which each correspond to one dimension. Then* $\mathbf{T}(i_1, \ldots, i_D)$ *is its single element at position* $i_1, \ldots, i_D$*. Furthermore, the symbol ':' is used when multiple elements are involved. More specifically, ':' indicates that an index is not fixed. For example,* $\mathbf{T}(:, \ldots, :) = \mathbf{T}$ *and* $\mathbf{T}(:, :, j_3, \ldots, j_d) \in \mathbb{R}^{J_1 \times J_2}$ *is a matrix obtained by fixing the indices* j_3 *to* j_d*. This also applies to matrices (two-dimensional tensors).*

Before discussing the use of tensor trains, we first formally define tensor trains. Tensor trains are a way to condensely present a tensor. They can be seen as a decomposition. A tensor train consists of D 'cores' $\mathcal{A}^{(1)}$ to $\mathcal{A}^{(D)}$. Each core is a three-dimensional tensor. The relation between the tensor and the cores of its tensor train are defined element-wise as:

$$\mathbf{T}(i_1, \ldots, i_D) = \mathcal{A}^{(1)}(:, i_1, :)\mathcal{A}^{(2)}(:, i_2, :) \ldots \mathcal{A}^{(d)}(:, i_D, :), \tag{3.30}$$

where the left hand side is defined in Definition 3.2.9. Every element of the tensor is the product of several matrices. Notice that because the left hand side is a scalar, the first term in the product is a row vector and the last term is a column vector. For brevity of notation, also define the *generator* operator $g(*)$ as:

$$\mathbf{T} = g(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(D)}), \tag{3.31}$$

From equation (3.30) it is clear that the second index of the cores relates to indices of the original tensor. The remaining first and third indices of the (three-dimensional) cores are

named the *tensor train ranks*. These tensor train ranks are relevant to the effectiveness of tensor train decompositions.

Regarding the use of tensor trains, they can be used to condensely represent tensors and also to reduce computation costs (Oseledets, 2011). Firstly, instead of storing the original tensor element-by-element, its tensor train cores can be stored. Depending on the tensor train ranks, this can decrease memory costs. Also notice that these two ways of storage scale differently with the number of dimensions. The former way scales exponentially, while the latter scales linearly (since only new cores would be added). Secondly, tensor trains allow performing many operations on the original tensor without explicitly constructing it in memory and at lower computational cost. Namely, these operations can be directly applied on the tensor trains representing the original tensor. For example, the inner product of two tensors can be computed directly from their tensor trains. Depending on the tensor train ranks, this may come at lower computational cost, as again the scaling of the cost is different. In short, tensor trains can be used to break exponential scaling and thus 'curse-of-dimensionality'.

In this paper, this property will mainly be exploited through the (tensor train) inner product:

Definition 3.2.10 *The inner product of two tensors is the sum of their element-wise multiplication (Oseledets, 2011). This is well defined if the tensors have the same size. If the two tensors have a tensor train representation, then this operation can be performed directly using those representations (Oseledets, 2011). Then this operation becomes 'curse-of-dimensionality' free in memory and computational cost*[1].

In the next section the parameter tensors are discussed.

3.3. MULTI-LINEAR LOW-RANK PARAMETER TENSORS

In this section we show that the LPV sub-Markov parameters can be organized into several multi-linear low-rank tensors. That is, all their matricizations are low-rank. This property is proven using the multi-linear decompositions discussed in Subsection 3.2.4. This low-rank property will be exploited by the novel method in the next section.

First we present a three-dimensional example to illustrate the parameter tensors for a simple case. Consider an LPV state-space system (3.3) with $m = 2$. Let this system be Single Input Single Output (SISO) output-error with LTI input matrix. Furthermore we choose the size of the parameter tensor to be 3-by-3-by-3, such that it contains all LPV sub-Markov parameters to be estimated for p = 4. Then, the parameter tensor for this example system is as shown in Fig. 3.1.

Notice that the parameter tensor shown in Fig. 3.1 contains the LPV sub-Markov parameters and has some similarities with Hankel matrices. For example, the front slice is exactly a Hankel matrix. Also, the LTI variant would be exactly a Hankel tensor. More specifically, this tensor has some block-Hankel tensor structure. Later in this section we will show that we can make statements about its multi-linear ranks.

Next we present the parameter tensors in multi-linear decomposition form for the general case. This decomposition will also prove (an upper bound on) the multi-linear

[1] In detail, the computational cost scales linearly with the dimension count and dimension sizes and as a third power of the tensor train ranks.

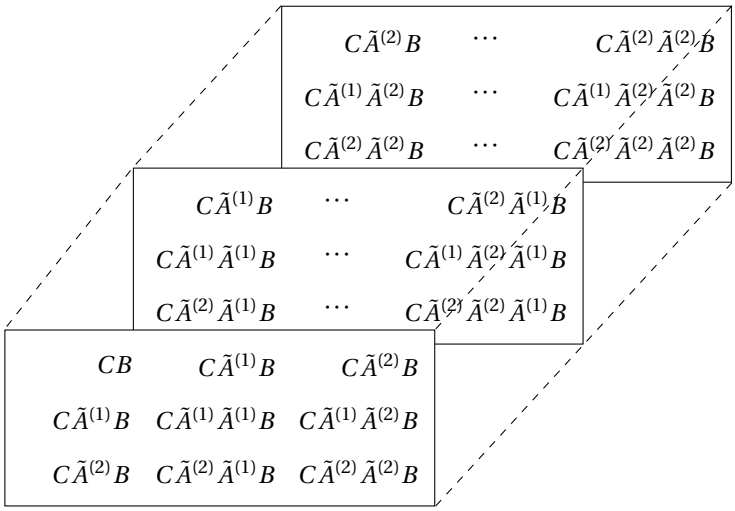


Figure 3.1: This figure shows the parameter tensor for the example system of Section 3.3.

rank (Subsection 3.2.4). The parameter tensors are denoted $\mathbf{H}_{o,\kappa}$ and are defined per output o and column of $\bar{B}(\kappa)$. The dimensions of a parameter tensor are all equal. Let the scalar t be this dimension (value), and D the number of dimensions. Both variables are user-chosen and will be defined explicitly later in this section. The relation of the two variables with the past window is discussed at the end of this section. First we present their multi-linear decomposition:

$$\mathbf{H}_{o,\kappa} = \mathcal{A}_H \bullet_1 F_o \bullet_2 F \bullet_3 \cdots \bullet_{D-1} F \bullet_D F_\kappa \in \mathbb{R}^{t \times \cdots \times t}, \tag{3.32}$$

where the tensor $\mathcal{A}_H$ and matrices F_o, F and F_κ will be defined next. The matrix F_o contains the output matrix and products of $\tilde{A}^{(*)}$ and is:

$$F_o = \begin{bmatrix} [C]_{o,:} \\ [C]_{o,:}\tilde{A}^{(1)} \\ \vdots \\ [C]_{o,:}\tilde{A}^{(m)} \\ [C]_{o,:}\tilde{A}^{(1)}\tilde{A}^{(1)} \\ [C]_{o,:}\tilde{A}^{(2)}\tilde{A}^{(1)} \\ \vdots \\ [C]_{o,:}\tilde{A}^{(m)}\tilde{A}^{(m)} \\ \vdots \end{bmatrix} \in \mathbb{R}^{t \times \hat{n}}, \tag{3.33}$$

where we let t be the height of this matrix, and the brackets with subscript are defined in Definition 3.2.2. Notice that this is by definition also the size of the first dimension of $\mathbf{H}_{o,\kappa}$. The matrix F contains products of $\tilde{A}^{(*)}$ in vectorized form as:

$$F = \begin{bmatrix} \text{vec}(I_{\hat{n}})^T \\ \text{vec}(\tilde{A}^{(1)})^T \\ \vdots \\ \text{vec}(\tilde{A}^{(m)})^T \\ \text{vec}(\tilde{A}^{(1)}\tilde{A}^{(1)})^T \\ \text{vec}(\tilde{A}^{(2)}\tilde{A}^{(1)})^T \\ \vdots \\ \text{vec}(\tilde{A}^{(m)}\tilde{A}^{(m)})^T \\ \vdots \end{bmatrix} \in \mathbb{R}^{t \times \hat{n}^2} \tag{3.34}$$

Notice that the width of F is $\hat{n}^2$, but this will be dealt with in the definition of $\mathcal{A}_H$. The matrix F_κ contains the matrix $\bar{B}$ and is defined as:

$$F_\kappa = \begin{bmatrix} ([\bar{B}]_{:,\kappa})^T \\ (\tilde{A}^{(1)}[\bar{B}]_{:,\kappa})^T \\ \vdots \\ (\tilde{A}^{(m)}[\bar{B}]_{:,\kappa})^T \\ (\tilde{A}^{(1)}\tilde{A}^{(1)}[\bar{B}]_{:,\kappa})^T \\ (\tilde{A}^{(2)}\tilde{A}^{(1)}[\bar{B}]_{:,\kappa})^T \\ \vdots \\ (\tilde{A}^{(m)}\tilde{A}^{(m)}[\bar{B}]_{:,\kappa})^T \\ \vdots \end{bmatrix} \in \mathbb{R}^{t \times \hat{n}} \tag{3.35}$$

These three matrices and the tensor $\mathcal{A}_H$ together form the LPV sub-Markov parameters. This tensor $\mathcal{A}_H$ consists purely of ones and zeros, and discards all inadmissible products between elements of the matrices F_*. These inadmissible products appear because instead of matrix products we have products of the vectorizations of those matrices. Therefore this tensor is named here as the admissibility tensor. We present the algorithm to generate $\mathcal{A}_H$ as a function of $\hat{n}$ and D in Appendix 3.A. Proof of the decomposition follows through straightforward computations. The size of this tensor $\mathcal{A}_H$ matches with the widths of F_* as shown in (3.32) and is $\begin{bmatrix} \hat{n} & \hat{n}^2 & \hat{n}^2 & \dots & \hat{n}^2 & \hat{n} \end{bmatrix}$. Using the definition of the multi-linear decomposition (Section 3.2.4), we can state that this size is the upper bound of the multi-linear rank of $\mathbf{H}_{o,\kappa}$.

Now that we know the multi-linear rank of $\mathbf{H}_{o,\kappa}$, we can state under which conditions it is multi-linear *low*-rank. Using Definition 3.2.8, this is the case if $\mathcal{A}_H$ is smaller than $\mathbf{H}_{o,\kappa}$ in all dimensions. Filling in their sizes gives:

Theorem 3.3.1 *The parameter tensor $\mathbf{H}_{o,\kappa}$ is multi-linear low-rank if $t > \hat{n}^2$.*

Since t is user-chosen, we will implicitly assume throughout this paper that $t > \hat{n}^2$.

In this paragraph we give details on the size of $\mathbf{H}_{o,\kappa}$ for completeness. The dimension t is:

$$t = \sum_{j=0}^{\bar{f}-1} m^j, \tag{3.36}$$

where $\bar{f}$ is named the incremental window. The maximum number of $\tilde{A}^{(*)}$ in sequence in F_* is exactly $\bar{f}-1$. It is interesting to note that for fixed $\bar{f}=2$, the full tensors are similar to the ones in our previous work (Gunes et al., 2017b). Furthermore, the number of dimensions D of $\mathbf{H}_{o,\kappa}$ is also user-chosen. Together they determine the size of $\mathbf{H}_{o,\kappa}$. For purpose of simple notation, we omit D and $\bar{f}$ from the notation of $\mathbf{H}_{o,\kappa}$ and other affected variables. These two also determine which LPV sub-Markov parameters will appear in $\mathbf{H}_{o,\kappa}$. Let the 'order' of an LPV sub-Markov parameter be the number of $\tilde{A}^{(*)}$ in sequence that it has plus one. Then the highest order of LPV sub-Markov parameters in $\mathbf{H}_{o,\kappa}$ is equal to:

$$h = 1 + (\bar{f}-1)D, \tag{3.37}$$

where h is named the regularization window. A natural choice would be $h = p$. However, we keep the option of choosing h open. In the remainder of this paper we let $h \leq p$. The availability of the choice of h will allow reducing computational load where needed.

Before presenting the novel method, first the data tensor is presented.

3.4. DATA TENSORS

In this section we present the data tensors and their tensor train decompositions, because they will be used in the numeric efficiency results in Section 3.5. Additionally we present the predictor-based output equation (3.14) using tensors for illustration purposes.

The data tensors will be defined using their tensor train decompositions. Therefore, these decompositions are presented first. For this purpose, define their tensor train ranks:

$$\begin{aligned} R &= \begin{bmatrix} 1 & 1 & \bar{f} & 2\bar{f}-1 & \dots & 1+(\bar{f}-1)D & 1 \end{bmatrix} \\ R_s &= R(s) \end{aligned} \tag{3.38}$$

Also, similar to the parameter tensors, the dimensions of the data tensors are each equal to t. This makes the size of every core R_s-by-t-by-R_{s+1}. Notice that these cores do not suffer from the 'curse-of-dimensionality'. Before presenting the cores, first some defini-

tions are needed. Define the zero matrices:

$$0_L(b) = \text{zeros}(R_s, \bar{O}(b) - 1) \tag{3.39a}$$

$$0_R(b) = \text{zeros}(R_s, \bar{f} - \bar{O}(b)) \tag{3.39b}$$

$$\bar{O} = \begin{bmatrix} \text{ones}(m^0, 1)1 \\ \text{ones}(m^1, 1)2 \\ \vdots \\ \text{ones}(m^{\bar{f}-1}, 1)\bar{f} \end{bmatrix} \tag{3.39c}$$

Also, define the matrices: $\bar{P}$:

$$\bar{P}^{(s)} = \begin{bmatrix} P_0^{(s)} \\ \vdots \\ P_{\bar{f}-1}^{(s)} \end{bmatrix} \in \mathbb{R}^{t \times R_s}, \tag{3.40}$$

where

$$\begin{aligned} P_0^{(s)} &= \begin{bmatrix} 1 & \dots & 1 \end{bmatrix} \in \mathbb{R}^{1 \times R_s} \\ P_1^{(s)} &= \begin{bmatrix} \mu_{k-1} & \dots & \mu_{k-R_s} \end{bmatrix} \in \mathbb{R}^{m \times R_s} \\ P_2^{(s)} &= \begin{bmatrix} \mu_{k-2} \otimes \mu_{k-1} & \dots & \mu_{k-R_s-1} \otimes \mu_{k-R_s} \end{bmatrix} \\ &\quad \in \mathbb{R}^{m^2 \times R_s} \\ P_{\bar{f}-1}^{(s)} &= \begin{bmatrix} (\mu_{k-(\bar{f}-1)} \otimes \cdots \otimes \mu_{k-1})^T \\ \vdots \\ (\mu_{k-(\bar{f}-1)-R_s+1} \otimes \cdots \otimes \mu_{k-R_s})^T \end{bmatrix}^T \\ &\quad \in \mathbb{R}^{m^{\bar{f}-1} \times R_s} \end{aligned} \tag{3.41}$$

Furthermore, define diag() as an operator with vector argument, which returns a diagonal matrix whose diagonal entries are the entries of its vector argument in the same order. With these definitions, the cores can be presented. In order to be consistent with previous work (Gunes et al., 2017b), we use a tensor train decomposition which has two extra cores. These cores are just vectors and therefore do not increase the number of dimensions of the data tensors. As a result, the tensor train decompositions have $D+2$ cores. Using the previous definitions, cores number 2 to $D+1$ can be presented using matrix slices as:

$$\begin{aligned} \mathcal{B}_k^{(s)}(:, b, :) &= \begin{bmatrix} 0_L(b) & \text{diag}(\bar{P}^{(s)}(b, :)) & 0_R(b) \end{bmatrix} \\ &\quad \in \mathbb{R}^{R_s \times R_{s+1}} \\ &\quad \forall b \in \{1, \dots, t\}, \forall s \in \{2, \dots, D+1\} \end{aligned} \tag{3.42}$$

Notice that these cores have a shift and a diagonal structure, which is also sparse. The shift structure appears as the zero matrices $0_L(b)$ and $0_R(b)$ change in width for b. The

diagonal structure appears due to the operator diag(∗). Together, this also results in sparsity. This structure appears, because the scheduling sequence samples always appear in sequence in the data (3.11b). Next, we define the two extra cores. They are the first and last cores. The first core is simply a unit scalar:

$$\mathcal{B}^{(1)} = 1, \tag{3.43}$$

and the last core is a vector:

$$\mathcal{B}^{(D+2)}_{k,\kappa} = \begin{bmatrix} \mu^T_{k-1} \otimes z^T_{k-1} \\ \vdots \\ \mu^T_{k-h} \otimes z^T_{k-h} \end{bmatrix} (:,\kappa) \in \mathbb{R}^h, \tag{3.44}$$

where $\mu^T_{k-j} \otimes z^T_{k-j}$ is a row vector of which we take its κ-th element. Similarly define another variant:

$$\bar{\mathcal{B}}^{(D+2)}_{k,\kappa} = \begin{bmatrix} \mu^T_{k-1} \otimes z^T_{k-1}/\bar{s}(1) \\ \vdots \\ \mu^T_{k-h} \otimes z^T_{k-h}/\bar{s}(h) \end{bmatrix} (:,\kappa) \in \mathbb{R}^h, \tag{3.45}$$

where $\bar{s}$ is defined in Algorithm 3.B. With these definitions, we can define the data tensors using their tensor train decomposition (Subsection 3.2.5). Define for use in the numerical efficiency results in Section 3.5, the unweighed data tensor:

$$\mathbf{Z}_{k,\kappa} = g(\mathcal{B}^{(1)}, \mathcal{B}^{(2)}_k, \ldots, \mathcal{B}^{(D+1)}_k, \mathcal{B}^{(D+2)}_{k,\kappa}) \tag{3.46}$$

Additionally, define (for use in the tensor variant of the predictor-based output equation):

$$\bar{\mathbf{Z}}_{k,\kappa} = g(\mathcal{B}^{(1)}, \mathcal{B}^{(2)}_k, \ldots, \mathcal{B}^{(D+1)}_k, \bar{\mathcal{B}}^{(D+2)}_{k,\kappa}), \tag{3.47}$$

which only differs in the last core. In this paper we use these two variants, in order simplify the respective equations in which they are used. These tensor train decompositions have been generalized from results previously presented in Gunes et al. (2017b) (for $\bar{f} = 2$). Proof of the tensor trains follows through straightforward computations. Now the data tensors have been defined.

Finally, we illustrate the role of the data tensor with the following output equation:

$$\begin{aligned} [y_k]_o \approx [C]_{o,:}\mathcal{K}\mathcal{Z}_k + [e_k]_o = \\ \sum_{\kappa=1}^{m(l+r)} < \mathbf{H}_{o,\kappa}, \bar{\mathbf{Z}}_{k,\kappa} > + [e_k]_o, \end{aligned} \tag{3.48}$$

where $< *, * >$ is the inner product (Definition 3.2.10). Notice that the top line is the classic predictor-based output equation (3.14) and the bottom line is its tensor form. This equation will play a key role in the next section.

In the next section we present how the tensor nuclear norm of the parameter tensors can be computed without 'curse-of-dimensionality'.

3.5. EFFICIENT COMPUTATION OF THE PARAMETER TENSOR NUCLEAR NORM

In this section we present how to compute the tensor nuclear norms of the parameter tensors 'curse-of-dimensionality' free. This is non-trivial, because the parameter tensors contain the LPV sub-Markov parameters which suffer from the 'curse-of-dimensionality'. We present three successive contributions to reach the goal. Firstly, we show how the parameter tensors can be expressed in the dual problem presented in Subsection 3.2.1. Secondly, we present how the arguments of the tensor nuclear norms at hand can be replaced by 'curse-of-dimensionality' free arguments while producing the exact same (norm) result. Thirdly, we show how these norm results can be computed 'curse-of-dimensionality' free using tensor trains. These contributions allow for efficiency of the method proposed in the next section.

