ARTICLE TEMPLATE

A recursive LMI-based algorithm for efficient vertex reduction in LPV systems

Adrián Sanjuan[a], Damiano Rotondo[b], Fatiha Nejjari[a], and Ramon Sarrate[a]

[a] Research Center for Supervision, Safety and Automatic Control (CS2AC), Universitat Politècnica de Catalunya (UPC). Address: Rambla Sant Nebridi 10, 08222 Terrassa, Spain. Edifici TR11;
[b] Department of Electrical Engineering and Computer Science (IDE), University of Stavanger, Kristine Bonnevies vei 22, 4021, Stavanger, Norway

ARTICLE HISTORY
Compiled August 10, 2021

ABSTRACT
This paper proposes a new algorithm to reduce the number of gains of a polytopic LPV controller considering generic tuples of vertices, for which a common controller gain can be used. The use of Frobenius norm and the inclusion of the input matrix in the LMIs perturbation matrix allows decreasing the conservativeness to select vertices which are combinable, with respect to existing conditions based on Gershgorin circles (Sanjuan, Rotondo, Nejjari, & Sarrate, 2019). The proposed algorithm is developed by defining a combinability metric that can be applied to an arbitrary partition of the set of vertices. Then, a recursive algorithm finds a lesser-fragmented combinable partition at each iteration by combining together two elements of a partition. The algorithm aims at finding combinable partitions with minimal cardinality in fewer attempts, with the objective of reducing the number of gains of a polytopic controller, always preserving the original performance specifications. The proposed method is validated using numerical examples, a twin rotor MIMO system (TRMS) and a two-link robotic manipulator.

KEYWORDS
Linear parameter varying (LPV); linear matrix inequalities (LMIs); vertex reduction; state-feedback control.

1. Introduction

Linear parameter varying (LPV) systems are among the most successful approaches to control nonlinear systems. By means of a suitable embedding of the nonlinearities within some *scheduling variables* (Kwiatkowski, Boll, & Werner, 2006; Rotondo, Puig, Nejjari, & Witczak, 2015), many nonlinear systems can be transformed in a so-called equivalent *quasi-LPV representation*. The term *quasi* refers to the dependence of the varying parameters on endogenous signals, such as states and/or inputs (Marcos &

Email addresses: adrian.sanjuan@upc.edu (ADRIÁN SANJUAN), damiano.rotondo@uis.no (DAMIANO ROTONDO), fatiha.nejjari@upc.edu (FATIHA NEJJARI), ramon.sarrate@upc.edu (RAMON SARRATE)

Balas, 2004). Then, linear-like analysis and design techniques can be applied straightforwardly to nonlinear systems, with the main advantage of retaining their simplicity (Apkarian, Gahinet, & Becker, 1995; Blanchini & Miani, 2003; Bokor & Balas, 2004; Pfifer & Seiler, 2015; Rotondo, Nejjari, & Puig, 2014). In the last years, a wide range of fields have benefited from the application of LPV control, such as wind power production (Ibáñez, Inthamoussou, & De Battista, 2019), aviation (López-Estrada, Ponsart, Theilliol, Zhang, & Astorga-Zaragoza, 2016; Rotondo, Cristofaro, Johansen, Nejjari, & Puig, 2019), or robotics (Rotondo, Puig, Nejjari, & Romera, 2015), just to name a few (the interested reader is referred to the survey paper (Hoffmann & Werner, 2015)).

When using LPV models, analysis and control synthesis tasks usually take the form of a set of parameter-dependent linear matrix inequalities (LMIs), which are not tractable computationally because they correspond to infinite-dimensional constraints. In order to reduce them to a finite number of conditions, some kind of manipulation should be performed, e.g. the parameter-varying matrices (among which one finds the LPV controller gain) can be described as a convex combination of constant *vertex* matrices, which is referred to in the literature as the *polytopic approach* (see the tutorial (Rotondo, Sánchez, Nejjari, & Puig, 2019) for further information). However, in polytopic approaches, the complexity of the controller depends *exponentially* on the number of scheduling variables, which means that a large number of these variables would lead to a much higher number of vertex controller gains that should be synthesized (Rizvi, Abbasi, & Velni, 2018). For instance, for the most popular polytopic modelling technique, which is the *bounding box* (Sun & Postlethwaite, 1998), the number of vertices n_v and the number of scheduling variables n_p are related as follows

$$n_v = 2^{n_p}. \tag{1}$$

This issue, which may cause the implementation of an LPV controller into a micro-controller to become computationally infeasible, does not affect only large-scale systems, but also small-scale laboratory setups. For instance, in (Rotondo, Nejjari, & Puig, 2013) LPV techniques were applied to a twin rotor multiple-input multiple-output system (TRMS), showing that the highly nonlinear coupled model could be described by an LPV model with 11 scheduling parameters, which correspond to 2048 vertex controller gains.

There is a twofold drawback which is inherent in a high vertex number. First of all, it increases the amount of memory that must be allocated in the micro-controller to store the controller gains (and, possibly, the observer gains, if an estimate-feedback control law is considered due to the inaccessibility of some states for measurement). Secondly, a high vertex number leads to a longer time required to compute the instantaneous gain as a weighted combination of vertex gains. This issue may lead to undesired delays, thus hindering the real-time implementation, especially in cases where the main dynamics of the plant to be controlled are particularly fast.

For this reason, a line of research that has received some recent interest is the reduction of the LPV controller complexity, which can be achieved through five different approaches:

(i) Principal component analysis (PCA), by detecting and neglecting the less significant directions in the parameter space without losing much information regarding the plant, thus allowing the attainment of a tighter parameter set (Jabali

& Kazemi, 2017; Kwiatkowski & Werner, 2008; Rizvi, Mohammadpour, Tóth, & Meskin, 2016). Notably, some recent advances in this domain have led to the application of autoencoder neural networks, which achieve reduction of the varying parameters without being restricted to behave as linear maps (Rizvi et al., 2018).

(ii) Gap metric (GM) techniques, by quantifying the distance between two models (El-Sakkary, 1985), and providing an indication of how much performance/robustness is lost when a controller designed for one local model is used for another one (Ahmadi & Haeri, 2018; Vizer & Mercere, 2014; Zribi, Chtourou, & Djemal, 2016).

(iii) Model order reduction, by reducing the dimension of the state vector while keeping a similar input-output behaviour to the non-reduced system (Abbas & Werner, 2010; Gőzse, Luspay, Péni, Szabó, & Vanek, 2016; Matz, Mourllion, & Birouche, 2018; Poussot-Vassal & Roos, 2012).

(iv) Polytope size reduction, by discarding infeasible combinations of time-varying parameters based on their interdependency (Robert, Sename, & Simon, 2009).

(v) LMI-based reduction, by designing a common controller gain for a set of vertices based on analysing how much the LMIs, which represent a set of specifications, are modified by this common controller (Sanjuan et al., 2019).

This last category of approaches is still premature, since the recent (and sole available) work (Sanjuan et al., 2019) only set the basis for its development towards a more mature stage. More specifically, (Sanjuan et al., 2019) provided a heuristic methodology to combine *two* vertices of an LPV polytopic model. Hence, the number of vertex controllers could be reduced from n_v to n_v-1, which was deemed to be a first necessary step to develop an iterative algorithm for achieving a bigger reduction. The heuristic algorithm proposed in (Sanjuan et al., 2019) was based on the use of Gershgorin circles to analyse the perturbation of LMIs when two vertices are combined together, in the sense of a common controller gain being used for both.

The main goal of the present work is to develop an algorithm that allows considering generic tuples of vertices, for which a common controller gain is used, in order to achieve an overall reduction from n_v vertices to a much smaller number n_c. This is attained by starting with the algorithm in (Sanjuan et al., 2019), but with two important modifications, that are the main contributions of this paper:

(i) the basic algorithm to combine two vertices is improved by introducing two innovations. First of all, it is shown that the use of the Frobenius norm (Quarteroni, Sacco, & Saleri, 2010), instead of Gershgorin circles, leads to less conservativeness. Secondly, the role of the input matrix in the LMIs, originally neglected by (Sanjuan et al., 2019), is taken into account by means of an appropriate redefinition of the perturbation matrix, thus achieving a further decrease of conservativeness;

(ii) a complete algorithm that achieves a reduction from n_v to n_c vertices is developed by extending the combinability metric such that it can be applied to an arbitrary partition of the set of vertices; then, a recursive algorithm finds a lesser-fragmented combinable partition at each iteration by combining together two elements of a partition.

By applying the developed algorithms, a more compact controller that preserves the

original performance specifications will be obtained. This reduction will certainly lead to an increase of the availability of the processor, so that the resulting controller becomes implementable in micro-controllers or small-scale laboratory setups with limited computational power or available resources.

Throughout the paper, several examples are used to show the application of the algorithm and validate its effectiveness: numerical systems, a TRMS, and a two-link robotic manipulator.

The remaining of the paper is structured as follows. In Section 2, the problem under consideration is introduced and formulated, while a review of the combinability ranking design as well as some improvements to increase the likelihood of selecting a combinable vertex pair are provided in Section 3. Section 4 presents the complete algorithm to reduce the number of vertices of a polytopic system and some examples that show the efficiency of the algorithm are analysed in Section 5. Finally, Section 6 outlines the main conclusions.

2. Problem formulation

Let us consider the following LPV system

$$\sigma\mathbf{x}(t) = \mathbf{A}(\boldsymbol{\theta}(t))\mathbf{x}(t) + \mathbf{B}(\boldsymbol{\theta}(t))\mathbf{u}(t), \tag{2}$$

where $\sigma\mathbf{x}(t)$ denotes $\dot{\mathbf{x}}(t)$ in the continuous-time case and $\mathbf{x}(t+1)$ in the discrete-time case, $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ and $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ are the state and the input vector, respectively, and $\mathbf{A}(\boldsymbol{\theta}(t)) \in \mathbb{R}^{n_x \times n_x}$ and $\mathbf{B}(\boldsymbol{\theta}(t)) \in \mathbb{R}^{n_x \times n_u}$ are parameter-dependent matrices, which depend on the vector of time-varying parameters $\boldsymbol{\theta}(t) \in \Theta \subset \mathbb{R}^{n_p}$.