First we show how the parameter tensors can be expressed in the dual problem presented in Subsection 3.2.1. One difficulty is that we have tensors instead of matrices. Therefore we first rewrite the dual form of $C\mathcal{K}$ matrix (containing the LPV sub-Markov parameters) in a form suitable for tensors as:

$$C\mathcal{K} = \alpha \mathcal{Z}^T = \sum_{k=p+1}^{N} \alpha_{:,k-p} \mathcal{Z}_k^T, \tag{3.49}$$

From this starting point, $C\mathcal{K}$ (or the LPV sub-Markov parameters) can be used to form the parameter tensors. A similar approach can be used to derive the dual form of the parameter tensors. This results in:

$$\mathbf{H}_{o,\kappa} = \sum_{k=p+1}^{N} \alpha_{o,k-p} \mathbf{Z}_{k,\kappa} \tag{3.50}$$

where $\mathbf{H}_{o,\kappa}$ and $\mathbf{Z}_{k,\kappa}$ are defined in (3.32) and (3.46), respectively. Notice that this equation is a direct result of equation (3.49). This returns the dual description of the parameter tensors. Notice that this step has not changed the (huge) sizes. These tensors will be the arguments of tensor nuclear norms.

Secondly, we address the size of the arguments of the nuclear norms at hand. Recall that the tensor nuclear norm is computed as the average matrix nuclear norm of all the matricizations (3.24). Therefore we work out the matricizations and their norms. Additionally we add the averaging factor $\frac{1}{D}$ of the norm in advance in order to simplify the objective functions in Section 3.6. The (scaled) matricizations are:

$$\begin{aligned} \frac{1}{D}\mathbf{H}_{o,\kappa,<n>} &= \frac{1}{D}\Big[\sum_{k=p+1}^{N} \alpha_{o,k-p}\mathbf{Z}_{k,\kappa}\Big]_{<n>} \\ &= \frac{1}{D}\sum_{k=p+1}^{N} \alpha_{o,k-p}\big[\mathbf{Z}_{k,\kappa}\big]_{<n>} \in \mathbb{R}^{t \times t^{D-1}} \end{aligned} \tag{3.51}$$

where n is the mode number. Notice that the reformulation of (3.49) allowed easy extraction of the dual parameters α, which will be important later in this paragraph. The

matrix nuclear norm of (3.51) is:

$$\frac{1}{D}||\mathbf{H}_{o,\kappa,<n>}||_* = \frac{1}{D}||\sum_{k=p+1}^{N} \alpha_{o,k-p}\left[\mathbf{Z}_{k,\kappa}\right]_{<n>}||_*, \tag{3.52}$$

The argument of this norm contains in dual form LPV sub-Markov parameters which suffer from the 'curse-of-dimensionality'. Therefore we propose replacing it by a different argument which is both 'curse-of-dimensionality' free in size and produces the exact same norm. Here we exploit the dual form and that the argument is a very wide matrix. For this purpose, rewrite:

$$\frac{1}{D}||\sum_{k=p+1}^{N} \alpha_{o,k-p}\left[\mathbf{Z}_{k,\kappa}\right]_{<n>}||_* = \frac{1}{D}\text{Trace}\sqrt{\sum_{k_i=p+1}^{N}\sum_{k_j=p+1}^{N} \alpha_{o,k_i-p}\alpha_{o,k_j-p}\left[\mathbf{Z}_{k_i,\kappa}\right]_{<n>}\left[\mathbf{Z}_{k_j,\kappa}\right]_{<n>}^{T}}, \tag{3.53}$$

such that the wide data matrix is multiplied with its transpose and the resulting matrix has the much smaller size t-by-t. We can also get rid of the summations. For that, define the following two matrices:

$$\mathbf{Z}_{\kappa,<n>} = \frac{1}{D}\begin{bmatrix} \left\{\left[\mathbf{Z}_{p+1,\kappa}\right]_{<n>}\right\}_{1,:} \\ \vdots \\ \left\{\left[\mathbf{Z}_{N,\kappa}\right]_{<n>}\right\}_{1,:} \\ \vdots \\ \left\{\left[\mathbf{Z}_{p+1,\kappa}\right]_{<n>}\right\}_{t,:} \\ \vdots \\ \left\{\left[\mathbf{Z}_{N,\kappa}\right]_{<n>}\right\}_{t,:} \end{bmatrix} \in \mathbb{R}^{t(N-p)\times t^{D-1}}, \tag{3.54}$$

whose width we will address in the next paragraph, and define the function $d_t(*)$ as:

$$d_t(\alpha_{o,:}) = \begin{bmatrix} \alpha_{o,:} & 0 & \cdots \\ 0 & \alpha_{o,:} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \in \mathbb{R}^{t\times t(N-p)} \tag{3.55}$$

Now it possible to rewrite (3.53) without summations as:

$$\text{Trace}\sqrt{d_t(\alpha_{o,:})\mathbf{Z}_{\kappa,<n>}\mathbf{Z}_{\kappa,<n>}^{T}d_t(\alpha_{o,:})^{T}}, \tag{3.56}$$

Finally, this equation can be cast back in normal tensor nuclear norm definition without trace as:

$$\frac{1}{D}||\mathbf{H}_{o,\kappa,<n>}||_* = ||d_t(\alpha_{o,:})\left[\mathbf{Z}_{\kappa,<n>}\mathbf{Z}^T_{\kappa,<n>}\right]^{1/2}||_*, \tag{3.57}$$

where the square root of a matrix is defined via the SVD (Signoretto et al., 2010). This nuclear norm is exactly equal to the one in (3.52), but has a argument which is 'curse-of-dimensionality' free in size.

Thirdly, we show how the nuclear norm presented in (3.57) can be computed 'curse-of-dimensionality' free using tensor trains. More specifically, we present how the matrix $\mathbf{Z}_{\kappa,<n>}\mathbf{Z}^T_{\kappa,<n>}$ can be computed 'curse-of-dimensionality' free. This is non-trivial, because $\mathbf{Z}_{\kappa,<n>}$ itself is not 'curse-of-dimensionality' free. We present a formula for $\mathbf{Z}_{\kappa,<n>}\mathbf{Z}^T_{\kappa,<n>}$ which directly shows how it can be computed 'curse-of-dimensionality' free. This formula is derived as follows.

Firstly, notice that the size of $\mathbf{Z}_{\kappa,<n>}\mathbf{Z}^T_{\kappa,<n>}$ is $t(N-p)$-by-$t(N-p)$ and thus 'curse-of-dimensionality' free in size, such that it can be computed entry-wise. Afterwards its square root can be readily obtained. So the problem boils down to computing the entries 'curse-of-dimensionality' free. A single entry of the product $\mathbf{Z}_{\kappa,<n>}\mathbf{Z}^T_{\kappa,<n>}$ using (3.54) is:

$$\left[\mathbf{Z}_{\kappa,<n>}\mathbf{Z}^T_{\kappa,<n>}\right]_{i,j} = \frac{1}{D^2}\left\{\left[\mathbf{Z}_{k_i,\kappa} \quad\right]_{<n>}\right\}_{\bar{t}_i,:}\left\{\left[\mathbf{Z}_{k_j,\kappa} \quad\right]_{<n>}\right\}^T_{\bar{t}_j,:}, \tag{3.58}$$

which is a scalar and where:

$$i = k_i - p + (\bar{t}_i - 1)(N-p) \tag{3.59a}$$

$$j = k_j - p + (\bar{t}_j - 1)(N-p) \tag{3.59b}$$

Notice that computing this single product in straight-forward manner would require storage of $2t^{D-1}$ elements. We wish to compute this product 'curse-of-dimensionality' free and will therefore use tensor trains (Subsection 3.2.5). In order to simplify derivations, we move to the inner product form (Definition 3.2.10). We use that the product (3.58) is a multiplication of a row vector with a column vector to obtain a scalar, to rewrite to:

$$\left[\mathbf{Z}_{\kappa,<n>}\mathbf{Z}^T_{\kappa,<n>}\right]_{i,j} = \frac{1}{D^2}\langle\left\{\left[\mathbf{Z}_{k_i,\kappa} \quad\right]_{<n>}\right\}_{\bar{t}_i,:}, \left\{\left[\mathbf{Z}_{k_j,\kappa}\right]_{<n>}\right\}_{\bar{t}_j,:}\rangle \tag{3.60}$$

The benefit of this inner product notation is that some properties are more clear. Notice that due to the definition of the inner product, any simultaneous reshape of the two vectors does not change the inner product result. The same applies to permutations and matricizations. So we can freely turn these two vectors into tensors. For numeric efficiency purposes, we are interested in tensors with suitable tensor train decompositions. We use the tensor train decomposition of $\mathbf{Z}_{k,\kappa}$ (3.46), to turn these vectors into tensor trains:

$$\mathbf{Z}^{(n,\bar{t})}_{k,\kappa} = g(\mathcal{B}^{(1)}, \mathcal{B}^{(2)}_k, \ldots, \mathcal{B}^{(n)}_k(:,\bar{t},:), \ldots, \mathcal{B}^{(D+1)}_k, \mathcal{B}^{(D+2)}_{k,\kappa}), \tag{3.61}$$

where the effect of $\bar{t}$ in (3.60) is accounted for by cutting the n-th core. Furthermore notice that the tensor trains for both vectors are cut at the same core number n due (3.60)

and thus have matching sizes. This allows rewriting the inner product into:

$$\left[\mathbf{Z}_{\kappa,<n>}\mathbf{Z}_{\kappa,<n>}^{T}\right]_{i,j} = \frac{1}{D^2} < \mathbf{Z}_{k_i,\kappa}^{(n,\bar{t}_i)}, \mathbf{Z}_{k_j,\kappa}^{(n,\bar{t}_j)} >, \tag{3.62}$$

where the left hand side equals the *inner product of two tensor trains*. We can perform this operation directly on the tensor trains 'curse-of-dimensionality' free in both memory and computation (Oseledets, 2011).

These results allow computing the tensor nuclear norm of the parameter tensors 'curse-of-dimensionality' free for the proposed method in the next section.

3

3.6. PREDICTOR-BASED TENSOR NUCLEAR NORM REGRESSION (PBTNNR)

Using the results from the previous sections we now have all the ingredients to derive the new Predictor Based Tensor Nuclear Norm Regression method (PBTNNR) algorithm. The proposed convex LPV subspace identification method does not suffer from the 'curse-of-dimensionality' in terms of memory or computational cost and uses its regularization to improve problem conditioning.

The difference between the proposed method and the regularized method of van Wingerden and Verhaegen (2009) and Gebraad et al. (2011a) is in their regularization term during the first regression step. This step suffers from the 'curse-of-dimensionality' and is therefore crucial. Therefore we only present the first regression step in this section and refer to Section 3.2.1 for details on subsequent steps. We first present our cost function using Section 3.3 in primal form for illustration, and afterwards directly in 'curse-of-dimensionality' free dual form using results from Section 3.5. The starting point is the objective function (van Wingerden and Verhaegen, 2009):

$$\min_\theta ||Y - [C\mathcal{K}](\theta)\mathcal{Z}||_F^2, \tag{3.63}$$

where $C\mathcal{K}$ and $\mathcal{Z}$ are defined and motivated in Subsection 3.2.1. The proposed method adds tensor nuclear norm regularization terms to this objective function. More specifically, the arguments of these norms are the parameter tensors defined in (3.32). For simplicity, all regularization terms are weighed in the objective function by a single tuning parameter λ. This results in the following objective function:

$$\min_\theta ||Y - [C\mathcal{K}](\theta)\mathcal{Z}||_F^2 + \lambda \sum_{o=1}^{l} \sum_{\kappa=1}^{m(l+r)} ||\mathbf{H}_{o,\kappa}(\theta)||_*, \tag{3.64}$$

where $\mathbf{H}_{o,\kappa}$ is defined in (3.32). Every scalar LPV sub-Markov parameter is parametrized by one optimization parameter. This applies to both $[C\mathcal{K}](\theta)$ and the parameters tensors $\mathbf{H}_{o,\kappa}(\theta)$. This also implies that some optimization parameters θ_i appear multiple times in $\mathbf{H}_{o,\kappa}(\theta)$. For purpose of illustration, we rewrite this equation using (3.48) using only $\mathbf{H}_{o,\kappa}(\theta)$ as:

$$\begin{aligned}\min_\theta \sum_{k,o} \left([y_k]_o - \sum_{\kappa=1}^{m(l+r)} < \mathbf{H}_{o,\kappa}(\theta), \bar{\mathbf{Z}}_{k,\kappa} > \right)^2 + \\ \lambda \sum_{o=1}^{l} \sum_{\kappa=1}^{m(l+r)} ||\mathbf{H}_{o,\kappa}(\theta)||_*,\end{aligned} \tag{3.65}$$

where we let $h = p$ (3.37) for ease of notation and $\bar{\mathbf{Z}}_{k,\kappa}$ is defined in (3.47). These objective functions are for illustration, and the proposed method uses the 'curse-of-dimensionality' free dual form derived in Section 3.5. This 'curse-of-dimensionality' free dual form of the objective function is:

$$\begin{aligned} &\min_{\alpha} ||Y - \alpha \mathcal{Z}^T \mathcal{Z}||_F^2 + \\ &\lambda \sum_{o=1}^{l} \sum_{\kappa=1}^{m(l+r)} \sum_{n=1}^{D} ||d_t(\alpha_{o,:}) \left[\mathbf{Z}_{\kappa,<n>} \mathbf{Z}_{\kappa,<n>}^T \right]^{1/2} ||_*, \end{aligned} \tag{3.66}$$

where the Lagrange multipliers α and the fit criterion are defined and motivated in Subsection 3.2.1, n is the mode number and the argument of the nuclear norm is derived in Section 3.5.

In the next section the simulation results are presented.

3.7. SIMULATION RESULTS

In this section simulation results are presented in order to compare the proposed method PBTNNR with the method of van Wingerden and Verhaegen (2009) in terms of variance and bias. Both methods can be run using the PBSID toolbox (van Wingerden and Verhaegen, 2009).

3.7.1. SIMULATION SETTINGS

In this subsection, the simulation settings and some related definitions are presented.

For statistical significance, the results presented in this paper are based on 100 Monte Carlo simulations. For every Monte Carlo simulation a different realization of both the input and the innovation vector is used. The scheduling sequence is kept the same. All methods use a model order equal to the system order, and are supplied the information that $\bar{D} = 0$ and that the output equation is LTI. For completeness, the future window variable for the SVD step is chosen equal to the past window p.

In this section we compare the novel method with the (Tikhonov) regularized LPV-PBSID$_{opt}$ (kernel) method of van Wingerden and Verhaegen (2009). The quality of the estimates is evaluated by investigating the Variance Accounted For (VAF) on a validation data set which is *different* from the one used for identification, in the sense that different realizations of both the input and the innovation vector are used. The VAF for single-output systems is defined as follows (van Wingerden and Verhaegen, 2009):

$$\text{VAF}(\bar{y}_k, \hat{y}_k) = \max\left\{1 - \frac{var(\bar{y}_k - \hat{y}_k)}{var(\bar{y}_k)}, 0\right\} 100\%$$

The noise-free simulated output of the system is used when evaluating the VAF, because this allows the VAF to reach 100% when the model is equal to the true system modulo global state-coordinate transformations. Notice that this is possible because the data is generated using simulations. The noise-free simulated output of the system is here denoted as $\bar{y}_k$. In similar sense the noise-free (simulated) model output is denoted as $\hat{y}_k$. The operator $var(*)$ denotes the variance of its argument. The tuning parameter λ of the proposed method has been chosen as follows. The λ with single significant

digit which yields the highest mean validation VAF is chosen. The computations are performed on an Intel i7 quad-core processor running at 2.7Ghz with 8 GB RAM. We provide the computation time both of the proposed method and the method of van Wingerden and Verhaegen (2009).

Several cases and their results are presented in the following subsections.

3.7.2. SIMULATION RESULTS CASE 1

In this subsection simulation results are presented for a case of Gunes et al. (2017a) in order to compare the proposed method with both the method of van Wingerden and Verhaegen (2009) and Gunes et al. (2017a) (PBTR).

This case uses the following LPV state-space system (3.3):

$$[A^{(1)}, A^{(2)}] = \left[\begin{array}{cc|cc} \frac{4}{15} & \frac{1}{15} & \frac{3}{20} & -\frac{1}{60} \\ -\frac{1}{6} & \frac{1}{30} & -\frac{1}{60} & \frac{3}{20} \end{array}\right],$$

$$[B^{(1)}, B^{(2)}] = \left[\begin{array}{c|c} 1 & 0.2 \\ 0 & 0.2 \end{array}\right], C = \begin{bmatrix} 1 & 0 \end{bmatrix},$$

and D is zero and K is LPV. The matrix $K^{(i)}$ for $i = \{1, \ldots, m\}$ is obtained from the Discrete Algebraic Ricatti Equation (DARE) with $A^{(i)}$, C and identity covariance of the concatenated process and measurement noise. Both the input vector u_k and the innovation vector e_k are white noise with unit power. The data size N is chosen as 200. Both methods are run with past window p equal to 6.

The system is evaluated at the scheduling sequence:

$$\mu_k^{(2)} = \cos(2\pi k \frac{20}{N})/2 + 0.2,$$

We present the VAF of the proposed method, the method of van Wingerden and Verhaegen (2009) and the non-convex refinement method of Gunes et al. (2017a) (PBTR) in Table 3.1, and a box-plot of the VAF in Fig. 3.2. We present the computation times for the convex methods. The average computation times are 6.6 seconds for the proposed method and 38 milliseconds for the method of van Wingerden and Verhaegen (2009) per simulation.

Method	VAF
LPV-PBSID$_{opt}$ (kernel)	96.6
PBTNNR (kernel, $\bar{f} = 2$, $D = 2$)	97.9 with $\lambda = 0.1$
PBTR (refinement method)	98.0

Table 3.1: Mean VAF for different methods for Case 1.

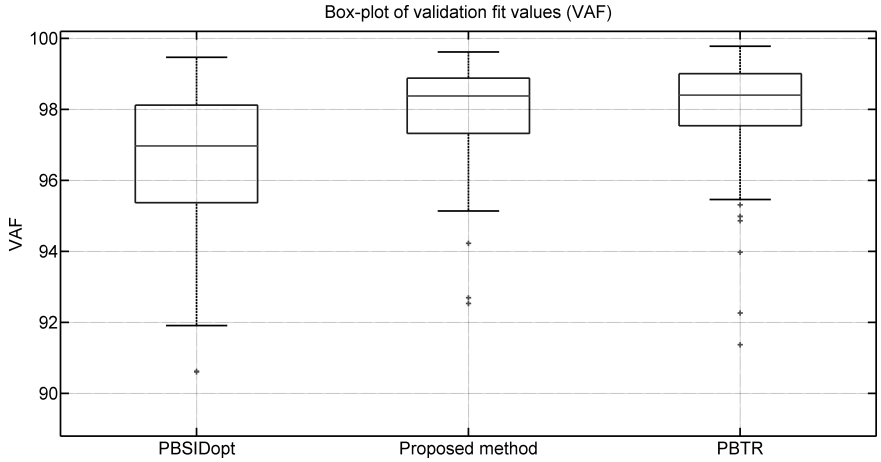


Figure 3.2: This figure shows a box-plot of the validation VAF of the 100 Monte Carlo simulations for three methods. Only the first two methods are convex; the last method is a non-convex refinement method.

From Table 3.1 and Fig. 3.2 it can be concluded that for this case the proposed method has higher performance than the method of van Wingerden and Verhaegen (2009) in terms of VAF. Additionally, the non-convex refinement method of Gunes et al. (2017a) has a higher VAF than the two convex methods.

3.7.3. SIMULATION RESULTS CASE 2

In this subsection simulation results are presented for a larger past window with a case which relates to the flapping dynamics of a wind turbine. This case has been used before in Felici et al. (2007); van Wingerden and Verhaegen (2009).

This case uses the following LPV state-space system (3.3):

$$[A^{(1)}, A^{(2)}] = \left[\begin{array}{cc|cc} 0 & 0.0734 & -0.0021 & 0 \\ -6.5229 & 0.4997 & -0.0138 & 0.5196 \end{array}\right],$$

$$[B^{(1)}, B^{(2)}] = \left[\begin{array}{c|c} -0.7221 & 0 \\ -9.6277 & 0 \end{array}\right], C = \begin{bmatrix} 1 & 0 \end{bmatrix},$$

and D is zero and K is LPV. The matrix $K^{(i)}$ for $i = \{1,\ldots,m\}$ is obtained from the Discrete Algebraic Ricatti Equation (DARE) with $A^{(i)}$, C and identity covariance of the concatenated process and measurement noise. The input vector u_k is white noise with unit power, the innovation vector e_k is also white noise and the signal-to-noise ratio is 40dB. The data size N is chosen as 100. Both methods are run with past window p equal to 15.

The system is evaluated at the scheduling sequence:

$$\mu_k^{(2)} = \cos(2\pi k \frac{20}{N})/2 + 0.2,$$

We present the VAF of both the novel method and the method of van Wingerden and Verhaegen (2009) in Table 3.2, and a box-plot of the VAF in Fig. 3.3. The average computation times are 23 seconds for the proposed method and 50 milliseconds for the method of van Wingerden and Verhaegen (2009) per Monte Carlo simulation.

Method	VAF
LPV-PBSID$_{opt}$ (kernel)	97.0
PBTNNR (kernel, $\bar{f} = 3$, $D = 3$)	98.1 with $\lambda = 0.01$

Table 3.2: Mean VAF for different methods for Case 2.

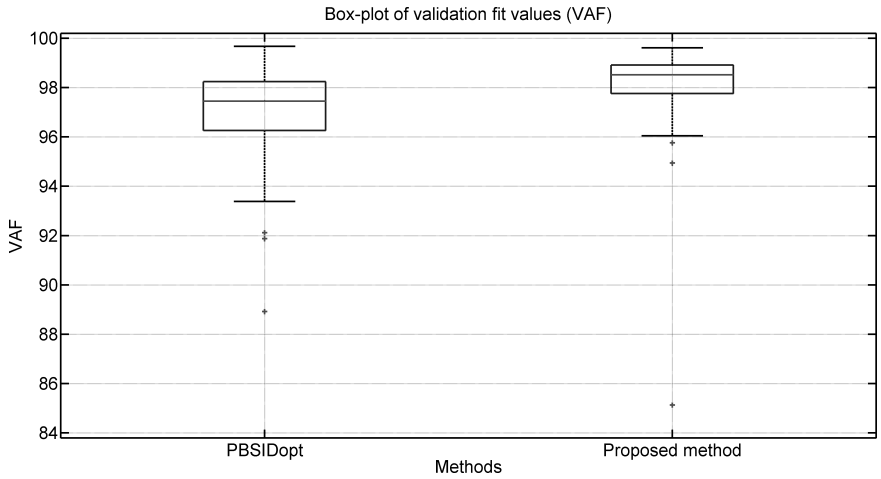


Figure 3.3: This figure shows a box-plot of the validation VAF of the 100 Monte Carlo simulations for two methods.

From Table 3.2 and Fig. 3.3 it can be concluded that for this case the proposed method has higher performance than the method of van Wingerden and Verhaegen (2009) in terms of VAF.

The conclusions are presented in the next subsection.

3.8. CONCLUSION

In this paper we proposed a novel convex LPV subspace identification method which exploits the multi-linear low-rank property of the parameter tensors. This is done by regularization with the tensor nuclear norms of parameter tensors. However, these tensors are reorganizations of the LPV sub-Markov parameters which inherently suffer from the 'curse-of-dimensionality'. There we additionally present three minor contributions to allow computing these norms 'curse-of-dimensionality' free. Firstly, we show how to express the parameter tensors in the dual form. Secondly, we show that the arguments

of the norms can be replaced by much smaller arguments without changing the result. Thirdly, we show how these arguments can be computed 'curse-of-dimensionality' free. Then, the proposed method becomes 'curse-of-dimensionality' free in memory and computation. Finally, simulation results showed that the novel method has higher performance than the regularized LPV-PBSID$_{opt}$ technique in terms of bias and variance.

3.A. ALGORITHM FOR THE ADMISSIBILITY TENSOR

First define the ones vector and zeros vector:

$$\bar{1}_c = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^T \in \mathbb{R}^{c\times 1}, \text{ for } c \geq 1$$
$$\bar{0}_c = \begin{bmatrix} 0 & \cdots & 0 \end{bmatrix}^T \in \mathbb{R}^{c\times 1}, \text{ for } c \geq 1,$$

where for $c \leq 0$ the result is defined as an empty vector.

Algorithm 3.A.1 *The admissibility tensor $\mathcal{A}_H$*
Input: System order $\hat{n}$ and dimension count $D \geq 2$
Output: Admissibility tensor $\mathcal{A}_H$

1. $v = \begin{bmatrix} 1 & \hat{n}+2 & 2\hat{n}+3 & \dots & \hat{n}^2 \end{bmatrix}^T$
2. *for* $d = 3$ *to* D *do*
   (a) $c = (\hat{n}+1)\hat{n}^{2(d-2)}$
   (b) $V_1 = v\bar{1}_{\hat{n}}^T$
   (c) $V_2 = c\bar{1}_{\hat{n}^{d-2}} \begin{bmatrix} 1 & 2 & \dots & \hat{n} \end{bmatrix}$
   (d) $v = vec(V_1 + V_2 - c)$
3. *end for*
4. $\mathcal{A}_H = zeros(\begin{bmatrix} \hat{n} & \hat{n}^2\bar{1}_{D-2}^T & \hat{n} \end{bmatrix})$
5. $\mathcal{A}_H(v) = 1$;

Proof follows through straightforward computations.

3.B. ALGORITHM FOR DUPLICATION CORRECTION FACTORS

Algorithm 3.B.1 *The duplication correction factors $\bar{s}$*
Input: Incremental window $\bar{f}$, dimension count D
Output: The duplication correction factors $\bar{s}$

1. $v = \begin{bmatrix} 1 & 2 & 4 & \cdots & 2^{\bar{f}-1} \end{bmatrix}$
2. *Take the Kronecker product of D v's:* $w = (v \otimes v \otimes \cdots \otimes v)$

3. $w = log_2 w$

4. *for* $a = 0$ *to* $(\bar{f} - 1)D$

   (a) $\bar{s}(a+1) = length(find(w == a))$

5. *end for*

Proof follows through straightforward computations.

BIBLIOGRAPHY

Balas, G. J. Linear, parameter-varying control and its application to aerospace systems. In *ICAS Congress Proceedings*, 2002.

Bianchi, F. D., Mantz, R. J., and Christiansen, C. F. Gain scheduling control of variable-speed wind energy conversion systems using quasi-LPV models. *Control Engineering Practice*, 13(2):247–255, 2005.

Brewer, J. Kronecker products and matrix calculus in system theory. *IEEE Transactions on circuits and systems*, 25(9):772–781, 1978.

Butcher, M., Karimi, A., and Longchamp, R. On the consistency of certain identification methods for linear parameter varying systems. *IFAC Proceedings Volumes*, 41(2):4018–4023, 2008.

Chiuso, A. The role of vector autoregressive modeling in predictor-based subspace identification. *Automatica*, 43(6):1034–1048, 2007.

Cox, P. and Tóth, R. Alternative form of predictor based identification of LPV-SS models with innovation noise. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 1223–1228. IEEE, 2016.

De Caigny, J., Camino, J. F., and Swevers, J. Interpolating model identification for SISO linear parameter-varying systems. *Mechanical Systems and Signal Processing*, 23(8): 2395–2417, 2009.

De Lathauwer, L., De Moor, B., and Vandewalle, J. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

Favoreel, W., De Moor, B., and Van Overschee, P. Subspace identification of bilinear systems subject to white inputs. *Automatic Control, IEEE Transactions on*, 44(6):1157–1165, 1999.

Felici, F., Van Wingerden, J.-W., and Verhaegen, M. Subspace identification of MIMO LPV systems using a periodic scheduling sequence. *Automatica*, 43(10):1684–1697, 2007.

Friedland, S. and Lim, L.-H. Computational complexity of tensor nuclear norm. *arXiv preprint arXiv:1410.6072*, 2014.

Gebraad, P. M., van Wingerden, J.-W., Fleming, P. A., and Wright, A. D. LPV subspace identification of the edgewise vibrational dynamics of a wind turbine rotor. In *Control Applications (CCA), 2011 IEEE International Conference on*, pages 37–42. IEEE, 2011a.

Gebraad, P. M., van Wingerden, J.-W., van der Veen, G. J., and Verhaegen, M. LPV subspace identification using a novel nuclear norm regularization method. In *American Control Conference (ACC)*, 2011b.

Giarré, L., Bauso, D., Falugi, P., and Bamieh, B. LPV model identification for gain scheduling control: An application to rotating stall and surge control problem. *Control Engineering Practice*, 14(4):351–361, 2006.

Golub, G. H., Heath, M., and Wahba, G. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.

Gunes, B., van Wingerden, J.-W., and Verhaegen, M. Predictor-based tensor regression (PBTR) for LPV subspace identification. *Automatica*, 79:235 – 243, 2017a. ISSN 0005-1098.

Gunes, B., van Wingerden, J.-W., and Verhaegen, M. Tensor networks for MIMO LPV system identification. *Submitted to International Journal of Control*, 2017b.

Gunes, B., van Wingerden, J.-W., and Verhaegen, M. Tensor nuclear norm LPV subspace identification. *IEEE Transactions on Automatic Control*, 2018.

Håstad, J. Tensor rank is NP-complete. *Journal of Algorithms*, 11(4):644–654, 1990.

He, X., Cai, D., and Niyogi, P. Tensor subspace analysis. In *Advances in neural information processing systems*, pages 499–506, 2005.

Hjalmarsson, H., Welsh, J. S., and Rojas, C. R. Identification of box-jenkins models using structured ARX models and nuclear norm relaxation. *IFAC Proceedings Volumes*, 45 (16):322–327, 2012.

Knudsen, T. Consistency analysis of subspace identification methods based on a linear regression approach. *Automatica*, 37(1):81–89, 2001.

Larimore, W. E. and Buchholz, M. ADAPT-LPV software for identification of nonlinear parameter-varying systems. *IFAC Proceedings Volumes*, 45(16):1820–1825, 2012.

Laurain, V., Gilson, M., Tóth, R., and Garnier, H. Refined instrumental variable methods for identification of LPV box-jenkins models. *Automatica*, 46(6):959–967, 2010.

Leith, D. J. and Leithead, W. E. Survey of gain-scheduling analysis and design. *International journal of control*, 73(11):1001–1025, 2000.

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

Recht, B., Fazel, M., and Parrilo, P. A. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.

Remmlinger, J., Buchholz, M., and Dietmayer, K. Identification of a bilinear and parameter-varying model for lithium-ion batteries by subspace methods. In *American Control Conference (ACC), 2013*, pages 2268–2273. IEEE, 2013.

Rugh, W. J. and Shamma, J. S. Research on gain scheduling. *Automatica*, 36(10):1401–1425, 2000.

Scherer, C. W. LPV control and full block multipliers. *Automatica*, 37(3):361–375, 2001.

Shamma, J. S. An overview of LPV systems. In *Control of linear parameter varying systems with applications*, pages 3–26. Springer, 2012.

Signoretto, M., De Lathauwer, L., and Suykens, J. A. Nuclear norms for tensors and their use for convex multilinear estimation. *Submitted to Linear Algebra and Its Applications*, 43, 2010.

Tóth, R. *Modeling and identification of linear parameter-varying systems*, volume 403. Springer, 2010.

Tóth, R., Abbas, H. S., and Werner, H. On the state-space realization of LPV input-output models: Practical approaches. *Control Systems Technology, IEEE Transactions on*, 20 (1):139–153, 2012.

van der Maas, R., van der Maas, A., and Oomen, T. Accurate frequency response function identification of LPV systems: A 2D local parametric modeling approach. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pages 1465–1470. IEEE, 2015.

van der Veen, G., van Wingerden, J.-W., Bergamasco, M., Lovera, M., and Verhaegen, M. Closed-loop subspace identification methods: an overview. *IET Control Theory & Applications*, 7(10):1339–1358, July 2013. ISSN 1751-8652.

van Wingerden, J.-W. and Verhaegen, M. Subspace identification of bilinear and LPV systems for open- and closed-loop data. *Automatica*, 45(2):372 – 381, 2009. ISSN 0005-1098.

van Wingerden, J.-W., Felici, F., and Verhaegen, M. Subspace identification of MIMO LPV systems using a piecewise constant scheduling sequence with hard/soft switching. In *European Control Conference (ECC), 2007*, pages 927–934. IEEE, 2007.

Vasilescu, M. A. O. and Terzopoulos, D. Multilinear analysis of image ensembles: Tensorfaces. In *Computer Vision ECCV 2002*, pages 447–460. Springer, 2002.

Verdult, V. and Verhaegen, M. Kernel methods for subspace identification of multivariable LPV and bilinear systems. *Automatica*, 41(9):1557–1565, 2005.

Verhaegen, M. and Hansson, A. Nuclear norm subspace identification (N2SID) for short data batches. *CoRR*, abs/1401.4273, 2014.

Vervliet, N., Debals, O., Sorber, L., Van Barel, M., and De Lathauwer, L. Tensorlab 3.0. *Available online. URL: http://www.tensorlab.net*, 2016.

Wassink, M. G., van de Wal, M., Scherer, C., and Bosgra, O. LPV control for a wafer stage: beyond the theoretical solution. *Control Engineering Practice*, 13(2):231–245, 2005.

Yu, C. and Verhaegen, M. Local subspace identification of distributed homogeneous systems with general interconnection patterns. *Proceedings of the 13th IFAC Symposium on System Identification, Beijing*, 2015.

4

TENSOR NETWORKS FOR MIMO LPV SYSTEM IDENTIFICATION

In this paper we present a novel Multiple Input Multiple Output (MIMO) LPV state-space refinement system identification algorithm that uses tensor networks. *Namely, we represent the LPV sub-Markov parameters, data and state-revealing matrix condensely and in exact manner using specific tensor networks. These representations circumvent the 'curse-of-dimensionality' as they inherit the properties of tensor trains. The proposed algorithm is 'curse-of-dimensionality'-free in memory and computation and has conditioning guarantees. Its performance is illustrated using simulation cases and additionally compared with existing methods.*

4.1. INTRODUCTION

In this paper the focus is on Linear Parameter Varying (LPV) systems. Because these systems are able to describe time-varying and non-linear systems, while allowing for powerful properties extended from linear system theory. Also, there are control design methods which can guarantee robust performance (Scherer, 2001), and several convex and non-convex identification methods exist. The dynamics of LPV systems are a function of the scheduling sequence, which contains the time-varying and non-linear effects. In this paper we focus on affine functions of known arbitrary scheduling sequences, which are relevant to wind turbine applications (van Wingerden and Verhaegen, 2009; Bianchi et al., 2005; Gebraad et al., 2011). Other applications are aircraft applications (Balas, 2002), compressors (Giarré et al., 2006), batteries (Remmlinger et al., 2013) and wafer stages (Wassink et al., 2005; van der Maas et al., 2015). The scheduling sequence can be any function of known variables. This representation allows us for example to take the effect of blade position on gravity forces and blade rotational speed into account (Gebraad et al., 2013).

In this paper the focus is on LPV identification methods. That is, from input-output and scheduling data we try to estimate an LPV model of the system. This model can

Parts of this chapter have been published in Gunes et al. (2017b).

then be used to design a controller. Several methods exist, which can be divided in two different ways. Firstly, global and local methods can be distinguished (Tóth, 2010; De Caigny et al., 2009; Shamma, 2012). Local methods utilize that LPV systems act as LTI systems when the scheduling sequence is kept at constant value. These methods identify LTI models at several fixed constant value scheduling sequences and then combine them into an LPV model through interpolation techniques. This does require the application to allow for such experiments. In contrast, global methods use only a single experiment. In this paper only global methods will be considered. Secondly, input-output (Tóth, 2010; Laurain et al., 2010) and state-space methods (van Wingerden and Verhaegen, 2009; Verdult et al., 2003; Cox and Tóth, 2016; Larimore and Buchholz, 2012) exist. The first produces input-output models and the latter state-space models. While in the Linear Time Invariant (LTI) framework these two types of models can be transformed back and forth, in the LPV case this is not trivial and problematic (Tóth et al., 2012). In this paper the focus is on state-space models, because they are the mainstream control design models and can inherently deal with Multiple Input Multiple Output (MIMO) problems. More specifically, we focus on predictor-based methods because they can deal with closed-loop data. However, one possible drawback of these methods is the 'curse-of-dimensionality'.

Namely, the number of to be estimated (LPV sub-Markov) parameters scales exponentially. This can give problems with memory usage and computational cost. Furthermore, the number of parameters can exceed the number of data points, resulting in ill-conditioned problems. Both problems demand special care.

In literature, these problems have been tackled using both convex methods and non-convex refinement methods. Convex methods such as proposed by van Wingerden and Verhaegen (2009); Gebraad et al. (2011) overcome the memory and computation cost problem by using a dual parametrization, and address the ill-condition with regularization. Regularization is a way to introduce a bias-variance trade-off in estimates. Refinement methods use a non-linear parametrization with few variables to overcome both problems. However, this returns a non-convex optimization problem which requires initialization by an initial estimate. Hence the goal of refinement methods is to refine initial estimates. The method proposed by Verdult et al. (2003) directly parametrizes the state-space matrices for this purpose, and works for open-loop data.

These problems have also been tackled in Gunes et al. (2017a, 2018). Those two papers and this paper differ in that they use different tensors and different tensor techniques. The different tensor techniques result in completely different methods. In Gunes et al. (2018), a parameter tensor is constructed whose matricizations are each low-rank. This allows exploitation through nuclear norms and has resulted in a convex subspace method. Tensor techniques are not used to tackle the 'curse-of-dimensionality' in that paper. It is worth remarking that some tensors of Gunes et al. (2018) have been generalized from tensors presented in this paper. In Gunes et al. (2017a), a parameter tensor is constructed which admits a constrained polyadic decomposition and parametrization. This resulted in a refinement method. However, the computation of the polyadic rank is NP-hard (Håstad, 1990) and approximation with a fixed polyadic rank in the Frobenius norm can be ill-posed (De Silva and Lim, 2008). This method is not 'curse-of-dimensionality'-free in memory or computation. In this paper, a tensor network ap-

proach will be used. The identification problem will be recast and optimized using tensor networks. The major difference with previous work is that this approach uses tensor networks. These allow the method to circumvent the explicit construction of the LPV sub-Markov parameters and makes the proposed refinement method 'curse-of-dimensionality'-free in memory and computation. This includes a novel technique for constructing the estimate state-revealing matrix directly from the estimate tensor estimate. Another difference with Gunes et al. (2018) is that this method is a refinement method, and does not limit the exploitation of the tensor structure to a regularization term. For completeness, the problem definition is to develop an LPV state-space refinement method which by exploiting underlying *tensor network* structure is 'curse-of-dimensionality'-free in memory and computation and successfully refines initial estimates from convex methods.

In this paper, we present the following novel contributions. Firstly, the LPV identification problem is recast and optimized using tensor networks to make the proposed refinement method 'curse-of-dimensionality'-free in memory and computation. This is done by circumventing the explicit construction of the LPV sub-Markov parameters. Namely, operations can be performed directly on the tensor network decompositions. In detail, the used class of tensor network (Batselier et al., 2017) is a generalization of tensor trains (Oseledets, 2011). They inherit tensor train efficiency results, and become tensor trains for single-output problems. These decompositions allow efficient storage and computation in the tensor network domain. This recast in tensor networks includes not only the LPV sub-Markov parameters, but also the data and state-revealing matrix. More specifically, the LPV sub-Markov parameters are shown to admit exact tensor network representation with ranks equal to the model order, and the data tensor admits exact and sparse tensor network representation. Secondly, these properties are exploited to obtain a condensed tensor network parametrization which can be optimized 'curse-of-dimensionality'-free in memory and computation. Thirdly, we propose an efficient way to obtain the estimate state sequence from the estimate tensor networks without explicitly constructing the LPV sub-Markov parameters. Additionally, we provide an upper bound on the condition numbers of its sub-problems.

The paper outline is as follows. In Section 4.3 our LPV identification problem is presented in the tensor network framework. Subsequently in Section 4.4 a tensor network identification method is proposed. Finally, simulations results and conclusions are presented. In the next section, background is provided on LPV identification and tensor trains (and networks).

4.2. BACKGROUND

Before presenting the novel results, relevant topics are reviewed. These topics are LPV identification with state-space matrices, tensor trains and tensor networks. Throughout the paper, for clarity matrices will be denoted by capital characters, tensors which are part of a tensor network decomposition by calligraphic characters and other tensors by bold characters.