An equivalent representation of the LPV system (2) is obtained by considering that the matrices $\mathbf{A}$ and $\mathbf{B}$ are described as a convex combination of constant matrices which are functions of $\boldsymbol{\theta}(t)$, i.e., they depend on $\boldsymbol{\theta}(t)$ in a polytopic way

$$\sigma\mathbf{x}(t) = \sum_{v=1}^{n_v} \alpha_v(\boldsymbol{\theta}(t))(\mathbf{A}_v\mathbf{x}(t) + \mathbf{B}_v\mathbf{u}(t)), \tag{3}$$

where $\mathbf{A}_v$ and $\mathbf{B}_v$, $v = 1, \ldots, n_v$, define the so-called vertex systems (in the following, with a slight abuse of language, they will be referred to as *vertices*), n_v is the number of vertices and α_v are the coefficients of the polytopic decomposition, such that

$$\begin{cases} \sum\limits_{v=1}^{n_v} \alpha_v(\boldsymbol{\theta}(t)) = 1 \\ \alpha_v(\boldsymbol{\theta}(t)) \geq 0 \qquad\qquad \forall v = 1, ..., n_v,\ \forall \boldsymbol{\theta}(t) \in \Theta \end{cases}. \tag{4}$$

Hereafter, an LPV state-feedback controller will be designed, so that the system (3) achieves the desired closed-loop performance, as follows

$$\mathbf{u}(t) = \mathbf{K}(\boldsymbol{\theta}(t))\mathbf{x}(t), \tag{5}$$

where $\mathbf{K}(\boldsymbol{\theta}(t)) \in \mathbb{R}^{n_u \times n_x}$ is the controller gain, which depends on $\boldsymbol{\theta}(t)$. In recent years,

the description of the specifications through LMIs has become popular because they are a versatile tool which allows to handle different types of performance conditions, e.g. stability, pole placement or guaranteed $\mathcal{H}_\infty$ performance. The LMIs are a set of parameterized inequalities with unknown variables $\mathbf{X} \succ \mathbf{0}$ (Lyapunov matrix) and $\mathbf{\Gamma}(\boldsymbol{\theta}(t)) = \mathbf{K}(\boldsymbol{\theta}(t))\mathbf{X}$ (Apkarian & Tuan, 2000) such that

$$\mathbf{F}\left(\mathbf{A}(\boldsymbol{\theta}(t)), \mathbf{B}(\boldsymbol{\theta}(t)), \mathbf{X}, \mathbf{\Gamma}(\boldsymbol{\theta}(t))\right) \prec 0 \quad \forall \theta \in \Theta. \tag{6}$$

However, (6) is computationally intractable because it corresponds to an infinite number of LMIs. For this reason, a polytopic representation of the LPV controller is chosen to reduce the number of constraints involved in the design. Consequently, (5) becomes

$$\mathbf{u}(t) = \sum_{v=1}^{n_v} \alpha_v(\boldsymbol{\theta}(t)) \mathbf{K}_v \mathbf{x}(t), \tag{7}$$

where $\mathbf{K}_v \in \mathbb{R}^{n_u \times n_x}$ are the vertex controller gains. Rewriting (6) at the vertices, taking into account the previous assumption, as

$$\mathbf{F}\left(\mathbf{A}_v, \mathbf{B}_v, \mathbf{X}, \mathbf{\Gamma}_v\right) \prec 0 \quad \forall v = 1, \ldots, n_v, \tag{8}$$

and solving (8), the vertex controller gains can be obtained as follows

$$\mathbf{K}_v = \mathbf{\Gamma}_v \mathbf{X}^{-1}. \tag{9}$$

As stated in the introduction, we are interested in decreasing the number of controller gains from n_v to n_c, where $n_c < n_v$. This objective is achieved by considering the same controller gain for a *combinable set* of vertices, of which we give a formal definition hereafter.

Definition 2.1 (Combinable partition)**.** A partition $\mathcal{S}$ of the set of vertices $\mathcal{V} = \{1, ..., n_v\}$ is said to be *combinable* for the set of LMIs (8), if there exist a matrix $\mathbf{X} \succ 0$ and matrices $\Gamma_s \in \mathbb{R}^{n_u \times n_x}$, such that

$$\mathbf{F}\left(\mathbf{A}_v, \mathbf{B}_v, \mathbf{X}, \mathbf{\Gamma}_s\right) \prec 0 \quad \forall v \in s \in \mathcal{S}. \tag{10}$$

Definition 2.2 (Combinable set)**.** A combinable set s is an element of a combinable partition $\mathcal{S}$

$$s \in \mathcal{S}. \tag{11}$$

The LPV state-feedback controller (7) of a combinable partition $\mathcal{S}$ will be given by

$$\mathbf{u}(t) = \sum_{i=1}^{|\mathcal{S}|} \alpha_{s_i} \mathbf{K}_i \mathbf{x}(t), \tag{12}$$

with

$$\alpha_{s_i} = \sum_{v \in s_i} \alpha_v(\boldsymbol{\theta}(t)), \tag{13}$$

where $|\mathcal{S}|$ is the cardinality of the partition $\mathcal{S}$ and s_i is an element of $\mathcal{S}$.

The problem of interest in this paper is to find an algorithm that searches for combinable partitions with minimal (or at least close to the minimal) cardinality in fewer attempts, with the goal of reducing the number of controller gains.

The paper will focus on the design of a polytopic controller, even though the proposed methodology can be extended to any other problem related to gain-scheduled design, e.g. observer or fault estimator design.

3. Improvements in the combinability ranking generation

3.1. Background

The paper (Sanjuan et al., 2019) developed a heuristic methodology based on Gershgorin circles that searched for two combinable vertices in a polytopic LPV model. The main ideas of this work are summarised hereafter.