4.2.1. LPV SYSTEM IDENTIFICATION WITH STATE-SPACE MATRICES

In this subsection LPV system identification with state-space matrices is reviewed (van Wingerden and Verhaegen, 2009). First define the following signals relevant to LPV state-space systems:

$$x_k \in \mathbb{R}^n,\ u_k \in \mathbb{R}^r,\ y_k \in \mathbb{R}^l,\ \mu_k \in \mathbb{R}^m, \tag{4.1}$$

as the state, input, output and scheduling sequence vector at sample number k. For later use, additionally define:

$$\begin{aligned} \mu_k &= \begin{bmatrix} \mu_k^{(1)} & \dots & \mu_k^{(m)} \end{bmatrix}^T \\ \mu &= \begin{bmatrix} \mu_1 & \dots & \mu_N \end{bmatrix} \end{aligned} \tag{4.2}$$

Predictor-based methods (Chiuso, 2007; van Wingerden and Verhaegen, 2009) use the innovation representation and predictor-based representation of LPV state-space systems. Therefore we directly present the assumed data-generating system as:

$$x_{k+1} = \sum_{i=1}^{m} \mu_k^{(i)} \left(A^{(i)} x_k + B^{(i)} u_k + K^{(i)} e_k \right) \tag{4.3a}$$

$$y_k = C x_k + e_k, \tag{4.3b}$$

where the variables $A^{(i)}$, $B^{(i)}$, C and $K^{(i)}$ are appropriately dimensioned state, input, output and observer matrices and e_k is the innovation term at sample k. Notice that the output equation has an LTI C matrix and the feed-through term D is the zero matrix. This assumption is made for the sake of presentation and simplicity of derivation, similar to what has been done in van Wingerden and Verhaegen (2009). This will not trivialize the bottleneck 'curse-of-dimensionality'. Furthermore, it is assumed that μ_k is known and this state-space system has affine dependency on μ_k like in van Wingerden and Verhaegen (2009). That is, $\mu_k^{(1)} = 1\ \forall k$. The predictor-based representation follows by substituting (4.3b) in (4.3a):

$$x_{k+1} = \sum_{i=1}^{m} \mu_k^{(i)} \left(\tilde{A}^{(i)} x_k + \bar{B}^{(i)} \begin{bmatrix} u_k \\ y_k \end{bmatrix} \right) \tag{4.4a}$$

$$y_k = C x_k + e_k, \tag{4.4b}$$

where $\tilde{A}^{(i)}$ is $A^{(i)} - K^{(i)} C$ and $\bar{B}^{(i)}$ is $[B^{(i)}, K^{(i)}]$. Notice that the states here are the observer states. Also, define p as a past window for later use. Furthermore, define the discrete-time time-varying transition matrix (van Wingerden and Verhaegen, 2009):

$$\tilde{A}_k = \sum_{i=1}^{m} \mu_k^{(i)} \tilde{A}^{(i)} \tag{4.5a}$$

$$\phi_{j,k} = \tilde{A}_{k+j-1} \dots \tilde{A}_{k+1} \tilde{A}_k \tag{4.5b}$$

Predictor-based methods make the assumption that this matrix is exactly zero when j is greater than or equal to the past window p (van Wingerden and Verhaegen, 2009; Chiuso, 2007):

$$\phi_{j,k} \approx 0\ \forall\ j \geq p \tag{4.6}$$

A similar approximation is also used in several LTI methods (van der Veen et al., 2013). If the predictor-based system is uniformly exponentially stable, then the approximation error can be made arbitrarily small by increasing p (Knudsen, 2001). In fact the introduced bias disappears as p goes to infinity, but is hard to quantify for finite p (Chiuso, 2007). This assumption results in the predictor-based data equation.

Before presenting this data equation, first some definitions must be given. For ease of presenting matrix sizes, define:

$$q = (l+r)\sum_{j=1}^{p} m^j, \tag{4.7}$$

Now define the extended LPV controllability matrix (van Wingerden and Verhaegen, 2009) as:

$$K_{(p)} = \begin{bmatrix} L_p & \dots & L_2 & \bar{B} \end{bmatrix} \qquad \in \mathbb{R}^{n\times q}, \tag{4.8a}$$

$$\bar{B} = \begin{bmatrix} \bar{B}^{(1)} & \dots & \bar{B}^{(m)} \end{bmatrix} \qquad \in \mathbb{R}^{n\times (l+r)m} \tag{4.8b}$$

$$L_2 = \begin{bmatrix} \tilde{A}^{(1)}\bar{B} & \dots & \tilde{A}^{(m)}\bar{B} \end{bmatrix} \qquad \in \mathbb{R}^{n\times (l+r)m^2} \tag{4.8c}$$

$$L_{i+1} = \begin{bmatrix} \tilde{A}^{(1)}L_i & \dots & \tilde{A}^{(m)}L_i \end{bmatrix} \qquad \in \mathbb{R}^{n\times (l+r)m^{i+1}}, 2 \geq i \geq p-1 \tag{4.8d}$$

For this purpose also define the data matrix (van Wingerden and Verhaegen, 2009) as:

$$Z_{k+p} = N^p_{k+p}\bar{z}_{k+p} \in \mathbb{R}^q, \tag{4.9}$$

where $\bar{z}_{k+p}$ is:

$$z_{k+p} = \begin{bmatrix} u_{k+p} \\ y_{k+p} \end{bmatrix} \in \mathbb{R}^{l+r}, \tag{4.10a}$$

$$\bar{z}_{k+p} = \begin{bmatrix} z_k \\ z_{k+1} \\ \vdots \\ z_{k+p-1} \end{bmatrix} \in \mathbb{R}^{p(l+r)}, \tag{4.10b}$$

where the subscript of $\bar{z}_{k+p}$ indicates its relation to y_{k+p} in equation 4.14, and N^p_{k+p} is:

$$P_{j|k} = \mu_{k+j-1} \otimes \cdots \otimes \mu_k \otimes I_{l+r} \in \mathbb{R}^{(l+r)m^j \times (l+r)}, \tag{4.11a}$$

$$N^p_{k+p} = \begin{bmatrix} P_{p|k} & 0 & \cdots & 0 \\ 0 & P_{p-1|k+1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & P_{1|k+p-1} \end{bmatrix} \in \mathbb{R}^{q\times p(l+r)}, \tag{4.11b}$$

where μ_k is a column vector and the operator $\otimes$ is the Kronecker product (Brewer, 1978). Notice that Z_{k+p} involves samples from k to $k+p-1$. With these definitions and equation (4.4), the states can be described by:

$$x_{k+p} = \phi_{p,k} x_k + K_{(p)} Z_{k+p} \tag{4.12}$$

Using assumption (4.6) gives:

$$x_{k+p} \approx K_{(p)} Z_{k+p}, \tag{4.13}$$

which yields the predictor-based data equation:

$$y_{k+p} \approx CK_{(p)} Z_{k+p} + e_{k+p}, \tag{4.14}$$

where the entries of $CK_{(p)}$ are named the LPV sub-Markov parameters.

However, the number of columns of $K_{(p)}$ (or rows of Z_{k+p}) is q (4.7), which scales exponentially with the past window. Additionally, as argued earlier in this section the past window affects the approximation error and bias. Hence, the resulting matrix sizes can yield problems with memory storage, computation costs and cause ill-conditioned estimation problems. For brevity, define:

Definition 4.2.1 *The curse-of-dimensionality*

In this paper we say that a vector, matrix, tensor or set suffers from the curse-of-dimensionality if the number of elements scale exponentially with the past window.

For later use, additionally define:

$$Z = \begin{bmatrix} Z_{p+1} & \dots & Z_N \end{bmatrix} \in \mathbb{R}^{q \times N-p}, \tag{4.15}$$

and

$$z = \begin{bmatrix} z_1 & \dots & z_N \end{bmatrix} \tag{4.16}$$

Notice that the width of Z is $N-p$ because every Z_k involves samples from $k-p$ to $k-1$. Thus when using all samples, this yields $N-p$ many Z_k.

Also for later use, define the partial extended observability matrix as in van Wingerden and Verhaegen (2009):

$$\Gamma^f = \begin{bmatrix} C \\ C\tilde{A}^{(1)} \\ \vdots \\ C\left(\tilde{A}^{(1)}\right)^{f-1} \end{bmatrix} \in \mathbb{R}^{fl \times n}, \tag{4.17}$$

where the future window f is chosen as $p \geq f \geq n$ (van der Veen et al., 2013). We assume this matrix is full column rank.

An estimate of the LPV state-space matrices can be obtained as follows (van Wingerden and Verhaegen, 2009). Firstly, the matrix $CK_{(p)}$ can be estimated using linear or non-linear regression using (4.14). Secondly, this estimate can be used to construct a state-revealing matrix whose Singular Value Decomposition (SVD) returns an estimate

of the state sequence. We define the state sequence as $X = \begin{bmatrix} x_{p+1} & \dots & x_N \end{bmatrix}$ and assume it to have full row-rank as in van Wingerden and Verhaegen (2009). If the predictor-based assumption (4.6) holds exactly, the following equality holds exactly too:

$$\Gamma^f K_{(p)} Z \approx \Gamma^f X, \tag{4.18}$$

such that its SVD:

$$\Gamma^f K_{(p)} Z = \bar{U} \Sigma V^T, \tag{4.19}$$

returns the states through $X = \Sigma V^T$ modulo global state-coordinate transformation. Additionally, the model order can be chosen by discarding the smallest singular values in Σ.

However, notice that $\Gamma^f K_{(p)}$ is not suitable when working with estimates, because it involves terms which are not estimated. One example are its bottom rows $C\big(\tilde{A}^{(1)}\big)^{f-1} K_{(p)}$. But under the same predictor-based assumption (4.6), these terms can be assumed zero to obtain the 'state-revealing matrix' R:

$$R = \begin{bmatrix} CL_p & CL_{p-1} & \cdots & CL_1 \\ 0 & C\tilde{A}^{(1)} L_{p-1} & \cdots & C\tilde{A}^{(1)} L_1 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & & C\big(\tilde{A}^{(1)}\big)^{f-1} L_1 \end{bmatrix} Z \in \mathbb{R}^{fl \times N-p}, \tag{4.20}$$

where the block-lower-triangular entries are assumed negligible. The estimate R can be constructed from the columns of the estimate $CK_{(p)}$. Therefore, the SVD of R is used to obtain an estimate state sequence. Additionally, for persistence of excitation it is assumed that $[\mu_1 \ \dots \ \mu_{N-p+1}]$ has rank m and $N - p + 1 > m$. Finally, the estimate state sequence allows readily obtaining the state-space matrices using two least squares problems (van Wingerden and Verhaegen, 2009).

4.2.2. TENSOR TRAINS AND NETWORKS

In this subsection the background on tensors and the tensor train framework (Oseledets, 2011) is presented. Furthermore we also present the slightly generalized tensor network framework of (Batselier et al., 2017).

Firstly, the notion of a tensor has to be defined. A tensor is a multi-dimensional generalization of a matrix. That is, it can have more than two dimensions. Formally define a tensor and how its elements are accessed:

Definition 4.2.2 *Consider a tensor* $\mathbf{T}$ *with* d *dimensions. Let its size be* $\mathbf{T} \in \mathbb{R}^{J_1 \times \cdots \times J_d}$. *Let* j_1 *to* j_d *be the* d *indices of the tensor, which each correspond to one dimension. Then* $\mathbf{T}(j_1, \dots, j_d)$ *is its single element at position* $j_1, \dots, j_d$. *Furthermore, the symbol ':' is used when multiple elements are involved. More specifically, ':' indicates that an index is not fixed. For example,* $\mathbf{T}(:, \dots, :) = \mathbf{T}$ *and* $\mathbf{T}(:, :, j_3, \dots, j_d) \in \mathbb{R}^{J_1 \times J_2}$ *is a matrix obtained by fixing the indices* j_3 *to* j_d.

The multi-dimensional structure of tensors can be exploited using techniques from the tensor framework such as tensor networks.

Secondly, tensor trains are discussed (Oseledets, 2011). They are condense decompositions of tensors. These tensor trains consist of d 'cores' $\mathcal{A}^{(1)}$ to $\mathcal{A}^{(d)}$, where each core is a three-dimensional tensor. Notice that to distinguish regular tensors and cores, calligraphic characters have been used for cores. The relation between a tensor and its tensor train is defined element-wise as:

$$\mathbf{T}(i_1,\ldots,i_d) = \mathcal{A}^{(1)}(:,i_1,:)\mathcal{A}^{(2)}(:,i_2,:)\ldots\mathcal{A}^{(d)}(:,i_d,:), \tag{4.21}$$

where the left hand side is defined in Definition 4.2.2. Notice that because the left hand side is a scalar, $\mathcal{A}^{(1)}(:,i_1,:)$ is a row vector and $\mathcal{A}^{(d)}(:,i_d,:)$ is a column vector. The remaining terms in the product are matrices. For brevity of notation, define the *generator* operator $g(*)$ as:

$$\mathbf{T} = g(\mathcal{A}^{(1)},\ldots,\mathcal{A}^{(d)}), \tag{4.22}$$

where $\mathcal{A}^{(*)}$ are the cores of the tensor train of $\mathbf{T}$

The tensor train decomposition allows describing a tensor with less variables and performing computations without constructing the original tensor and at lower computational cost. To specify this, first, define the *tensor train ranks* as the first and third dimension of each core. In this paragraph, let each tensor train rank [1] be $\bar{r}$ and let the size of the original tensor be $\in \mathbb{R}^{J\times\cdots\times J}$. Then the memory cost of the tensor train scales as $\mathcal{O}(J\bar{r}^2)$, while the memory cost of the original tensor scales as $\mathcal{O}(J^d)$. Hence the memory cost can be reduced, depending on the size of the tensor train ranks. It is also possible to decrease the tensor train ranks by introducing an approximation (with guaranteed error bounds (Oseledets, 2011)). The computational cost can also be reduced, by performing computations using the tensor train without constructing the original tensor. This is possible for many operations (Oseledets, 2011). For example, consider the inner product of the example tensor with itself. This is equivalent to the sum of its individual entries squared. A straight-forward computation scales with J^d, while performing the computation directly on its tensor train using Oseledets (2011) scales with dJr^3. Hence the computation cost can be reduced, depending on the size of the tensor train ranks. Most importantly, the memory and computation cost scale differently when using tensor trains.

This property allows tensor trains to break exponential scaling and 'curse-of-dimensionality' (Definition 4.2.1). For brevity, also define:

Definition 4.2.3 *'Curse-of-dimensionality'-free in memory and computation*

In this paper we say that an algorithm is 'curse-of-dimensionality'-free in memory and computation if its memory usage and computational cost does not scale exponentially with the past window, but for example as a polynomial.

Furthermore, in order to be able to deal with multiple output systems, we use the slightly generalized tensor network of (Batselier et al., 2017). This generalization is that we allow $\mathcal{A}^{(1)}(:,i_1,:)$ to be a full matrix instead of a row vector. As a result, the right hand side of (4.21) would return a column vector instead of a scalar. This generalization preserves the previously presented properties. In the remainder of this paper, we will refer to this specific tensor network directly as 'tensor network'.

Additionally, we define the inner product for later use:

[1] except the first and last one, as they are fixed at one (4.21)

Definition 4.2.4 *The inner product of two tensors is the sum of their element-wise multiplication (Oseledets, 2011). This is well defined if the tensors have the same size. If one tensor has an additional leading dimension, we use the definition of (Oseledets, 2011) to obtain a column vector containing inner products of slices of the larger tensor with the other tensor.*

If the two tensors have a tensor train or network representations, then this operation can be performed directly using those representations (Oseledets, 2011). The resulting computational cost scales linearly with the dimension count and dimension sizes and as a third power of the tensor train ranks.

The inner product operation can be performed using the Tensor Train Toolbox (Oseledets, 2011). The technical descriptions of this subsection will be used in the next section to derive the tensor networks for our LPV identification problem.

4.3. THE LPV IDENTIFICATION PROBLEM IN TENSOR NETWORK FORM

In this section we present several key aspects of the LPV identification problem using tensor networks (Subsection 4.2.2). The LPV sub-Markov parameters, their associated data matrix (4.9), the system output, the model output and the state-revealing matrix are represented using tensor networks. This allows directly formulating the proposed method based on tensor networks in the next section. Through the use of these tensor networks the proposed method is 'curse-of-dimensionality'-free in memory and computation (Definition 4.2.3).

4.3.1. AN ILLUSTRATION FOR A SIMPLE CASE

In this subsection we present an example to illustrate the two more complicated tensor networks. Namely, we derive and present the tensor networks for the LPV sub-Markov parameters and their associated data matrix (4.9). The following simplified SISO LPV state-space system is considered:

$$x_{k+1} = \sum_{i=1}^{m=2} \mu_k^{(i)} \left(A^{(i)} x_k \right) + B u_k \tag{4.23a}$$

$$y_k = C x_k + e_k, \tag{4.23b}$$

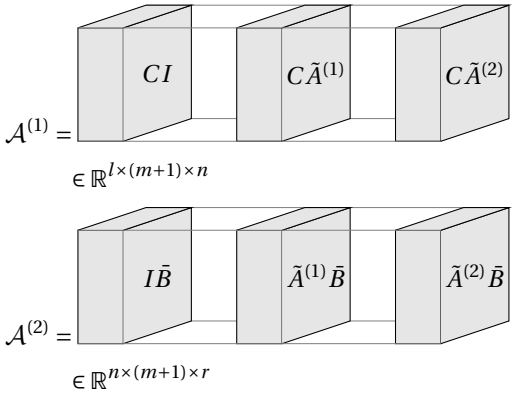
with one input, one output and two states. This simplification also affects other equations. Because there is only one input, the tensor network is also just a tensor train. We consider the following, first few, LPV sub-Markov parameters corresponding to $p = 3$:

$$\begin{aligned} &C\tilde{A}^{(1)}\tilde{A}^{(1)}\bar{B}, C\tilde{A}^{(1)}\tilde{A}^{(2)}\bar{B}, \ldots \\ &C\tilde{A}^{(2)}\tilde{A}^{(1)}\bar{B}, C\tilde{A}^{(2)}\tilde{A}^{(2)}\bar{B}, \ldots \\ &C\tilde{A}^{(1)}\bar{B}, C\tilde{A}^{(2)}\bar{B}, \ldots \\ &C\bar{B} \qquad\qquad\qquad\qquad\qquad\qquad \in \mathbb{R}^{1\times 7} \end{aligned}$$

These parameters can be rearranged into the following tensor:

$$\mathbf{T} = \begin{bmatrix} C\bar{B} & C\tilde{A}^{(1)}\bar{B} & C\tilde{A}^{(2)}\bar{B} \\ C\tilde{A}^{(1)}\bar{B} & C\tilde{A}^{(1)}\tilde{A}^{(1)}\bar{B} & C\tilde{A}^{(1)}\tilde{A}^{(2)}\bar{B} \\ C\tilde{A}^{(2)}\bar{B} & C\tilde{A}^{(2)}\tilde{A}^{(1)}\bar{B} & C\tilde{A}^{(2)}\tilde{A}^{(2)}\bar{B} \end{bmatrix} \in \mathbb{R}^{3\times 3}, \tag{4.24}$$

where some entries appear multiple times, such that it admits an exact tensor train network decomposition with its cores equal to:



Notice that the ranks of this tensor network are equal to the system order. In detail, the illustrated sub-tensors can be accessed using for example $\mathcal{A}^{(1)}(:,1,:) = CI$. Proof of this decomposition follows through straightforward computations.

Details on its generalization are as follows. Firstly, while the parameter tensor for this example is a two-dimensional tensor (matrix), the general case produces a higher dimensional tensor. Secondly, while for this example the matrix $\bar{B}$ is only a vector and absorbed into the second core for simplicity, it will appear in a separate core for the general case. The matrix C will not appear in a separate core, because optimizing this core (with alternating least squares) would stringently require $l \geq n$. The general case will be presented in the next subsection.

Next the tensor network of the associated data matrix (4.9) is illustrated using the same example. For one sample, the following vector is obtained:

$$\begin{aligned} Z_k = N^p_{k+p}\bar{z}_{k+p} &= \begin{bmatrix} \mu_{k-1}\otimes\mu_{k-2} & 0 & 0 \\ 0 & \mu_{k-1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{k-3} \\ u_{k-2} \\ u_{k-1} \end{bmatrix} \\ &= \begin{bmatrix} (\mu_{k-1}\otimes\mu_{k-2})u_{k-3} \\ \mu_{k-1}u_{k-2} \\ u_{k-1} \end{bmatrix} \in \mathbb{R}^{7\times 1}, \end{aligned} \tag{4.25}$$

where μ_k is a column vector and $u_j = u_j^T$ is a scalar. This data can be rearranged into the

following tensor:

$$\mathbf{Z}_k = \begin{bmatrix} u_{k-1} & \mu_{k-1}^T u_{k-2}/2 \\ \mu_{k-1} u_{k-2}/2 & \mu_{k-1}\mu_{k-2}^T u_{k-3} \end{bmatrix} \in \mathbb{R}^{3\times 3} \tag{4.26a}$$

$$= \begin{bmatrix} 1(1 \otimes u_{k-1})^T & 1(\mu_{k-1} \otimes u_{k-2}/2)^T \\ \mu_{k-1}(1 \otimes u_{k-2}/2)^T & \mu_{k-1}(\mu_{k-2} \otimes u_{k-3})^T \end{bmatrix}, \tag{4.26b}$$

where some entries appear multiple times (and are divided by two), such that it admits an exact tensor train network decomposition. We present its cores as sliced matrices:

$$\mathcal{B}_k^{(1)}(:,1,:) = [1,0] \tag{4.27a}$$

$$\mathcal{B}_k^{(1)}(:,1+b,:) = [0,\mu_{k-1}^{(b)}],\ b \in \{1,\ldots,m\} \tag{4.27b}$$

$$\mathcal{B}_k^{(2)}(:,1,:) = \begin{bmatrix} 1 \otimes u_{k-1} \\ 1 \otimes u_{k-2}/2 \end{bmatrix} \tag{4.27c}$$

$$\mathcal{B}_k^{(2)}(:,1+b,:) = \begin{bmatrix} \mu_{k-1}^{(b)} \otimes u_{k-2}/2 \\ \mu_{k-2}^{(b)} \otimes u_{k-3} \end{bmatrix},\ b \in \{1,\ldots,m\} \tag{4.27d}$$

The division by two appears in entries which appear twice in order to allow for a simple tensor network output equation in the next subsection. For the general case, these factors will appear as binomial coefficients. Proof of this decomposition follows through straightforward computations. In the next subsection the general case for this subsection and the tensor network output equation are presented.

4.3.2. THE GENERAL CASE

In this subsection we present the tensor networks of the LPV sub-Markov parameters (4.8) and the associated data matrix (4.9) for the general case. Additionally, we present the predictor-based data equation (4.14) using these tensor networks.

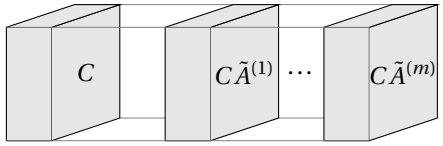
The classic form of this equation is:

$$y_k \approx CK_{(p)} Z_k + e_k, \tag{4.28}$$

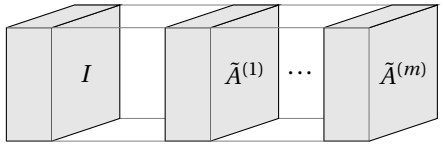
which has the bottleneck that the matrices $CK_{(p)}$ and Z_k suffer from the 'curse-of-dimensionality' (Definition 4.2.1).

Therefore we first present their tensor network decompositions. Define the cores of

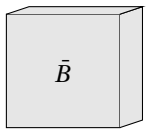
the tensor network of the LPV sub-Markov parameters in three-dimensional view as:

$\mathcal{A}^{(1)} =$ 

$\in \mathbb{R}^{l\times(m+1)\times n}$

$\mathcal{A}^{(s)} =$ 

$\in \mathbb{R}^{n\times(m+1)\times n}, \forall s \in \{2,\ldots,p-1\}$

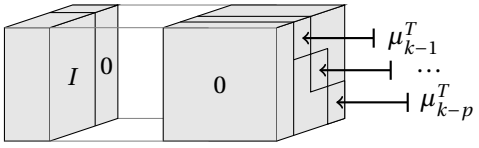
$\mathcal{A}^{(p)} =$ 

$\in \mathbb{R}^{n\times m(l+r)\times 1}$,

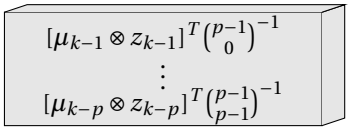
or equivalently,

$$\mathcal{A}^{(1)}(:,1,:) = C \tag{4.29a}$$
$$\mathcal{A}^{(1)}(:,1+b,:) = C\tilde{A}^{(b)} \quad ,\forall b \in \{1,\ldots,m\} \tag{4.29b}$$
$$\mathcal{A}^{(s)}(:,1,:) = I_n \quad ,\forall s \in \{2,\ldots,p-1\} \tag{4.29c}$$
$$\mathcal{A}^{(s)}(:,1+b,:) = \tilde{A}^{(b)} \quad ,\forall b \in \{1,\ldots,m\}, \forall s \in \{2,\ldots,p-1\} \tag{4.29d}$$
$$\mathcal{A}^{(p)} = \bar{B} \tag{4.29e}$$

Also, define the cores of the tensor network of the associated data matrix (per sample) in three-dimensional view as:

$\mathcal{B}^{(s)} =$ 

$, \in \mathbb{R}^{s\times(m+1)\times s+1}, \forall s \in \{1,\ldots,p-1\}$

$\mathcal{B}^{(p)} =$ 

$\in \mathbb{R}^{p\times m(l+r)\times 1}$,

where $[\mu_{k-j} \otimes z_{k-j}]^T$ is a row vector, and equivalently,

$$\mathcal{B}_k^{(s)}(:,1,:) = \begin{bmatrix} I_s & 0 \end{bmatrix} \in \mathbb{R}^{s \times s+1}$$
$$\forall s \in \{1,\ldots,p-1\} \tag{4.30a}$$

$$\mathcal{B}_k^{(s)}(:,1+b,:) = \begin{bmatrix} 0 & \mu_{k-1}^{(b)} & 0 & \cdots & 0 \\ 0 & 0 & \mu_{k-2}^{(b)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \mu_{k-s}^{(b)} \end{bmatrix} \in \mathbb{R}^{s \times s+1}$$
$$\forall b \in \{1,\ldots,m\}, \forall s \in \{1,\ldots,p-1\} \tag{4.30b}$$

$$\mathcal{B}_k^{(p)} = \begin{bmatrix} [\mu_{k-1} \otimes z_{k-1}]^T \binom{p-1}{0}^{-1} \\ \vdots \\ [\mu_{k-p} \otimes z_{k-p}]^T \binom{p-1}{p-1}^{-1} \end{bmatrix}$$
$$\in \mathbb{R}^{p \times m(l+r)}, \tag{4.30c}$$

Notice that these cores have this specific sparse form, because Kronecker products of subsequent scheduling sequence samples appear in the data (see for example (4.11)). Proof of both decompositions follows through straightforward computations. The binomial coefficients in this last core divide entries of the full tensor by how many times they appear in the full tensor. These coefficients may give finite precision problems for $p \geq 58$. These cores are defined like this to keep the following data equation simple.

Using these two tensor networks, the predictor-based data equation can be recast as:

$$y_k \approx CK_{(p)}Z_k + e_k = <\mathbf{T}, \mathbf{Z}_k> + e_k, \tag{4.31}$$

where $<*,*>$ is the inner product (Definition 4.2.2), and $\mathbf{T}$ and $\mathbf{Z}_k$ are defined by their tensor networks:

$$\mathbf{T} = g(\mathcal{A}^{(1)},\ldots,\mathcal{A}^{(p)})$$
$$\in \mathbb{R}^{l \times (m+1) \times \cdots \times (m+1) \times m(l+r)} \tag{4.32a}$$

$$\mathbf{Z}_k = g(\mathcal{B}_k^{(1)},\ldots,\mathcal{B}_k^{(p)})$$
$$\in \mathbb{R}^{1 \times (m+1) \times \cdots \times (m+1) \times m(l+r)} \tag{4.32b}$$

In detail, the full tensor $\mathbf{T}$ has $l(l+r)m(m+1)^{p-1}$ elements of which $l(l+r)\sum_{j=1}^{p} m^j$ (4.7) are unique. The advantage of this recast is that $<\mathbf{T},\mathbf{Z}_k>$ can be computed 'curse-of-dimensionality'-free in memory and computation using tensor networks (Definition 4.2.2).

In the next subsection the state-revealing matrix is recast using tensor networks.

4.3.3. THE STATE-REVEALING MATRIX

In this subsection we present how the (estimate) state-revealing matrix (4.20) can be constructed from the (estimate) parameter tensor network (4.31) and the data sequence

'curse-of-dimensionality'-free in memory and computation. Afterwards the (estimate) state sequence and state-space matrices can be obtained using Subsection 4.2.1.

Constructing the (estimate) state-revealing matrix R (4.20) in straight-forward manner from the (estimate) LPV sub-Markov parameters and associated data matrix is problematic, because they both suffer from the 'curse-of-dimensionality'. Therefore we present how it can be constructed with tensor networks using the operator $h()$ as:

$$R = h(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(p)}, \mu, z, f) \tag{4.33a}$$

$$R(\theta) = h(\mathcal{A}^{(1)}(\theta), \ldots, \mathcal{A}^{(p)}(\theta), \mu, z, f), \tag{4.33b}$$

where R and $\mathcal{A}^{(*)}$ are defined in (4.20) and (4.29) respectively, and the parametrization is discussed in detail in the next section. Because the definition of $h()$ is lengthy, its details are provided in Appendix 4.C. This equation allows computing the (estimate) state-revealing matrix 'curse-of-dimensionality'-free in memory and computation.

In the next section, the proposed method is presented.

4.4. THE PROPOSED METHOD

In the section the proposed method is presented. The proposed refinement predictor-based method uses a tensor network parametrization in order to obtain refined LPV state-space estimates 'curse-of-dimensionality'-free in memory and computation. This can be as exploitation of the underlying tensor structure.

4.4.1. PARAMETRIZATION

A tensor network parametrization of the LPV sub-Markov parameters is used by the proposed method. That is, the LPV sub-Markov parameter tensor (4.32) is parametrized as:

$$\mathbf{T}(\theta) = g(\mathcal{A}^{(1)}(\theta_1), \ldots, \mathcal{A}^{(p)}(\theta_p)), \tag{4.34}$$

with each core parametrized as black-box. Using the data equation (4.31), the model output can be described as:

$$y_k(\theta) = < \mathbf{T}(\theta), \mathbf{Z}_k > \tag{4.35}$$

Furthermore, the estimated state-revealing matrix (4.20) is:

$$R(\theta) = h(\mathcal{A}^{(1)}(\theta_1), \ldots, \mathcal{A}^{(p)}(\theta_p), \mu, z, f) \tag{4.36}$$

Notice that the system order is generally not known, so the sizes of the cores will depend on the model order $\tilde{n}$ instead. We underline that this parametrization is a free parametrization and not canonical. Furthermore the core and state estimates are modulo global state-coordinate transformation. In total there are $l(m+1)\tilde{n} + (\tilde{n}(m+1)\tilde{n})(p-2) + \tilde{n}m(l+r)$ parameters to estimate. Notice that the number of LPV sub-Markov parameters is generally many times larger ($l(l+r)\sum_{j=1}^{p} m^j$ (4.7)). The parameter tensor $\mathbf{T}(\theta)$ has over-parametrization: an LPV sub-Markov parameter can appear multiple times and each will be parametrized differently. The chosen parametrization allows for well-known (tensor network) optimization techniques.

The optimization of these parameters is performed using Alternating Least Squares (ALS) for tensor networks (Oseledets, 2011; Batselier et al., 2017). That is, we start with an initialization of the cores and iteratively update them one-by-one. Every update boils down to solving a least squares problem (4.55). Afterwards we make an orthogonalization step in order to improve the conditioning of the next sub-problem. This orthogonalization is also performed on the initialization of the cores. We refer to (Oseledets, 2011; Batselier et al., 2017; Chen et al., 2016) for this common step. For this orthogonalization step, some assumptions are made on the sizes of the cores. Let $\mathcal{A}^{(s)}(\theta)$ be $\in \mathbb{R}^{a\times b\times c}$. Then for $1 \leq s \leq p-1$ we assume $ab \geq c$, and for $2 \leq s \leq p$ we assume $bc \geq a$.

This optimization approach enjoys some nice properties. Firstly, the optimization has local linear convergence under some mild condition (Holtz et al., 2012; Rohwedder and Uschmajew, 2013). Secondly, the condition number of the least squares problem of every sub-problem has upper bounds (Holtz et al., 2012). The latter result allows exchanging the assumption that each sub-problem is well-conditioned by the assumption that this upper bound holds (Holtz et al., 2012) and is finite. We show how to compute these bounds for our problem 'curse-of-dimensionality'-free in memory and computation in Appendix 4.B.

In the next subsection the algorithm of the proposed method is summarized.

4.4.2. ALGORITHM

Algorithm 4.4.1 *The proposed method*

Input: an initial estimate of the LPV state-space matrices, μ, z, p, f , termination conditions

Output: a refined estimate of the LPV state-space matrices

1. *Initialize the tensor network cores $\mathcal{A}(\theta)$* (4.34)*. This can be done using an initial estimate of the LPV state-space matrices and the formulas* (4.29)*. This initial estimate can be obtained from a ('curse-of-dimensionality'-free in memory and computation) convex method, such as proposed by Gunes et al. (2018); van Wingerden and Verhaegen (2009).*

2. *Compute the tensor network cores $\mathcal{B}$ of the data* (4.30)

3. *Optimize the tensor network cores $\mathcal{A}(\theta)$ using the tensor network ALS (Holtz et al., 2012) and Subsection 4.4.1 until the termination conditions (discussed in the next paragraph) are met. The sub-problems of this ALS are given in Appendix 4.A.*

4. *Use the estimate cores to compute the estimate state-revealing matrix using* (4.33)

5. *Use an SVD to obtain an estimate of the states as described in Subsection 4.2.1*

6. *Solve two least squares problem to obtain a (refined) estimate of the LPV state-space matrices as described in van Wingerden and Verhaegen (2009)*

The entire algorithm is 'curse-of-dimensionality'-free in memory and computation. Some details on the algorithm are worth mentioning. Firstly notice that the initial estimate also provides a well-supported choice of the tensor network ranks, so

these ranks can be kept fixed during optimization. It is worth remarking that methods without fixed ranks exist such as the Density Matrix Renormalization Group (DMRG) algorithm (White, 1992). Secondly, there are several ways to terminate the optimization. One option is to terminate when the residual changes less than 0.1% after updating all cores forward and backward. Another option is to terminate after a few such 'sweeps'. We do not recommend a termination condition on the absolute residual itself for this application, as the LPV data equation (4.14) can have non-negligible bias. Furthermore, in the simulations we will solve the least squares problem for every update with added Tikhonov regularization. More specifically, instead of solving $||\bar{Y} - \bar{V}^{(s)}\text{vec}(\mathcal{A}^{(s)})||_F^2$ we solve $||\bar{Y} - \bar{V}^{(s)}\text{vec}(\mathcal{A}^{(s)})||_F^2 + \lambda||\text{vec}(\mathcal{A}^{(s)})||_F^2$ where the definitions are given in Appendix 4.A. The tuning parameter λ is chosen using Generalized Cross Validation (GCV) (Golub et al., 1979). Additionally, the computation cost of this algorithm scales favourably with the past window as a fifth power. This is dominated by the computation cost of the inner products in the ALS step. In the next section simulation results are presented to evaluate the performance of the proposed method.

4.5. SIMULATION RESULTS

In this section we illustrate the performance of the proposed method with simulation results.

4.5.1. SIMULATION SETTINGS

In this section the general simulation settings, constant over every case, are presented.

The following information is passed to the identification methods. All methods are passed the input, output and scheduling sequences. The model order is chosen equal to the system order. The future window in any SVD step is chosen equal to the past window. The evaluated methods are all global LPV identification methods for known arbitrary scheduling sequences. All cases generate data based on a finite-dimensional discrete affine state-space LPV system. Together with the proposed non-convex method, the convex LPV-PBSID$_{opt}$ (kernel) method by van Wingerden and Verhaegen (2009), the non-convex Predictor-Based Tensor Regression (PBTR) method by Gunes et al. (2017a) and the non-convex polynomial method by Verdult et al. (2003) are evaluated. More specifically, the LPV-PBSID$_{opt}$ with its own Tikhonov regularization and Generalized Cross Validation (GCV), and the output error variant of the polynomial method (Verdult et al., 2003) is used. The proposed method is terminated after five sweeps. All the non-convex methods are initialized from the estimate of the convex method. Additionally, results are presented of the proposed method initialized in 'random' manner as follows. Firstly, m (4.3) random, stable LTI systems with the chosen model order and matching input and output counts are generated. Secondly, these m LTI systems are combined in straight-forward manner into an LPV system. Thirdly, this LPV system is regarded as an initial estimate. The matrix K is generated in the same way as B. These methods are or will soon be available with the PBSID toolbox (van Wingerden and Verhaegen, 2009). The convex method (Gunes et al., 2018) from previous work is not evaluated, because it is not related to the core contributions of the current paper. All results are based on 100 Monte Carlo simulations. For each Monte Carlo simulation, a different realization of

both the input and the innovation vector is used, while the scheduling sequence is kept the same.

The produced estimates are then scored using the Variance Accounted For (VAF) criterion on a validation data set. That is, fresh data not available to the methods is used which has different realizations of both the input and the innovation vector. The VAF for single-output systems is defined as follows (van Wingerden and Verhaegen, 2009):

$$\text{VAF}(\bar{y}_k, \hat{y}_k) = \max\left\{1 - \frac{var(\bar{y}_k - \hat{y}_k)}{var(\bar{y}_k)}, 0\right\}100\%$$

The variable $\bar{y}_k$ denotes the noise-free simulated output of the system, and similarly $\hat{y}_k$ denotes the noise-free (simulated) model output. The operator $var(*)$ returns the variance of its argument. If a non-convex refinement method returns an estimate with identification VAF fifteen percent lower than that of its initialization, then the refined estimate is rejected and substituted by the initialization. Notice that this only involves identification data. Also notice that this does not prevent local minima, but only serves to reject known poor optimization results. Finally, the computations are performed on an Intel i7 quad-core processor running at 2.7GHz with 8 GB RAM, and the computation times are provided. In the next subsections, we present several cases and the parameter counts.

4.5.2. SIMULATION RESULTS CASE 1

In this subsection, a case which relates to the flapping dynamics of a wind turbine is used to evaluate the performance of the proposed method and existing methods. This case has been used before in (Felici et al., 2007; van Wingerden and Verhaegen, 2009). Consider the following LPV state-space system (4.3):

$$[A^{(1)}, A^{(2)}] = \left[\begin{array}{cc|cc} 0 & 0.0734 & -0.0021 & 0 \\ -6.5229 & 0.4997 & -0.0138 & 0.5196 \end{array}\right],$$

$$[B^{(1)}, B^{(2)}] = \left[\begin{array}{c|c} -0.7221 & 0 \\ -9.6277 & 0 \end{array}\right], C = \begin{bmatrix} 1 & 0 \end{bmatrix},$$

where D is zero and K is LPV. The matrix $K^{(i)}$ for $i = \{1, \ldots, m\}$ has been obtained from the Discrete Algebraic Ricatti Equation (DARE) using $A^{(i)}$ and C and identity covariance of the concatenated process and measurement noise. The input u_k is chosen as white noise with unit power and and the innovation e_k is also chosen as white noise with unit power. The past window is 6. The data size N is 200.

The system is evaluated at the affine scheduling sequence:

$$\mu_k^{(2)} = \cos(2\pi k \frac{10}{N})/2 + 0.2,$$

The results are shown in Table 4.1, Fig 4.1 and 4.2. The average computation times are 0.033, 15, 0.82, 10 and 9.4 seconds for respectively the LPV-PBSID$_{opt}$, the PBTR, the polynomial and the proposed method (for pseudo-random and normal initialization) per Monte Carlo simulation.

Method	VAF%
LPV-PBSID$_{opt}$ (kernel)	92.2 ±0.5
PBTR	91.1 ±0.6
Polynomial method	94.6 ±0.3
Proposed method (r.i.)	94.8 ±0.3
Proposed method	95.8 ±0.2

Table 4.1: Mean VAF for different methods for Case 1. The text 'r.i.' indicates random initialization, as discussed in Subsection 4.5.1.