Let us consider a system with combinable vertices i and j, such that a common controller gain $\mathbf{K}_i = \mathbf{K}_j$ can be designed, which means that $\mathbf{\Gamma}_i = \mathbf{\Gamma}_j = \mathbf{\Gamma}^{\{i,j\}}$. For these vertices, (8) will be stated as

$$\begin{cases} \mathbf{F}\left(\mathbf{A}_i, \mathbf{B}_i, \mathbf{X}, \mathbf{\Gamma}^{\{i,j\}}\right) \prec 0 \\ \mathbf{F}\left(\mathbf{A}_j, \mathbf{B}_j, \mathbf{X}, \mathbf{\Gamma}^{\{i,j\}}\right) \prec 0 \end{cases} . \tag{14}$$

By considering that the vertex i can be perceived as a perturbation with respect to the vertex j and vice versa, the LMIs (14) can be rewritten as

$$\begin{cases} \mathbf{F}\left(\mathbf{A}_i, \mathbf{B}_i, \mathbf{X}, \mathbf{\Gamma}^{\{i,j\}}\right) \prec 0 \\ \mathbf{F}\left(\mathbf{A}_i, \mathbf{B}_i, \mathbf{X}, \mathbf{\Gamma}^{\{i,j\}}\right) + \Delta\mathbf{F}\left(*\right)^{\{i,j\}} \prec 0 \end{cases} \Leftrightarrow \begin{cases} \mathbf{F}\left(\mathbf{A}_j, \mathbf{B}_j, \mathbf{X}, \mathbf{\Gamma}^{\{i,j\}}\right) - \Delta\mathbf{F}\left(*\right)^{\{i,j\}} \prec 0 \\ \mathbf{F}\left(\mathbf{A}_j, \mathbf{B}_j, \mathbf{X}, \mathbf{\Gamma}^{\{i,j\}}\right) \prec 0 \end{cases} , \tag{15}$$

with

$$\Delta\mathbf{F}\left(*\right)^{\{i,j\}} = \mathbf{F}\left(\mathbf{A}_j, \mathbf{B}_j, \mathbf{X}, \mathbf{\Gamma}^{\{i,j\}}\right) - \mathbf{F}\left(\mathbf{A}_i, \mathbf{B}_i, \mathbf{X}, \mathbf{\Gamma}^{\{i,j\}}\right), \tag{16}$$

where $\Delta\mathbf{F}\left(*\right)^{\{i,j\}}$ is the perturbation matrix that represents the interaction between vertices i and j.

By analysing how much $\Delta\mathbf{F}\left(*\right)$ (15) affects the LMIs, it is possible to determine which vertices are more susceptible of being combinable. (Sanjuan et al., 2019) presented an algorithm to search for a combinable vertex pair, which was based on calculating a combinability ranking, using the steps detailed in Algorithm 1.

The algorithm as presented in (Sanjuan et al., 2019) has two issues. First of all, a combinability metric based on Gershgorin circles, referred in the following as $r_{\mathsf{GC}}^{\{i,j\}}$, is applied to $\Delta\mathbf{F}\left(*\right)$, which is equivalent to using an L_∞-norm. This introduces conservativeness due to the fact that the worst case-scenario for $\Delta\mathbf{F}\left(*\right)$ is considered.

Algorithm 1 : Vertex reduction ($n_v \rightarrow n_v - 1$)

1: Solve the LMIs (8) and obtain matrices $\mathbf{X}$ and $\mathbf{\Gamma}_v$. Note that a controller gain is designed for each vertex in this step
2: Determine $\Delta\mathbf{F}(*)^{\{i,j\}}$ (16) for each vertex pair
3: Compute a combinability metric $r_*^{\{i,j\}}$ for each $\Delta\mathbf{F}(*)^{\{i,j\}}$
4: Generate the combinability ranking by sorting $r_*^{\{i,j\}}$ in ascending order
5: Combine the vertices following the combinability ranking until a feasible solution is found or n_k attempts have been made

Secondly, the perturbation matrix $\Delta\mathbf{F}(*)$ is built by considering only the terms in the LMIs (15) which depend directly on the Lyapunov matrix $\mathbf{X}$, somehow neglecting the role of the input matrix $\mathbf{B}$ and the controller gains $\mathbf{\Gamma}^{\{i,j\}}$ (transformed through $\mathbf{X}$, see Eq.(9)).

In the following, these two issues will be attenuated by using a new metric to generate the combinability ranking (Section 3.2) and by redefining the perturbation matrix (Section 3.3). The decrease of conservativeness brought by these innovations will be later illustrated by means of an example (Section 3.4).

3.2. Metric definition

In Algorithm 1, a combinability metric is applied to the perturbation matrix $\Delta\mathbf{F}(*)$ in order to determine which vertices are more combinable. In (Sanjuan et al., 2019), the chosen metric $r_*^{\{i,j\}}$ was based on Gershgorin circles, as follows

$$r_{\mathsf{GC}}^{\{i,j\}} = ||\Delta\mathbf{F}(*)^{\{i,j\}}||_\infty = \max_k \left(|\Delta f_{kk}^{\{i,j\}}| + \sum_{\substack{l=1 \\ l\neq k}}^{n} |\Delta f_{kl}^{\{i,j\}}| \right), \tag{17}$$

where $\Delta f_{kl}^{\{i,j\}}$ is the element of matrix $\Delta\mathbf{F}(*)^{\{i,j\}}$ located at row k and column l. As stated previously, although quite intuitive, this combinability metric is conservative because it considers the "worst case scenario".