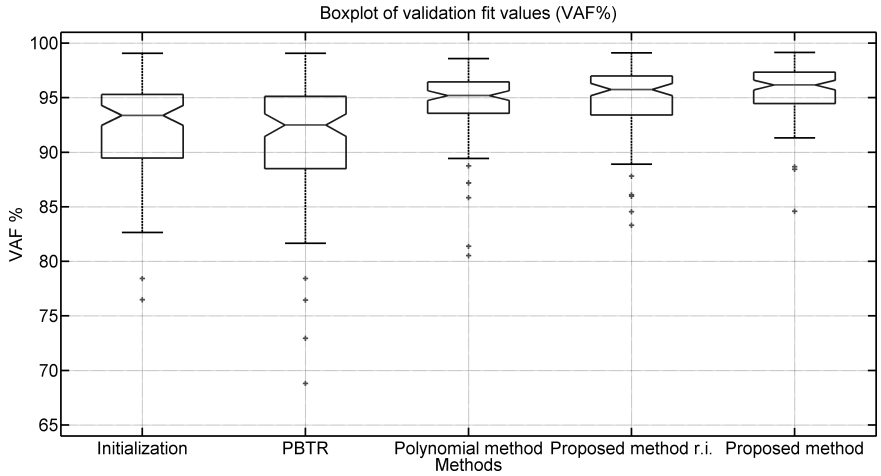


Figure 4.1: This figure shows a box-plot of the validation VAF of the 100 Monte Carlo simulations for five methods. The initialization method is LPV-PBSID$_{opt}$ (kernel), and the other four are refinement methods. The text 'r.i.' indicates random initialization, as discussed in Subsection 4.5.1.

From Table 4.1, Fig 4.1 and 4.2 it is visible that for this case the proposed method and the polynomial method can refine the initial estimates successfully. Two additional, minor results are that the proposed method may converge quickly and perform well for random initialization.

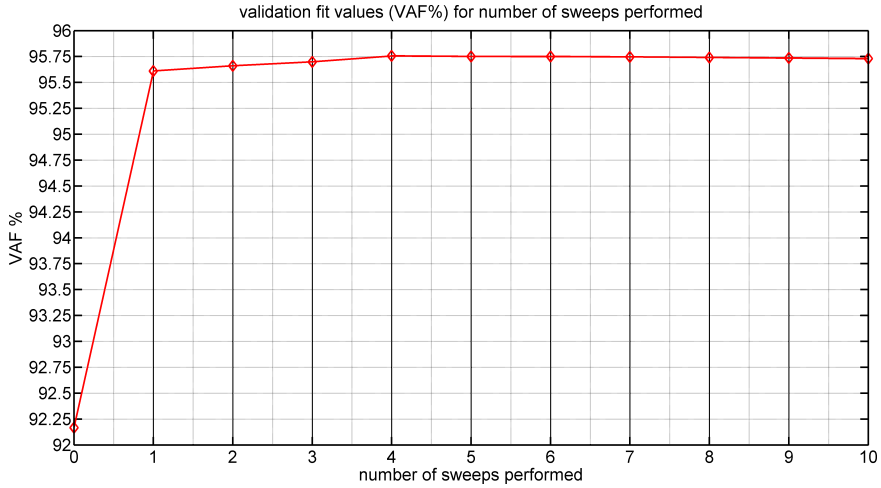


Figure 4.2: This figure shows how the validation VAF of the proposed method, averaged over the Monte Carlo simulations, is affected by the number of sweeps. The VAF at 0 sweeps performed is the initialization. This sweep is defined in Subsection 4.4.1.

4.5.3. SIMULATION RESULTS CASE 2

In this subsection, the proposed method is shown to work for a MIMO case with higher system order. Consider the following LPV state-space system (4.3) with:

$$A^{(1)} = \begin{bmatrix} \frac{-15}{30} & 0 & 0 & 0 \\ \frac{2}{30} & \frac{-17}{30} & \frac{-4}{30} & \frac{38}{30} \\ \frac{2}{30} & \frac{-2}{30} & \frac{-19}{30} & \frac{2}{30} \\ 0 & 0 & 0 & \frac{21}{30} \end{bmatrix}, A^{(2)} = \begin{bmatrix} \frac{24}{30} & 0 & 0 & 0 \\ \frac{9}{30} & \frac{15}{30} & \frac{-18}{30} & \frac{-18}{30} \\ \frac{9}{30} & \frac{-9}{30} & \frac{6}{30} & \frac{9}{30} \\ 0 & 0 & 0 & \frac{-1}{30} \end{bmatrix}$$

$$A^{(3)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{-2}{30} & \frac{2}{30} & \frac{4}{30} & \frac{-2}{30} \\ \frac{-2}{30} & \frac{2}{30} & \frac{4}{30} & \frac{-1}{30} \\ 0 & 0 & 0 & 0 \end{bmatrix}, A^{(4)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{6}{30} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{6}{30} \end{bmatrix},$$

and

$$B^{(1)} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, B^{(2)} = B^{(3)} = B^{(4)} = B^{(1)}/4$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

where D is zero and K is LPV. The matrix $K^{(i)}$ for $i = \{1, \ldots, m\}$ has been obtained from the Discrete Algebraic Ricatti Equation (DARE) using $A^{(i)}$ and C and identity covariance of the concatenated process and measurement noise. Every input signal is chosen as white noise with unit power and and every innovation signal is chosen as white noise with half a unit power. The past window is 6. The data size N is 200.

The system is evaluated at the affine scheduling sequence:

$$\mu_k^{(2)} = \text{sawtooth}(1 + 2\pi k\frac{1}{N})/3 + 0.2$$
$$\mu_k^{(3)} = \text{sawtooth}(-1 + 2\pi k\frac{1}{N}/3.5 + 0.5\pi)$$
$$\mu_k^{(4)} = \cos(2\pi k\frac{1}{N}/3) + 0.2,$$

where 'sawtooth' is a sawtooth wave function with peaks of -1 and 1. It is -1 when its argument is a multiple of 2π. The results are shown in Table 4.2, Fig. 4.3 and 4.4. The average computation times are 0.020 and 13 seconds for respectively the LPV-PBSID$_{opt}$ method and the proposed method per Monte Carlo simulation.

Method	Output Channel	VAF%
LPV-PBSID$_{opt}$ (kernel)	1	67.3 ±2.9
	2	54.5 ±3.0
	3	60.3 ±3.3
Proposed method	1	79.7 ±1.9
	2	73.9 ±2.3
	3	74.2 ±2.4

Table 4.2: Mean VAF for different methods for Case 2.

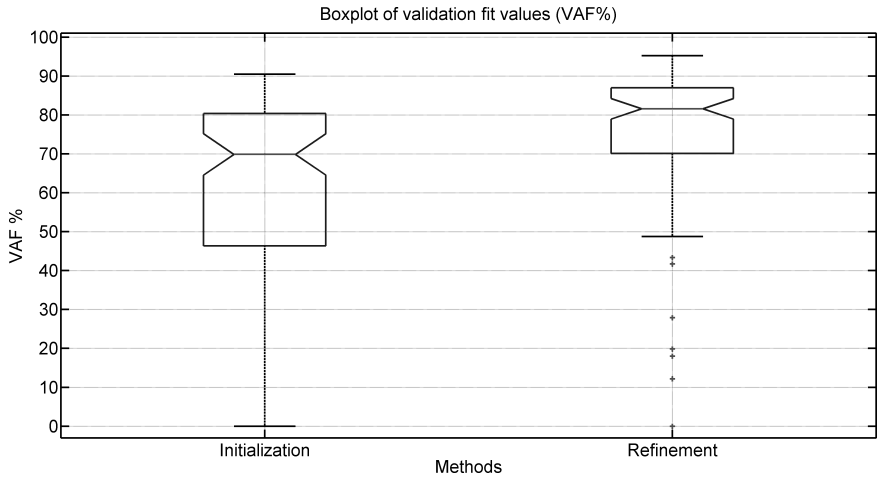


Figure 4.3: This figure shows a box-plot of the validation VAF of the 100 Monte Carlo simulations for two methods. The initialization method is LPV-PBSID$_{opt}$ (kernel), and the refinement method is the proposed method.

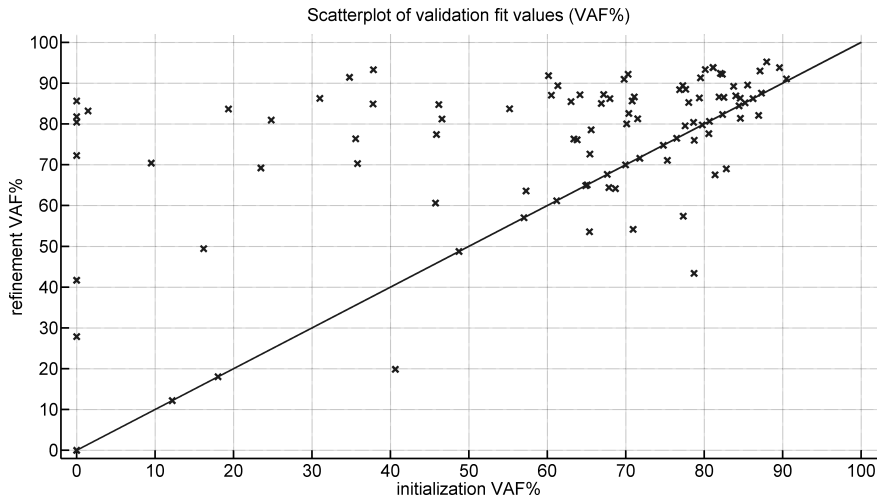


Figure 4.4: This figure shows a scatter-plot of the validation VAF of the 100 Monte Carlo simulations for two methods. The initialization method is LPV-PBSID$_{opt}$ (kernel), and the refinement method is the proposed method.

From the results of Table 4.2, Fig. 4.3 and 4.4, it is visible that the proposed method can refine initial estimates for this case. We do remark that a few 'refined' estimates have decreased VAF, which is possible due to the two compared methods optimizing different objective functions.

4.5.4. PARAMETER COUNTS

We present the parameter counts of several methods in Table 4.3.

This table shows that a straight-forward black-box parametrization, like LPV-PBSID$_{opt}$ without kernels, results in a parameter count which scales exponentially. A more condense parametrization can be obtained by using kernels van Wingerden and Verhaegen (2009) or a non-linear parametrization. The parameter count of the proposed method is comparable with the one of PBTR and larger than the one of the polynomial method. The parameter counts for Case 2 show that the parameter counts of the evaluated methods is much smaller than is the case for a straight-forward approach.

Method	Parameter count formula	Case 1	Case 2
LPV-PBSID$_{opt}$ (primal)	$l(l+r)\sum_{j=1}^{p} m^j$	252	81900
LPV-PBSID$_{opt}$ (kernel)	lN	200	600
PBTR	$nl+n(l+r)m+n^2m(p-1)$	50	412
Polynomial method	$nl+nrm+n^2m$	14	108
Proposed method	$l(m+1)n+n^2(m+1)(p-2)$ $+n(l+r)m$	62	460

Table 4.3: Comparison of the parameter counts for the (first) estimation step, for model order equal to system order.

4.6. CONCLUSIONS

In this paper it was shown that the MIMO LPV identification problem with state-space matrices admits a tensor network description. That is, the LPV sub-Markov parameters, data and state-revealing matrix were condensely and exactly presented as tensor networks. Additionally, it was shown how to obtain the (estimate) state-revealing matrix directly from (estimate) tensor networks. These results provided a tensor network perspective on the MIMO LPV identification problem. Then a refinement method was proposed which is 'curse-of-dimensionality'-free in memory and computation and has conditioning guarantees. The performance of the proposed method was illustrated using simulation cases and additionally compared with existing methods.

4.A. DERIVATION OF THE LEAST SQUARES PROBLEM

The proposed method involves the optimization of a tensor network using ALS. At every update of a tensor network core, a least squares problem must be solved. In this appendix we derive that least squares problem.

Using the algorithm for the inner products of two tensor networks (Oseledets, 2011)

and (4.29) (4.30), we can state that:

$$< \mathbf{T}, \mathbf{Z}_k >= \prod_{s=1}^{s=p} \sum_{i_s=1}^{n_s} \mathcal{A}^{(s)}(:, i_s, :) \otimes \mathcal{B}_k^{(s)}(:, i_s, :) \tag{4.40}$$

where n_s is:

$$n_s = m + 1 \text{ for } s \leq p - 1, \text{ otherwise } m(l + r) \tag{4.41}$$

and the operator $\prod$ is defined as:

$$\prod_{i=1}^{d} M_i = M_1 M_2 \dots M_d \tag{4.42}$$

Then we can isolate the s-th core:

$$< \mathbf{T}, \mathbf{Z}_k >= V_{-,k}^{(s)} \Big(\sum_{i_s=1}^{n_s} \mathcal{A}^{(s)}(:, i_s, :) \otimes \mathcal{B}_k^{(s)}(:, i_s, :) \Big) V_{+,k}^{(s)}, \tag{4.43}$$

where $\mathcal{A}^{(s)} \in \mathbb{R}^{\bar{m} \times n_s \times \bar{n}}$, $\mathcal{B}_k^{(s)} \in \mathbb{R}^{\bar{p} \times n_s \times \bar{q}}$, and

$$\begin{aligned} V_{-,k}^{(s)} &= I_{\bar{m}\bar{p}} \text{ for } s = 1, \\ &\prod_{t=1}^{t=s-1} \sum_{i_t=1}^{n_t} \mathcal{A}^{(t)}(:, i_t, :) \otimes \mathcal{B}_k^{(t)}(:, i_t, :) \text{ otherwise,} \end{aligned} \tag{4.44}$$

and

$$\begin{aligned} V_{+,k}^{(s)} &= I_{\bar{n}\bar{q}} \text{ for } s = p, \\ &\prod_{t=s+1}^{t=p} \sum_{i_t=1}^{n_t} \mathcal{A}^{(t)}(:, i_t, :) \otimes \mathcal{B}_k^{(t)}(:, i_t, :) \text{ otherwise} \end{aligned} \tag{4.45}$$

Using the Kronecker algebra trick in (Batselier et al., 2017), we can rewrite:

$$\begin{aligned} < \mathbf{T}, \mathbf{Z}_k >= &\Big((V_{+,k}^{(s)})^T \otimes V_{-,k}^{(s)} \Big) \\ &\sum_{i_s=1}^{n_s} \text{vec}\Big(\mathcal{A}^{(s)}(:, i_s, :) \otimes \mathcal{B}_k^{(s)}(:, i_s, :) \Big), \end{aligned} \tag{4.46}$$

where we additionally pulled the summer out of the vectorization operator vec$(*)$. Next we use the following equivalence (Turkington, 2013). For a matrix M of size $\bar{m}$-by-$\bar{n}$ and a matrix N of size $\bar{p}$-by-$\bar{q}$:

$$\text{vec}(M \otimes N) = (I_{\bar{n}} \otimes \bar{K}_{\bar{q}\bar{m}} \otimes I_{\bar{p}})(I_{\bar{m}\bar{n}} \otimes \text{vec}(N)) \text{vec}(M), \tag{4.47}$$

where the matrix $\bar{K}$ is defined such that:

$$\bar{K}_{\bar{m}\bar{n}} \text{vec}(M) = \text{vec}(M^T) \tag{4.48}$$

Now we can fill in:

$$\begin{aligned} M &= \mathcal{A}^{(s)}(:, i_s, :) \\ N &= \mathcal{B}_k^{(s)}(:, i_s, :) \end{aligned} \tag{4.49}$$

to obtain:

$$\begin{aligned} <\mathbf{T},\mathbf{Z}_k> = &\left((V_{+,k}^{(s)})^T \otimes V_{-,k}^{(s)}\right)(I_{\bar{n}} \otimes \bar{K}_{\bar{q}\bar{m}} \otimes I_{\bar{p}}) \sum_{i_s=1}^{n_s} \\ &(I_{\bar{m}\bar{n}} \otimes \text{vec}(\mathcal{B}_k^{(s)}(:,i_s,:))\text{vec}(\mathcal{A}^{(s)}(:,i_s,:)) \end{aligned} \tag{4.50}$$

Notice that the sizes are constant over i_s. Next we can rewrite without summation:

$$\begin{aligned} <\mathbf{T},\mathbf{Z}_k> = &\left((V_{+,k}^{(s)})^T \otimes V_{-,k}^{(s)}\right)(I_{\bar{n}} \otimes \bar{K}_{\bar{q}\bar{m}} \otimes I_{\bar{p}}) \\ &\begin{bmatrix} I_{\bar{m}\bar{n}} \otimes \text{vec}(\mathcal{B}_k^{(s)}(:,1,:)) & \dots \end{bmatrix} \\ &\begin{bmatrix} \text{vec}(\mathcal{A}^{(s)}(:,1,:)) \\ \vdots \\ \text{vec}(\mathcal{A}^{(s)}(:,n_s,:)) \end{bmatrix} \end{aligned} \tag{4.51}$$

We can further simplify this equation by reorganizing the right-most matrix into $\text{vec}(\mathcal{A}^{(s)})$ as follows. Define the matrix $\bar{V}_k^{(s)}$ in pseudo-code:

$$\begin{aligned} \bar{V}_k^{(s)} = &\text{reshape}(\text{permute}(\text{reshape}(\\ &\left((V_{+,k}^{(s)})^T \otimes V_{-,k}^{(s)}\right)(I_{\bar{n}} \otimes \bar{K}_{\bar{q}\bar{m}} \otimes I_{\bar{p}}) \\ &\begin{bmatrix} I_{\bar{m}\bar{n}} \otimes \text{vec}(\mathcal{B}_k^{(s)}(:,1,:)) & \dots \end{bmatrix}, \\ &[l,\bar{m},n_s,\bar{n}]),[1,2,4,3]),[l,\bar{m}n_s\bar{n}]), \end{aligned} \tag{4.52}$$

where the two functions are defined as follows. The 'reshape' function changes the size of the dimensions of its first argument as specified in its second argument. The ordering of the elements is not changed. The 'permute' function changes the indexing of its first argument as specified in its second argument. Then, we obtain (using (4.31)):

$$y_k \approx <\mathbf{T},\mathbf{Z}_k> + e_k = \bar{V}_k^{(s)}\text{vec}(\mathcal{A}^{(s)}) + e_k \tag{4.53}$$

We can stack this over all samples. Define:

$$\bar{V}^{(s)} = \begin{bmatrix} \bar{V}_{p+1}^{(s)} \\ \vdots \\ \bar{V}_N^{(s)} \end{bmatrix} \in \mathbb{R}^{l(N-p)\times \bar{m}n_s\bar{n}}, \bar{Y} = \begin{bmatrix} y_{p+1} \\ \vdots \\ y_N \end{bmatrix}, \bar{E} = \begin{bmatrix} e_{p+1} \\ \vdots \\ e_N \end{bmatrix} \in \mathbb{R}^{l(N-p)}, \tag{4.54}$$

such that:

$$\bar{Y} = \bar{V}^{(s)}\text{vec}(\mathcal{A}^{(s)}) + \bar{E}, \tag{4.55}$$

where it is assumed that $l(N-p)$ is greater than or equal to the number of elements of each $\mathcal{A}^{(s)}$. Additionally, upper bounds are derived on the condition number of $\bar{V}^{(s)}$ in Appendix 4.B. This is the least squares problem that has to be solved when updating a core.

4.B. THE CONDITION GUARANTEE

The proposed method uses ALS to optimize its tensor network cores. This involves solving a number of sub-problems (4.55), whose conditioning depends on the condition number of $\bar{V}^{(s)}$. In this appendix we derive an upper bound on this condition number using Holtz et al. (2012). Notice that this property is in part due to the orthogonalization steps of tensor network ALS. Additionally, we provide a means to compute it 'curse-of-dimensionality'-free in memory and computation.

Firstly, define the condition number of a matrix as the ratio of the largest singular value of that matrix to the smallest singular value. If the smallest singular value is zero, then the condition number is by convention infinite. Let "cond()" be an operator whose output is the condition number of its argument.