In this paper, we propose to use the Frobenius norm (Quarteroni et al., 2010) in order to improve the combinability ranking. More specifically, the Frobenius-based combinability metric is defined as

$$r_{\mathsf{F}}^{\{i,j\}} = ||\Delta\mathbf{F}(*)^{\{i,j\}}||_{\mathsf{F}} = \sqrt{\sum_{k=1}^{n}\sum_{l=1}^{n} |\Delta f_{kl}^{\{i,j\}}|^2} = \sqrt{\sum_{l=1}^{n} \sigma_l^2(\Delta\mathbf{F}(*)^{\{i,j\}})}, \tag{18}$$

where $\sigma_l(\Delta\mathbf{F}(*)^{\{i,j\}})$ denotes the singular values of $\Delta\mathbf{F}(*)^{\{i,j\}}$.

The Frobenius-based combinability metric (18) quantifies the perturbation matrix as the Euclidean norm of its singular values. The main difference between $r_{\mathsf{GC}}^{\{i,j\}}$ and $r_{\mathsf{F}}^{\{i,j\}}$ is that the former considers the worst possible location where the maximum real

value of the eigenvalues of the LMI can be, whereas the latter takes into account all singular values of the perturbation matrix.

3.3. Perturbation matrix definition

The perturbation matrix $\Delta\mathbf{F}(*)^{\{i,j\}}$ plays an important role in Algorithm 1 because it quantifies how much the negative definiteness of the matrix on the left-hand side of the LMIs (15) can be affected by the use of a common controller gain for the vertex pair $\{i, j\}$. It must be highlighted that an appropriate definition of this matrix might improve the performance of the combinability ranking, in the sense of placing the combinable vertex pairs in the top positions.

In this paper, we propose to analyse the effect of using a common controller for vertices i and j, generating two different perturbation matrices on the LMIs.

The perturbation matrix $\mathbf{\Delta F1}$ is built by considering that the solver generates a common controller gain $\mathbf{\Gamma}_l$ for vertices i and j

$$\begin{cases} \mathbf{F}\left(\mathbf{A}_i, \mathbf{B}_i, \mathbf{X}, \mathbf{\Gamma}_l\right) \prec 0 \\ \mathbf{F}\left(\mathbf{A}_j, \mathbf{B}_j, \mathbf{X}, \mathbf{\Gamma}_l\right) \prec 0 \end{cases} \Leftrightarrow \begin{cases} \mathbf{F}\left(\mathbf{A}_i, \mathbf{B}_i, \mathbf{X}, \mathbf{\Gamma}_l\right) \prec 0 \\ \mathbf{F}\left(\mathbf{A}_i, \mathbf{B}_i, \mathbf{X}, \mathbf{\Gamma}_l\right) + \mathbf{\Delta F1}^{\{i,j\}} \prec 0 \end{cases}, \tag{19}$$

whereas the second perturbation matrix $\mathbf{\Delta F2}$ is obtained by assuming that the solver provides a feasible solution when the controller gain $\mathbf{\Gamma}_j$ is used for vertices i and j

$$\begin{cases} \mathbf{F}\left(\mathbf{A}_i, \mathbf{B}_i, \mathbf{X}, \mathbf{\Gamma}_j\right) \prec 0 \\ \mathbf{F}\left(\mathbf{A}_j, \mathbf{B}_j, \mathbf{X}, \mathbf{\Gamma}_j\right) \prec 0 \end{cases} \Leftrightarrow \begin{cases} \mathbf{F}\left(\mathbf{A}_i, \mathbf{B}_i, \mathbf{X}, \mathbf{\Gamma}_i\right) - \mathbf{\Delta F2}^{\{i,j\}} \prec 0 \\ \mathbf{F}\left(\mathbf{A}_j, \mathbf{B}_j, \mathbf{X}, \mathbf{\Gamma}_j\right) \prec 0 \end{cases}. \tag{20}$$

Thus, the perturbation matrices, taking into account (19) and (20), are

$$\mathbf{\Delta F1}^{\{i,j\}} = \mathbf{F}\left(\mathbf{A}_j, \mathbf{B}_j, \mathbf{X}, \mathbf{\Gamma}_l\right) - \mathbf{F}\left(\mathbf{A}_i, \mathbf{B}_i, \mathbf{X}, \mathbf{\Gamma}_l\right), \tag{21}$$

$$\mathbf{\Delta F2}^{\{i,j\}} = -\left(\mathbf{F}\left(\mathbf{A}_i, \mathbf{B}_i, \mathbf{X}, \mathbf{\Gamma}_j\right) - \mathbf{F}\left(\mathbf{A}_i, \mathbf{B}_i, \mathbf{X}, \mathbf{\Gamma}_i\right)\right). \tag{22}$$

It must be highlighted that, from the set of LMIs generated by applying the specifications defined by the designer, the LMI in which only a single vertex is involved in its definition is the one considered to formulate the perturbation matrices (21) and (22).

When input matrix $\mathbf{B}$ is constant, the perturbation matrix (21) corresponds to the one presented in (Sanjuan et al., 2019). From the point of view of vertex i, $\mathbf{\Delta F1}^{\{i,j\}}$in (21) and $\mathbf{\Delta F2}^{\{i,j\}}$in (22) can be interpreted as perturbations due to the system and to the controller, respectively.

In this paper, the combinability metric that has been considered in Algorithm 1 to evaluate $\mathbf{\Delta F1}^{\{i,j\}}$ and $\mathbf{\Delta F2}^{\{i,j\}}$ is as follows

$$r_{*}^{\{i,j\}} = \delta_1 \cdot r_{*}^{\{i,j\}}\left(\mathbf{\Delta F1}^{\{i,j\}}\right) + \delta_2 \cdot r_{*}^{\{i,j\}}\left(\mathbf{\Delta F2}^{\{i,j\}}\right), \tag{23}$$

where δ_1 and δ_2 are the perturbation matrix weights. In the remaining of the paper, the weighting factors will be considered as $\delta = \delta_1 = \delta_2 = 0.5$ for the sake of simplicity. A detailed study of the optimal choice of these parameters as well as alternative expressions for the combinability metric falls out of the scope of this paper.

The structure of the perturbation matrices (21) and (22) will depend on the performance conditions imposed by the designer. For example, when a quadratic $\mathcal{D}-$stability specification, with a circular region with center $(-q, 0)$ and radius r, is considered and the matrix $\mathbf{B}$ is common for each vertex, the LMI is given by (Chilali & Gahinet, 1996)

$$\boldsymbol{\mathcal{D}}_{r_v} = \begin{pmatrix} -r\mathbf{X} & q\mathbf{X} + \mathbf{G}_v \\ q\mathbf{X} + \mathbf{G}_v^T & -r\mathbf{X} \end{pmatrix} \prec 0, \tag{24}$$

where

$$\mathbf{G}_v = \mathbf{A}_v\mathbf{X} + \mathbf{B}\boldsymbol{\Gamma}_v. \tag{25}$$

Then, the perturbation matrices are the following

$$\Delta\mathbf{F1}^{\{i,j\}} = \begin{pmatrix} 0 & (\mathbf{A}_j - \mathbf{A}_i)\mathbf{X} \\ \mathbf{X}(\mathbf{A}_j - \mathbf{A}_i)^T & 0 \end{pmatrix}, \tag{26}$$

$$\Delta\mathbf{F2}^{\{i,j\}} = -\begin{pmatrix} 0 & \mathbf{B}(\boldsymbol{\Gamma}_j - \boldsymbol{\Gamma}_i) \\ (\boldsymbol{\Gamma}_j - \boldsymbol{\Gamma}_i)^T\mathbf{B}^T & 0 \end{pmatrix}. \tag{27}$$

For the interested reader, Appendix A reports the perturbation matrices in the case where a guaranteed $\mathcal{H}_\infty$ bound were considered as performance specification.

Algorithm 2 is based on Algorithm 1 and incorporates the improvements described in Sections 3.2 and 3.3.

Algorithm 2 : Improved vertex reduction ($n_v \rightarrow n_v - 1$)

1: Solve the LMIs (8) and obtain matrices $\mathbf{X}$ and $\boldsymbol{\Gamma}_v$ (non-combined vertices)
2: Determine $\Delta\mathbf{F1}^{\{i,j\}}$ and $\Delta\mathbf{F2}^{\{i,j\}}$ using (21)-(22) for each vertex pair
3: Compute $r_{\bar{\mathsf{F}}}^{\{i,j\}}$ as in (23)
4: Generate the combinability ranking by sorting $r_{\bar{\mathsf{F}}}^{\{i,j\}}$ in ascending order
5: Combine the vertices following the combinability ranking until a feasible solution is found or n_k attempts have been made

3.4. Illustrative example

The benefits of Algorithm 2 in contrast to Algorithm 1 are assessed by means of one among the sets of academic systems previously considered in (Sanjuan et al., 2019). This set consists of 250 academic systems with three states, three inputs and five vertices. The objective is to design $n_v - 1$ controller gains for each system, considering that each of them has a maximum number of combinable vertex pairs equal to 2.

Fig. 1 presents the results obtained using Algorithms 1 and 2 in order to find a combinable vertex pair in fewer attempts. As summarized in Table 1, the results denoted as L1 correspond to the application of the method proposed in (Sanjuan et al., 2019) which uses the combinability metric r_{GC} and only the perturbation matrix $\mathbf{\Delta F1}$. On the other hand, the results denoted as L2, L3 and L4 show the improvements brought by the innovations proposed in the previous sections.

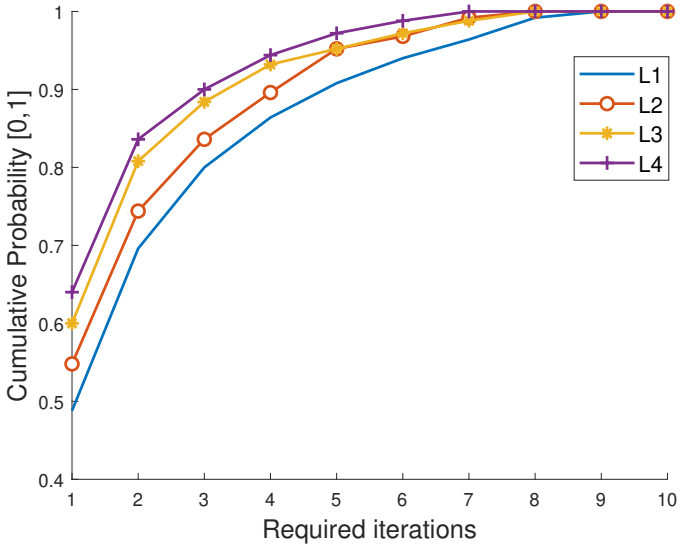

Figure 1.: Vertex reduction from n_v to $n_v - 1$. Comparison between different approaches.