Define using (4.31):

$$\check{Y} = \begin{bmatrix} y_{p+1}^T \\ \vdots \\ y_N^T \end{bmatrix}, \check{E} = \begin{bmatrix} e_{p+1}^T \\ \vdots \\ e_N^T \end{bmatrix}, \check{\mathbf{Z}} = \begin{bmatrix} \text{vec}(\mathbf{Z}_{p+1})^T \\ \vdots \\ \text{vec}(\mathbf{Z}_N)^T \end{bmatrix}, \tag{4.56}$$

such that we can rewrite (4.31) as:

$$\check{Y} \approx (CK_{(p)}Z)^T + \check{E} = \check{\mathbf{Z}}(\mathbf{T}_{<1>})^T + \check{E} \tag{4.57}$$

Additionally, we assume that the chosen initialization of the tensor network cores results in all $\bar{V}^{(s)}$ (4.55) being invertible at the start of the optimization. Then, using the results of (Holtz et al., 2012), we can now state that the condition number of each $\bar{V}^{(s)}$ (4.55) is upper bounded by the condition number of the matrix $\check{\mathbf{Z}}$.

Next, we provide means to compute the latter condition number in a way which is 'curse-of-dimensionality'-free in memory and computation using tensor trains. Notice that $\check{\mathbf{Z}}$ suffers from the 'curse-of-dimensionality', such that the computation of its condition number is problematic. However, since $\check{\mathbf{Z}}$ only has $N-p$ rows, the following trick can be applied. For any wide matrix M, $\text{cond}(M) = \text{cond}((MM^T)^{1/2})$ holds. Notice that MM^T is smaller than the wide M. Hence we use that:

$$\text{cond}(\check{\mathbf{Z}}) = \text{cond}((\check{\mathbf{Z}}\check{\mathbf{Z}}^T)^{1/2}), \tag{4.58}$$

where the size of $(\check{\mathbf{Z}}\check{\mathbf{Z}}^T)^{1/2}$ is only $N-p$-by-$N-p$. Hence $\text{cond}((\check{\mathbf{Z}}\check{\mathbf{Z}}^T)^{1/2})$ can be computed if $(\check{\mathbf{Z}}\check{\mathbf{Z}}^T)$ is available. In other words, the size and square root do not induce any 'curse-of-dimensionality', such that the problem reduces to computing the following equation 'curse-of-dimensionality'-free in memory and computation. This reduced problem can be tackled using tensor trains. Using (4.56):

$$[\check{\mathbf{Z}}\check{\mathbf{Z}}^T](i,j) = \text{vec}(\mathbf{Z}_{p+i})^T \text{vec}(\mathbf{Z}_{p+j}), \tag{4.59}$$

or equivalently:

$$[\check{\mathbf{Z}}\check{\mathbf{Z}}^T](i,j) = <\mathbf{Z}_{p+i}, \mathbf{Z}_{p+j}> \tag{4.60}$$

which is an inner product of two tensor trains (Definition 4.2.2). This inner product and, with it, the upper bound on the condition number can be computed 'curse-of-dimensionality'-free in memory and computation.

4.C. THE OPERATOR FOR THE STATE-REVEALING MATRIX

In Subsection 4.2.1 it has been discussed how the estimate (LPV sub-Markov) parameters can be used to obtain an estimate of the state sequence. This is done by constructing the estimate state-revealing matrix $R(\theta)$ and taking its SVD. In this appendix, we show how to obtain this matrix from the estimate tensor networks 'curse-of-dimensionality'-free in memory and computation.

Recall that R has been defined as (4.20):

$$R = \begin{bmatrix} CL_p & CL_{p-1} & \cdots & CL_1 \\ 0 & C\tilde{A}^{(1)}L_{p-1} & \cdots & C\tilde{A}^{(1)}L_1 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & & C\big(\tilde{A}^{(1)}\big)^{f-1}L_1 \end{bmatrix} Z \in \mathbb{R}^{fl \times N-p} \tag{4.61}$$

Notice that both matrices of the product suffer from the 'curse-of-dimensionality'. However, we can avoid this problem by using the tenor networks as in (4.33) :

$$R = h(\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(p)}, \mu, z, f) \tag{4.62a}$$

$$R(\theta) = h(\mathcal{A}^{(1)}(\theta), \ldots, \mathcal{A}^{(p)}(\theta), \mu, z, f), \tag{4.62b}$$

where some operator $h(*)$ is used which is 'curse-of-dimensionality'-free in memory and computation.

This operator is non-trivial, because of the following reason. The rows of R involve subsets of the LPV sub-Markov parameters and the relation of these subsets to the tensor network decomposition is complex. Firstly, the LPV sub-Markov parameters appear multiple times in the parameter tensor and require averaging. Secondly, the tensor network cores contain identity matrices which add further ambiguity. The remainder of this appendix provides details on the functionality of this operator. The operator is presented formally in Algorithm 4.C.2.

The operator $h(*)$ consists of several loops. The major loop is over the outputs. That is, we split the full problem into l single-output sub-problems. Then we solve each sub-problem individually and merge the results. How we split and merge is presented in full detail in Algorithm 4.C.2. How we solve each sub-problem will be introduced next. In the remainder of this appendix we consider one such sub-problem with output number o for clarity.

For each such sub-problem, we compute the state-revealing matrix (4.20) with C replaced by $C_o = C(o,:)$. The top row of this matrix is $C_o K_{(p)} Z$ and can be computed efficiently using inner products of tensor networks (4.31), where C is substituted by C_o. The other rows are more involved.

The difficulty of the other rows is that we do not have a single inner product, because these rows involve only subsets of the LPV sub-Markov parameters. Namely, row number t only involves LPV sub-Markov parameters whose product begins with $C_o(\tilde{A}^{(1)})^{t-1}$ (4.20). Therefore we compute these rows in two parts. The first part relates to the required begin of the product, and the second part relates to the 'free' part.

Before presenting the computation, we first provide insight into how LPV sub-Markov parameters appear multiple times in the parameter tensor and how to deal with

this. The reason LPV sub-Markov parameters appear multiple times in the parameter tensor, is that they can be constructed in different ways from the tensor train cores. For example consider the (1,2,1)-th and (2,1,1)-th entry of the estimate parameter tensor for three cores:

$$\begin{aligned} \mathbf{T}(1,2,1)(\theta) = &\quad [C_o](\theta_1) \quad [\tilde{A}^{(1)}](\theta_2) \quad [B^{(1)}](\theta_3) \\ \mathbf{T}(2,1,1)(\theta) = &\quad [C_o\tilde{A}^{(1)}](\theta_1) \quad [I](\theta_2) \quad [B^{(1)}](\theta_3) \end{aligned} \tag{4.63}$$

Both relate to the same LPV sub-Markov parameter. To be exact, LPV sub-Markov parameters with c many $\tilde{A}^{(*)}$'s appear $\binom{p-1}{c}$ times. This phenomenon poses a problem, as we have several estimates of the same LPV sub-Markov parameter. It turns out that discarding all but one estimate is not viable, because it changes the model output. However, averaging estimates does not change the model output and removes the ambiguity. This is possible because the estimates are multiplied with the same data during the computation of the model output. We will explain later in this subsection how to perform this averaging efficiently.

First we define the matrix S in order to store all possible ways of fixing the cores to obtain the subset of LPV sub-Markov parameters relevant to one row of the state-revealing matrix. Let the number of cores we fix be δ. Then we need to fix δ cores, of which $t-1$ cores to $\tilde{A}^{(1)}$ and the rest as I. The last core we have to fix as $\tilde{A}^{(1)}$, for reasons we will explain later in this subsection. This becomes a pick $t-2$ cores out of $\delta-1$ cores problem. This can be done in $\binom{\delta-1}{t-2}$ ways. For every possible way, we put a row in S where the s-th element of that row is a 1 if core s is fixed at $\tilde{A}^{(1)}$ and a 0 otherwise for s is one to δ. We summarize this algorithm:

Algorithm 4.C.1 $S = \mathcal{S}(\delta, t)$

For every possible way of picking $t-2$ cores out of $\delta-1$ cores, put a row in S, where the s-th element of that row is a 1 *if core s is picked and a* 0 *otherwise. Let the δ-th element of that row be* 1*. This results in a $\binom{\delta-1}{t-2}$-by-δ matrix.*

We illustrate the result of the algorithm with an example:

$$\mathcal{S}(\delta = 4, t = 4) = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \tag{4.64}$$

In this paragraph we discuss why we need δ in the computations. The LPV sub-Markov parameters related to the t-th row all start with $C(\tilde{A}^{(1)})^{t-1}$, and we can isolate these by fixing cores. This we can do by fixing the first $t-1$ cores as $C\tilde{A}^{(1)}$ and $\tilde{A}^{(1)}$. The remaining cores are 'free'. We can also do this while fixing more cores, by fixing some at I. That can also leave a number of cores 'free'. Therefore to capture all estimates of the same LPV sub-Markov parameter, we have to consider $\delta = t-1$ to $p-1$. Additionally, to avoid counting estimates double, we let the last fixed core always be fixed at $\tilde{A}^{(1)}$. Now we return to how to perform the computations of the operator $h(*)$.

The first part of the computation relates to the fixed cores. The fixed cores are simply matrices. More specifically, they are slices of the cores corresponding to either I or $\tilde{A}^{(1)}$.

They form a series of matrix products. This product returns a row vector, as we consider one output at a time. We define this product as $\bar{L}$.

The second part of the computation relates to the 'free' cores. Notice that for the top row we had no fixed cores and directly used a tensor network inner product. Here we again use inner products. We form a reduced tensor network from the free cores. To match this smaller size tensor network in size, we also define a reduced data tensor network. This tensor network can simply be computed using (4.30) with p substituted by $p-\delta$, and the following change.

Additionally, we have to change the duplication correction factors (4.30) to account for the effect of δ not being just zero. Therefore we define the matrix D_r:

$$D_r = \begin{bmatrix} \binom{p-(t-1)}{1}\binom{p-\delta-1}{0}^{-1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \binom{p-(t-1)}{p-\delta}\binom{p-\delta-1}{p-\delta-1}^{-1} \end{bmatrix}, \tag{4.65}$$

such that changing the last data tensor network core as:

$$\check{\mathcal{B}}_k^{(p-\delta)} = D_r \check{\mathcal{B}}_k^{(p-\delta)} \tag{4.66}$$

ensures the correct duplication correction factors for the computation of the state-revealing matrix. Notice that $\check{\mathcal{B}}_k^{(p-\delta)}$ is a matrix. This then forms the tensor network of the reduced data tensor.

Then we combine these computations to obtain the state-revealing matrix of the sub-problem. Namely, we post-multiply L with the inner product of the reduced parameter and reduced data tensor networks. This is done over several loops, over: the sample, row, δ and rows of S. We remark that the operator $R(*)$ is 'curse-of-dimensionality'-free in memory and computation.

Next we summarize the computation of the operator $h(*)$ in the following algorithm in pseudo-code.

Algorithm 4.C.2 *The operator $h(*)$*
Input: (estimate) $\mathcal{A}^{(1)},\ldots,\mathcal{A}^{(p)},\mu,z,f$
Output: (estimate) state-revealing matrix R

1. $R = \bar{0} \in \mathbb{R}^{fl\times N-p}$
2. *for* $o = 1$ *to* l *do*
3. $\bar{\mathcal{A}}_o^{(1)} = \mathcal{A}^{(1)}(o,:,:)$
4. $R^{(o)} = \bar{h}(\bar{\mathcal{A}}^{(1)},\mathcal{A}^{(2)},\ldots,\mathcal{A}^{(p)},\mu,z,f)$
   using Algorithm 4.C.3
5. $v = o:l:fl$
6. $R(v,:) = R^{(o)}$

7. end for

Algorithm 4.C.3 *The operator* $\bar{h}(*)$
Input: (estimate) $\bar{\mathcal{A}}^{(1)}, \mathcal{A}^{(2)}, \ldots, \mathcal{A}^{(p)}, \mu, z, f$
Output: (estimate) state-revealing matrix $R^{(o)}$

1. $R^{(o)} = \bar{0} \in \mathbb{R}^{f \times N-p}$

2. for $k = p+1$ *to* N *do*

3. $R(1, k-p) =< \mathbf{T}, \mathbf{Z}_k >$ *using* (4.31)

4. for row number $t = 2$ *to* f *do*

5. for fixed cores number $\delta = t-1$ *to* $p-1$ *do*

6. $S = \mathcal{S}(\delta, t) \in \mathbb{R}^{\bar{c} \times \delta}$ *using Algorithm 4.C.1*

7. for $c = 1$ *to* $\bar{c}$

8. $\bar{L} = \mathcal{A}^{(1)}(:, 1 + S(c,1), :)$

9. for $s = 2$ *to* δ

10. $\bar{L} = \bar{L}\mathcal{A}^{(s)}(:, 1 + S(c,s), :)$

11. end for

12. $\check{\mathbf{T}} = g(\mathcal{A}^{(1+\delta)}, \ldots, \mathcal{A}^{(p)})$ *using* (4.22)

13. Compute $\check{\mathcal{B}}_k^{(1)}$ *to* $\check{\mathcal{B}}_k^{(p-\delta)}$ *using* (4.30)
with p *substituted by* $p-\delta$

14. $\check{\mathcal{B}}_k^{(p-fixno)} = D_r \check{\mathcal{B}}_k^{(p-\delta)}$ *using* (4.65)

15. $\check{\mathbf{Z}}_k = g(\check{\mathcal{B}}_k^{(1)}, \ldots, \check{\mathcal{B}}_k^{(p-\delta)})$ *using* (4.22)

16. $R^{(o)}(t, k-p) = R^{(o)}(t, k-p) +$
$\bar{L} < \check{\mathbf{T}}, \check{\mathbf{Z}}_k > / \bar{c}$

17. end for

18. end for

19. end for

20. end for

BIBLIOGRAPHY

Balas, G. J. Linear, parameter-varying control and its application to aerospace systems. In *ICAS Congress Proceedings*, 2002.

Batselier, K., Chen, Z., and Wong, N. Tensor network alternating linear scheme for MIMO volterra system identification. *Automatica*, 84:26–35, 2017.

Bianchi, F. D., Mantz, R. J., and Christiansen, C. F. Gain scheduling control of variable-speed wind energy conversion systems using quasi-LPV models. *Control Engineering Practice*, 13(2):247–255, 2005.

Brewer, J. Kronecker products and matrix calculus in system theory. *IEEE Transactions on circuits and systems*, 25(9):772–781, 1978.

Chen, Z., Batselier, K., Suykens, J. A., and Wong, N. Parallelized tensor train learning for polynomial pattern classification. *arXiv preprint arXiv:1612.06505*, 2016.

Chiuso, A. The role of vector autoregressive modeling in predictor-based subspace identification. *Automatica*, 43(6):1034–1048, 2007.

Cox, P. and Tóth, R. Alternative form of predictor based identification of LPV-SS models with innovation noise. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 1223–1228. IEEE, 2016.

De Caigny, J., Camino, J. F., and Swevers, J. Interpolating model identification for SISO linear parameter-varying systems. *Mechanical Systems and Signal Processing*, 23(8): 2395–2417, 2009.

De Silva, V. and Lim, L.-H. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1084–1127, 2008.

Felici, F., Van Wingerden, J.-W., and Verhaegen, M. Subspace identification of MIMO LPV systems using a periodic scheduling sequence. *Automatica*, 43(10):1684–1697, 2007.

Gebraad, P. M., van Wingerden, J.-W., Fleming, P. A., and Wright, A. D. LPV subspace identification of the edgewise vibrational dynamics of a wind turbine rotor. In *Control Applications (CCA), 2011 IEEE International Conference on*, pages 37–42. IEEE, 2011.

Gebraad, P. M., van Wingerden, J.-W., Fleming, P. A., and Wright, A. D. LPV identification of wind turbine rotor vibrational dynamics using periodic disturbance basis functions. *Control Systems Technology, IEEE Transactions on*, 21(4):1183–1190, 2013.

Giarré, L., Bauso, D., Falugi, P., and Bamieh, B. LPV model identification for gain scheduling control: An application to rotating stall and surge control problem. *Control Engineering Practice*, 14(4):351–361, 2006.

Golub, G. H., Heath, M., and Wahba, G. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.

Gunes, B., van Wingerden, J.-W., and Verhaegen, M. Predictor-based tensor regression (PBTR) for LPV subspace identification. *Automatica*, 79:235 – 243, 2017a. ISSN 0005-1098.

Gunes, B., van Wingerden, J.-W., and Verhaegen, M. Tensor networks for MIMO LPV system identification. *Submitted to International Journal of Control*, 2017b.

Gunes, B., van Wingerden, J.-W., and Verhaegen, M. Tensor nuclear norm LPV subspace identification. *IEEE Transactions on Automatic Control*, 2018.

Håstad, J. Tensor rank is NP-complete. *Journal of Algorithms*, 11(4):644–654, 1990.

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

Knudsen, T. Consistency analysis of subspace identification methods based on a linear regression approach. *Automatica*, 37(1):81–89, 2001.

Larimore, W. E. and Buchholz, M. ADAPT-LPV software for identification of nonlinear parameter-varying systems. *IFAC Proceedings Volumes*, 45(16):1820–1825, 2012.

Laurain, V., Gilson, M., Tóth, R., and Garnier, H. Refined instrumental variable methods for identification of LPV box-jenkins models. *Automatica*, 46(6):959–967, 2010.

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

Remmlinger, J., Buchholz, M., and Dietmayer, K. Identification of a bilinear and parameter-varying model for lithium-ion batteries by subspace methods. In *American Control Conference (ACC), 2013*, pages 2268–2273. IEEE, 2013.

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

Scherer, C. W. LPV control and full block multipliers. *Automatica*, 37(3):361–375, 2001.

Shamma, J. S. An overview of LPV systems. In *Control of linear parameter varying systems with applications*, pages 3–26. Springer, 2012.

Tóth, R. *Modeling and identification of linear parameter-varying systems*, volume 403. Springer, 2010.

Tóth, R., Abbas, H. S., and Werner, H. On the state-space realization of LPV input-output models: Practical approaches. *Control Systems Technology, IEEE Transactions on*, 20 (1):139–153, 2012.

Turkington, D. A. *Generalized vectorization, cross-products, and matrix calculus*. Cambridge University Press, 2013.

van der Maas, R., van der Maas, A., and Oomen, T. Accurate frequency response function identification of LPV systems: A 2D local parametric modeling approach. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pages 1465–1470. IEEE, 2015.

van der Veen, G., van Wingerden, J.-W., Bergamasco, M., Lovera, M., and Verhaegen, M. Closed-loop subspace identification methods: an overview. *IET Control Theory & Applications*, 7(10):1339–1358, July 2013. ISSN 1751-8652.

van Wingerden, J.-W. and Verhaegen, M. Subspace identification of bilinear and LPV systems for open- and closed-loop data. *Automatica*, 45(2):372 – 381, 2009. ISSN 0005-1098.

Verdult, V., Bergboer, N., and Verhaegen, M. Identification of fully parameterized linear and nonlinear state-space systems by projected gradient search. In *Proceedings of the 13th IFAC Symposium on System Identification, Rotterdam*, 2003.

Wassink, M. G., van de Wal, M., Scherer, C., and Bosgra, O. LPV control for a wafer stage: beyond the theoretical solution. *Control Engineering Practice*, 13(2):231–245, 2005.

White, S. R. Density matrix formulation for quantum renormalization groups. *Physical review letters*, 69(19):2863, 1992.

5

CONCLUSIONS AND RECOMMENDATIONS

In this final chapter, the conclusions about the proposed methods for LPV state-space identification using tensor techniques are presented and the research question is reflected upon. Additionally, some recommendations for future research are discussed.

5.1. CONCLUSIONS

In this thesis it is shown that the LPV system identification problem has an underlying *tensor structure* and how it can be exploited. This can result in more accurate LPV models which in turn allow for higher performance of LPV controllers and systems. This affects many applications, such as wind turbines, overhead cranes and bio-mechanics.

In this thesis three novel methods are proposed which exploit the tensor structure of the LPV identification problem using tailored tensor techniques. These methods are one subspace method and two refinement methods. The subspace method exploits tensor structure using tensor nuclear norms, whereas existing subspace methods do not. The refinement methods follow a vastly different approach than any existing refinement method. They do not circumvent the 'curse-of-dimensionality' through direct parametrization of the state-space matrices, but rather tackle it using tensor techniques. Hence the cost functions of the initializing method and the refinement method are in line, which may benefit the quality of the initialization of the refinement method. The novelty of the proposed methods lies mainly in how the problematic first regression step of predictor-based approaches is tackled. To summarize, the proposed methods are focussing on exploiting tensor structure to be 'curse-of-dimensionality'-free (in memory and computation) and to have improved variance [1].

Before discussing the detailed conclusions, the research question will be repeated. *"Do exact, low-rank polyadic, multi-linear singular value, tensor network decompositions of the LPV sub-Markov parameters exist, and if so, what are they and how can they be exploited to obtain methods which are 'curse-of-dimensionality'-free in memory and*

[1] More specifically, in this thesis improved variance is defined as higher variance accounted for than the LPV-PBSID$_{\text{opt}}$ method with regularization and kernels which is taken as the 'base-line' method.

computation and have improved variance?" This question can be split into two more manageable sub-questions. Firstly, do such exact, low-rank tensor decompositions of the LPV sub-Markov parameters exist and if so, what are they? Secondly, how can these decompositions be exploited to obtain methods with these properties?

These two sub-questions can be explained as follows. The challenge of this thesis is dealing with the 'curse-of-dimensionality' of LPV predictor-based methods. It has been proposed to use tensor techniques, which can break 'curse-of-dimensionality'. Their effectiveness however depends on whether (preferably exact) low-rank tensor decompositions are available of the problematic variables. This explains the first sub-question. Once such tensor decompositions are available, their properties can be exploited. The thesis goal is to, by exploiting this tensor structure, develop LPV identification techniques which are 'curse-of-dimensionality'-free in memory and computation and have improved variance. This explains the second sub-question. The research question will be discussed using these two sub-questions.

FIRST SUB-QUESTION

The first sub-question is on whether exact, low-rank tensor decompositions of the LPV sub-Markov parameters exist and what they are. For the polyadic approach, an exact but not low-rank polyadic decomposition has been presented. It has to be remarked that there might exist other decompositions which do have the desired low-rank property. For the Multi-Linear Singular Value Decomposition (MLSVD) and tensor network decomposition, exact and low-rank decompositions of the LPV sub-Markov parameters have been proven to exist and have been presented algebraically.

These exact, low-rank decompositions are obtained through organizing the LPV sub-Markov parameters into block-Hankel tensors. The rank of these block-Hankel tensors gives information on the underlying system order of the LPV state-space system. Namely, its MLSVD has ranks equal to the system order and the square of the system order. Also, the tensor network decomposition has ranks exactly equal to the system order.

SECOND SUB-QUESTION

The second sub-question is on how to exploit the presented tensor decompositions to obtain methods which are 'curse-of-dimensionality'-free in memory and computation and have improved variance. In this thesis, both subspace (initializing) and refinement methods have been developed.

Firstly, a convex subspace method has been proposed which exploits the tensor structure using a convex regularization term. This regularization term contains (tensor) nuclear norms and utilizes the fact that the proposed block-Hankel tensor is (multi-linear) low-rank. Additionally, the memory and computation aspects have been tackled using tailored data kernels. Simulation results show the proposed method has a higher VAF than the base-line method for the presented cases.

Secondly, two refinements methods have been proposed. For the polyadic method, the used polyadic decomposition is not low-rank, and the method is not 'curse-of-dimensionality'-free. However, the proposed tensor network method is 'curse-of-dimensionality'-free in memory and computation. This method recasts the LPV identification problem in tensor networks and performs the operations directly on these

networks. In other words, the LPV sub-Markov parameters are never constructed explicitly in memory. An additional, possible benefit is that these refinement methods have cost functions which are very similar to those of their initializing (convex predictor-based) methods. This property may yield better initializations of these refinement methods. Both refinement methods have been shown to successfully refine (or increase mean validation VAF of) initial estimates using simulations.

The conclusions are summarized in Table 5.1.

Method	decomposition	exact	low-rank	class	'c.o.d.'-free	VAF
'base-line'	none	N/A	N/A	initializer	+	N/A
Chapter 2	polyadic	+	-	refiner	-	++
Chapter 3	MLSVD	+	+	initializer	+	+
Chapter 4	tensor network	+	++	refiner	+	++

Table 5.1: This table summarizes the discussed properties of the proposed methods. Methods are graded with '-', '+' and '++' for respectively bad, good and very good properties in a category. Notice that refinement methods do require initialization and involve non-convex optimization. Also, the word 'c.o.d-free' is abbreviated from: 'curse-of-dimensionality'-free in memory and computation, and is a desired property.

5.2. RECOMMENDATIONS

In this thesis the focus has been on the development of identification methods which exploit tensor structure, and not the details of choosing past and future windows, input and scheduling signals or regularization parameters. It would be of practical interest to investigate how to choose these parameters. For example, for the proposed polyadic method the choice of the past window is especially interesting, because the method is not 'curse-of-dimensionality'-free in memory and computation. This means that for a given identification problem the optimal past window for this method can be beyond the memory capabilities of current computers. It would be of interest to have techniques to check whether this is the case for a given identification problem. Another example is that for the proposed tensor network method there are conditioning guarantees available which depend on the data. This means it could be beneficial to design inputs and if possible scheduling signals with these guarantees in mind.

For the proposed refinement methods, standard Alternating Least Squares (ALS) schemes were used. However, it would be of interest to apply more advanced variants. The first example is about polyadic decompositions. This decomposition represents a tensor as a sum of (outer) products of vectors. It is well-known in literature that if some of these vectors are (almost) collinear, then a standard ALS scheme optimization of this decomposition may converge slowly. It should be investigated whether this collinear property does appear for the proposed polyadic method, and whether it imposes a significant problem. If so, then advanced ALS variants should be investigated and applied.

The second example is about the proposed tensor network method. For this method also a standard ALS (tensor network) scheme was used. This required fixing the tensor network ranks a priori. These are now fixed at the model order of the initial estimate, which may not be optimal for the tensor network. However, the Density Matrix Renormalization Group optimization approach (DMRG) does not have this requirement. That is, by using DMRG the ranks can be allowed to vary during optimization of the network. Hence this DMRG approach may allow obtaining smaller rank tensor network estimates with comparable performance. This could benefit computation and reduce over-fitting. Additional questions in this DMRG setting would be how to deal with the a priori knowledge that the ranks are all equal.

For the proposed subspace method, it would be of interest to assess the obtainable computational speed when the accelerated variant of the Alternating Direction Method of Multipliers (ADMM) and a dedicated algorithm, such as in Python or C, are used. Also, the ADMM could be initialized using an initial estimate. This can be done using any of the other LPV predictor-based subspace kernel methods. The possible benefit is that the ADMM routine may start closer its optimum and thus converge in less time. This also involves making more use of existing optimization tools in literature.

The proposed block-Hankel tensors are also relevant for LTI problems and their use for LTI identification methods should be investigated. These tensors can be simplified from the LPV case to the LTI case to obtain new tensors, which can still be low-rank and admit exact, low-rank tensor decompositions. This property is due to the fact that the Markov parameters are products of matrices, which holds both in the LTI and the LPV setting.

ACKNOWLEDGEMENTS

I would like to thank Jan-Willem van Wingerden, Michel Verhaegen and the Design for Reliability project for giving me the opportunity to perform research at Delft Center for Systems and Control (DCSC), and for their support and feedback during the project. I would like to separately thank Jan-Willem again for the freedom he gave me during the project and for his understanding and support.

I would also like to thank several (former) colleagues from DCSC. I would like to thank Arne and Paul van den Hof for their supervision and priceless feedback leading to the start of my PhD. I appreciate the support of my office room mate Edwin, who spent considerable effort into helping me with several small and big things. The support of the other DCSC wind group PhD's is also appreciated: Sachin, Pieter, Sjoerd, Hildo, Patricio and later also Bart, Joeri, Sebastiaan and Nikolaos. Reinier has supported me a lot regarding theoretic results, and was always open to discuss tensor algebra. Separately, I would like to thank Renshi and Chengpu for their good pointers. These pointers helped me enormously, several years later. I enjoyed playing fooseball and having lunch at work, including the anecdotes and propositions of Laurens, Kim, Hans, Paulo, Dean, Elisabeth and my other colleagues.

I would like to thank Philippe, Kim and Mariya from VUB and Hong Kong for the insights and introducing me to the tensor field community. The support of Stoyan Kanev from Energie Centrum Nederland is also appreciated.

I would also like to thank my parents and brothers for their support.

Finally, I thank my wife for her love and support. Together with my daughter they have been my main source of motivation and happiness.

SUMMARY

Many industrial applications, such as wind turbines and overhead cranes, would benefit in terms of performance by using Linear Parameter Varying (LPV) robust controllers instead of Linear Time Invariant (LTI) ones. This is because the underlying LPV models can describe time-varying dynamics (through external time-varying parameters) for higher modelling accuracy. The performance of these LPV controllers depend on the quality of the LPV models. While there are powerful LPV control design methodologies available, obtaining LPV models is still in its infancy. LPV models can be obtained from experimental data under normal operation by using (predictor-based) system identification, but a 'curse-of-dimensionality' appears. Namely, the to be estimated (LPV sub-Markov) parameters are generally to numerous to store in memory or compute with straight-forward techniques. Techniques which do not have this problem are 'curse-of-dimensionality'-free in memory and computation. Additionally, the number of parameters can exceed the number of data samples causing estimates to have high variance. In literature, tensor techniques are used to break 'curse-of-dimensionality' by exploiting underlying tensor structure. In this thesis it is shown that the LPV system identification problem does have an underlying *tensor structure*. The goal of this thesis is to develop LPV system identification techniques which are 'curse-of-dimensionality'-free in memory and computation and have improved variance [2] by exploiting the tensor structure.

In order to exploit the tensor structure, suitable tensor decompositions of the problematic variables are needed. Tensor decompositions are (preferably exact) condensed representations of (variables re-organized into) tensors. This reduces memory usage and can simplify computations. How condense a decomposition is depends on the rank of the decomposition. In this thesis a tensor decomposition is said to be low-rank, if its rank is less than what is maximally needed to describe every tensor of the same size. In this thesis, the three most well-known and well-understood tensor decomposition approaches are considered: polyadic, Multi-Linear Singular Value Decomposition (MLSVD) and tensor networks. For the polyadic approach an exact, but not low-rank, polyadic decomposition has been presented. It has to be remarked that there might exist other decompositions which do have the desired low-rank property. For the MLSVD and tensor network decomposition, exact and low-rank decompositions of the LPV sub-Markov parameters have been proven to exist and presented algebraically.

These suitable decompositions hinge on the forming of block-Hankel tensors. These are generalizations of the Hankel matrix and share useful properties. The Hankel matrix has constant skew diagonals and its rank gives information on the system order. As a powerful result, the proposed tensor network decomposition has ranks exactly equal to the system order.

[2] More specifically, in this thesis improved variance is defined as higher variance accounted for. The LPV-PBSID$_{\text{opt}}$ method with regularization and kernels is taken as the 'base-line' method to compare against.

These proposed tensor decompositions can now be exploited to obtain methods which are 'curse-of-dimensionality'-free in memory and computation and have improved variance. In this thesis both convex and non-convex methods have been developed. Non-convex (refinement) methods can be used to refine estimates obtained from convex methods. The proposed convex method exploits tensor structure through novel tensor nuclear norm regularization. This regularization is a way to introduce a bias-variance trade-off in estimates, where some small bias is introduced to reduce large variances. These proposed tensor nuclear norms exploit the property that the MLSVD of the LPV sub-Markov parameters is exact and low-rank. Simulation results show the proposed method has a improved variance for the presented cases. Also, two non-convex (refinement) methods have been proposed. Firstly, the polyadic method appeared to not be 'curse-of-dimensionality'-free, because the used decomposition is not low-rank. Secondly, the tensor network method is 'curse-of-dimensionality'-free in memory and computation. Namely, it recasts the LPV identification problem in tensor networks and performs the operations directly on these networks. In other words, the problematic LPV sub-Markov parameters are never constructed explicitly in memory. An additional, possible benefit is that these refinement methods optimize (cost) functions which are very similar to those of their initializing methods. Both refinement methods have been shown to successfully refine initial estimates using simulations. Hence, the three proposed methods can have improved variance.

SAMENVATTING

De prestatie van veel industriële applicaties, zoals windturbines en bovenloopkranen, hebben baat bij het gebruik van robuuste regelaars voor Lineair Parameter Variërende (LPV) systemen in plaats van Lineair Tijd Invariante (LTI) systemen. De onderliggende LPV-modellen kunnen tijd-variërende dynamica beschrijven (via externe tijd-variërende parameters) en resulteren in accuratere modellen ten opzichte van LTI-modellen. De accuratere modellen kunnen gebruikt worden voor het ontwerp van betere regelaars. De prestatie van deze LPV regelaars hangt af van de kwaliteit van de LPV-modellen. Hoewel effectieve LPV regelaar-ontwerptechnieken beschikbaar zijn, staat het verkrijgen van LPV modellen nog in de kinderschoenen. LPV modellen kunnen verkregen worden aan de hand van experimentele data, gedurende de normale werking van het systeem, met behulp van (voorspeller-gebaseerde) systeemidentificatie. Echter, een 'vloek-van-dimensionaliteit' verschijnt: er zijn te grote aantallen (LPV sub-Markov) parameters die geschat moeten worden. Het is een probleem om deze parameters in computergeheugen op te slaan of om mee te rekenen met standaard technieken. Speciale technieken die dit probleem niet hebben zullen 'vloek-van-dimensionaliteit'-vrij in geheugen en berekening worden genoemd. Bovendien kan het aantal parameters groter zijn dan het aantal data punten, waardoor schattingen hoge variantie krijgen.

In de literatuur worden tensortechnieken gebruikt om 'vloek-van-dimensionaliteit' te doorbreken door onderliggende tensorstructuren uit te buiten. In dit proefschrift wordt aangetoond dat het LPV systeemidentificatieprobleem een onderliggend tensorstructuur heeft. Het doel van dit proefschrift is het ontwikkelen van LPV systeemidentificatietechnieken die 'vloek-van-dimensionaliteit'-vrij in geheugen en berekening zijn en verbeterde variantie [3] hebben door middel van het uitbuiten van deze tensorstructuur.

Om de tensorstructuur uit te kunnen buiten zijn geschikte tensorontbindingen van de problematische variabelen nodig. Tensorontbindingen zijn (bij voorkeur exacte) efficiënte beschrijvingen van variabelen, georganiseerd in tensoren. Het gebruik van tensorontbindingen vermindert geheugengebruik en mogelijk ook de rekenlast. Hoe efficiënt een tensorontbinding is, hangt af van de rang van de tensorontbinding. In dit proefschrift wordt een tensorontbinding 'lage rang' genoemd, wanneer zijn rang kleiner is dan wat maximaal nodig is om elke tensor van dezelfde grootte te beschrijven. In dit proefschrift worden alleen de drie meest bekende en goed begrepen tensorontbindingen geëvalueerd: polyadische ontbinding, Multilineaire Singulierewaardenontbinding (MS) en ontbinding in tensornetwerken. Voor de polyadische aanpak wordt een exacte, maar niet lage-rang, polyadische tensorontbinding gepresenteerd. Een opmerking is dat er wellicht andere tensorontbindingen zijn die wel de beoogde lage-rang eigenschap heb-

[3] Specifiek is 'verbeterd variantie' in dit proefschrift gedefinieerd als een hogere Verklaarde Variantie waarde. Hierbij wordt vergeleken met een vergelijkbaar, bestaand methode in de literatuur (Van Wingerden en Verhaegen, 2009).

ben. Voor de MS- en de tensornetwerk-aanpakken zijn exacte, lage-rang tensorontbindingen van de LPV sub-Markov parameters bewezen te bestaan en zijn in algebraïsche vorm gepresenteerd.

Deze geschikte tensorontbindingen zijn gebaseerd op het vormen van blok-Hankel tensoren. Deze zijn generalisaties van de Hankelmatrix en hebben vergelijkbare bruikbare eigenschappen. De Hankelmatrix heeft constante antidiagonalen en zijn rang geeft informatie over de orde van de dynamica in het onderliggende systeem. Een sterk resultaat is dat de rangen van de voorgelegde tensornetwerkontbinding exact gelijk zijn aan de systeemorde.

De voorgelegde tensorontbindingen kunnen nu uitgebuit worden om methodes te verkrijgen die 'vloek-van-dimensionaliteit'-vrij zijn in geheugen en berekening en verbeterde variantie hebben. In dit proefschrift zijn zowel convexe als niet-convexe methodes ontwikkeld. Niet-convexe (verfijnings)methoden kunnen gebruikt worden om schattingen van convexe methodes te verfijnen. De voorgelegde convexe methode buit tensorstructuur uit met behulp van tensor-nucleaire-norm regularisatie. Deze voorgelegde tensor nucleaire normen buiten de eigenschap uit dat de MS van de LPV sub-Markov parameters exact en lage-rang zijn. Simulatieresultaten laten zien dat de voorgelegde methode een verbeterde variantie heeft voor de gepresenteerde voorbeelden. Er zijn ook twee niet-convexe (verfijnings)methoden voorgelegd. Ten eerste, de polyadische methode blijkt niet 'vloek-van-dimensionaliteit'-vrij in geheugen en berekening zijn doordat de gebruikte tensorontbinding niet lage-rang is. Ten tweede, de tensornetwerk-methode is daarentegen 'vloek-van-dimensionaliteit'-vrij in geheugen en berekening. Namelijk, het herschrijft het LPV systeemidentificatieprobleem met tensornetwerken en voert de berekeningen direct op deze netwerken uit. In andere woorden, de problematische LPV sub-Markov parameters worden nooit expliciet opgebouwd in het geheugen. Een aanvullend, mogelijk voordeel is dat deze twee verfijningsmethoden kostenfuncties optimaliseren die erg vergelijkbaar zijn met die van methodes die zijn gebruikt om hun initiële schattingen te verkrijgen. Voor beide verfijningsmethoden wordt laten zien dat deze initiële schattingen succesvol kunnen verfijnen. Dit laat zien dat de drie voorgelegde methodes verbeterde variantie kunnen hebben.

BIBLIOGRAFIE

van Wingerden, J.-W. and Verhaegen, M. Subspace identification of bilinear and LPV systems for open- and closed-loop data. *Automatica*, 45(2):372 – 381, 2009. ISSN 0005-1098.

LIST OF PUBLICATIONS

Journal papers

3. Bilal Gunes, Jan-Willem van Wingerden, and Michel Verhaegen. Tensor networks for MIMO LPV system identification. *Submitted to the International Journal of Control, 2017*.

2. Bilal Gunes, Jan-Willem van Wingerden, and Michel Verhaegen. Tensor Nuclear Norm LPV Subspace Identification. *Transactions on Automatic Control, 2018*.

1. Bilal Gunes, Jan-Willem van Wingerden, and Michel Verhaegen. Predictor-based tensor regression (PBTR) for LPV subspace identification. *Automatica, 79:235-243, 2017*.

Conference papers

3. Bilal Gunes, Jan-Willem van Wingerden, and Michel Verhaegen. Tensor regression for LPV subspace identification. SYmposium on System IDentification (SYSID), 2015. IFAC-PapersOnLine 48, no. 28 (2015): 421-426.

2. Bilal Gunes, Jan-Willem van Wingerden, and Michel Verhaegen. Tensor regression for LTI subspace identification: free parametrizations. SYmposium on System IDentification (SYSID), 2015. IFAC-PapersOnLine 48, no. 28 (2015): 909-914.

1. Bilal Gunes, Jan-Willem van Wingerden, and Michel Verhaegen. Tensor regression for LTI subspace identification. American Control Conference (ACC), 1131-1136, 2015.

Curriculum Vitæ

Bilal Gunes was born on March 20th, 1990 in Amsterdam, The Netherlands.

He followed secondary education from 2002 to 2008 at the Comenius Lyceum in Amsterdam. In 2008 he started his studies in Mechanical Engineering at the Delft University of Technology and obtained the Bachelor degree in 2011. In September 2013 he received his Master of Science degree in Control Engineering from the Delft University of Technology. His graduation project was titled “A novel network prediction error identification method”.

Between October and November 2013 he was a researcher at the Delft University of Technology.

In January 2014, he started his Ph.D. project the Delft Center for Systems and Control, titled "A tensor approach to linear parameter varying system identification". This project was funded by TKI Wind Op Zee under the Design For Reliability (D4REL) project. During his Ph.D. research he assisted in several courses and participated in various international conferences and events.