Label	Metric $\Delta\mathbf{F1}$	Metric $\Delta\mathbf{F2}$
L1	$r_{\mathsf{GC}}^{\{i,j\}}\left(\mathbf{\Delta F1}^{\{i,j\}}\right)$	0
L2	$r_{\mathsf{F}}^{\{i,j\}}\left(\mathbf{\Delta F1}^{\{i,j\}}\right)$	0
L3	$r_{\mathsf{GC}}^{\{i,j\}}\left(\mathbf{\Delta F1}^{\{i,j\}}\right)$	$r_{\mathsf{GC}}^{\{i,j\}}\left(\mathbf{\Delta F2}^{\{i,j\}}\right)$
L4	$r_{\mathsf{F}}^{\{i,j\}}\left(\mathbf{\Delta F1}^{\{i,j\}}\right)$	$r_{\mathsf{F}}^{\{i,j\}}\left(\mathbf{\Delta F2}^{\{i,j\}}\right)$

Table 1.: Description of the labels L1, L2, L3 and L4

L2 uses the same perturbation matrix as L1, but the combinability ranking is obtained using r_{F}. It can be seen that L2 outperforms L1. For instance, a combinable vertex pair was found at the first attempt in about 55% of the 250 systems when L2 was used, whereas only in about 49% of the systems when L1 was considered.

On the other hand, the results denoted by L3 have been obtained using both perturbation matrices $\mathbf{\Delta F1}$ and $\mathbf{\Delta F2}$, keeping the same combinability metric as L1, i.e. $r_{\overline{\mathsf{GC}}}^{\{i,j\}}$ (23), which provides an even bigger improvement. Indeed, L3 improves the probability of finding a combinable vertex pair in the first iteration of about 11%.

Finally, the best results have been obtained when both innovations are applied (results denoted as L4). In this case, which corresponds to Algorithm 2, a combinable vertex pair was found in about 64% of the systems during the first iteration, and in more than 80% in the second iteration.

It can be concluded that the innovations discussed in this section improve the performance of the combinability ranking, and are worth of being considered in the development of a complete algorithm that achieves further reduction of vertices from n_v to n_c, which will be done in the following section.

4. Searching for a minimal combinable partition

The goal of this section is to describe an algorithm that allows to find a combinable partition with the minimal (or close to the minimal) number of elements that preserves the original performance specifications. This combinatorial optimization problem is modelled as an ordered tree, where every node represents a partition of the set of vertices of the LPV system. The root node corresponds to the partition whose elements are all singletons. The child nodes of a given node are ordered according to the combinability metric of their corresponding partition.

4.1. Combinability metric for a partition

The combinability metric described in Section 3.2 considered the case in which two vertices were to be combined. Hereafter, a metric for evaluating the combinability of a partition $\mathcal{P}$ is proposed, as follows

$$r_{\diamond}^{\mathcal{P}} = \max_{p \in \mathcal{P}} r_{\diamond}^{p}, \tag{28}$$

where $r_{\diamond}^{p}$ denotes the combinability metric for the set p, defined as

$$r_{\diamond}^{p} = \begin{cases} \max\limits_{i,j \in p;\ i<j} r_{\diamond}^{\{i,j\}} & |p| > 1 \\ 0 & |p| = 1 \end{cases}, \tag{29}$$

and taking into account that p is an element of the partition $\mathcal{P}$, $|p|$ is the cardinality of the set p, i and j are vertices of the set p and $r_{\diamond}^{\{i,j\}}$ is a two vertex combinability metric, e.g. the Frobenius-based metric (23), which will be used throughout the remaining of this paper.

The combinability metric of a partition (29) is induced from the fact that a single non-combinable vertex pair in a vertex set p of the partition $\mathcal{P}$ ($p \in \mathcal{P}$) determines that this distribution is not combinable.

4.2. Recursive vertex reduction

A solution to the combinatorial optimization problem will be provided by the heuristic depth-first search algorithm denoted as Algorithm 3. This algorithm, which is based on Algorithm 2, tries to find a lesser-fragmented combinable partition at each recursive call. Vertex reduction is achieved by combining two elements of a combinable partition in order to decrease its cardinality. The result of this procedure will be denoted by *level*.

Definition 4.1 (Candidate partition)**.** ς is a candidate partition extracted from the combinable partition $\mathcal{S}$, if ς is obtained from $\mathcal{S}$ by replacing two of its elements with their union, as follows

$$\varsigma = (\mathcal{S} \backslash \{s_m, s_n\}) \cup (s_m \cup s_n) \quad s_m, s_n \in \mathcal{S}. \tag{30}$$

Definition 4.2 (Level candidates set)**.** Σ is a level candidates set for the combinable partition $\mathcal{S}$, if Σ contains all candidate partitions extracted from $\mathcal{S}$:

$$\Sigma = \{\varsigma_1, \varsigma_2, ..., \varsigma_{n_{cp}}\}, \tag{31}$$

where ς_i is a candidate partition and n_{cp} is the number of partitions that can be generated by combining two elements of $\mathcal{S}$.

Algorithm 3 requires the following inputs to proceed with the reduction:

- the structure *node*, which refers to the corresponding combinable partition *node.*$\mathcal{S}$ and the set of assessed partitions *node.setAssdPart* which records the partitions that have been already assessed during the search until reaching this node;
- $\mathcal{S}^*$, which represents the current solution (i.e., the partition with the minimal number of elements);
- *SysData*, which contains all the information required to compute the LMIs;
- the structure *param*, which stores three parameters $\texttt{sLVL}_{max}$, n_k, and $\texttt{nLMI}_{max}$, which constrain the search;

whereas the output of the function is the updated solution $\mathcal{S}^*$.

The search starts by generating the level candidates set Σ (step 1), combining all the possible combinations of two elements of *node.*$\mathcal{S}$, and evaluating them using a combinability metric (step 2). The LMIs that correspond to a candidate partition are evaluated (at step 14 by function *isCombinablePartition*) following the level candidates ranking (i.e. sorting the level candidate set in increasing combinability metric order), as long as the corresponding partition has not been considered previously. The function *isNewPartition* (step 10) determines whether a certain candidate partition has been analysed already or was discarded during this search. This information is determined taking into account the trace of candidate partitions corresponding to already assessed nodes (stored in *childNode.setAssdPart*).

If the LMIs provide a feasible solution, which means that a combinable partition *childNode.*$\mathcal{S}$ has been found, $\mathcal{S}^*$ is updated as long as this child node partition has fewer elements than the current best solution (step 18). Then, a new level is created

(step 20). Indeed, if a candidate partition of a given node is not combinable, neither will be the partitions that correspond to its descendants. When a level is fully analysed, the search algorithm backtracks to the previous level, by returning $\mathcal{S}^*$ (step 24).

In order to reduce the computational burden of the depth-first search, three parameters ($\texttt{sLVL}_{max}$, n_k, and $\texttt{nLMI}_{max}$) are used to constrain the search at step 7. At most, n_k candidate partitions per level will be assessed, from which $\texttt{sLVL}_{max}$ levels will be created at most. The remaining candidate partitions of the current level, and those corresponding to their descendants, are discarded. The parameter $\texttt{nLMI}_{max}$ sets an upper bound on the total number of LMIs to be solved during the search, which is accounted for by the global variable $n_{\texttt{LMI}}$. The search terminates if this bound is reached or the best reduction is achieved (condition $|S^*| = 1$), which means that all vertices will use the same controller gain (i.e., a robust controller gain exists for the considered system and design specifications).

To prune the search properly, a suitable choice of parameters n_k and $\texttt{sLVL}_{max}$ is required. On the one hand, the set of candidate partitions to be assessed per level is constrained to the n_k partitions with a higher combinability metric. Thus, this parameter should be set according to the expected probability of success in finding a combinable partition, which depends on the metric performance (see Fig. 1). Choosing a lower value for n_k is recommended for those cases where finding combinable partitions among the first positions of the level candidates ranking is very likely.

On the other hand, the set of combinable partitions per level for which a new level will be created is constrained to the $\texttt{sLVL}_{max}$ partitions with a higher combinability metric. Thus, this parameter allows to further constrain the size of the search tree. Additionally, note that $\texttt{sLVL}_{max} \leq n_k$ should hold.

Finally, the search parameter $\texttt{nLMI}_{max}$ has an important role in complex systems. The size of the search tree increases exponentially with the number of vertices of the LPV system, as long as $\texttt{sLVL}_{max}$ is bigger than one. Hence, $\texttt{nLMI}_{max}$ limits the computational burden of the search.

In the worst case, Algorithm 3 will have to explore all possible partitions, which corresponds to the Bell number. Therefore, the algorithm has worst-case exponential time complexity. However, the fact that even the partitions that correspond to the node descendants of a non-combinable partition are not combinable makes Algorithm 3 have a much better average-case performance than a brute-force search.

The parameters defined in Algorithm 3 reduce the worst-case time complexity of the search at the expense of optimality. Parameter $\texttt{nLMI}_{max}$ provides constant time complexity, whereas n_k and $\texttt{sLVL}_{max}$, subexponential complexity. Fig. 2 shows the worst-case complexity dependence of Algorithm 3 on n_k for a particular number of vertices, $n_v = 8$, $\texttt{sLVL}_{max} = n_k$ and $\texttt{nLMI}_{max} = 4500$.

The range of possible values for n_k goes from 1 to the number of candidate partitions at level $n_v - 1$, which is $n_v(n_v - 1)/2$. In general, a low value for n_k must be chosen in order to achieve a notable reduction of the execution time of Algorithm 3. For instance, in this particular example, to get a 50% reduction on the worst-case running time of the algorithm, n_k should be set to 6.

Algorithm 3 : $\mathcal{S}^*$:= SearchMinimalPartition(node, $\mathcal{S}^*$, SysData, param)

 # $n_{\texttt{LMI}}$ is a global variable initialized to 0 #
Generate the level candidates set Σ that corresponds to node.$\mathcal{S}$
Compute the combinability metric $r_{\mathsf{F}}^{\{\varsigma_i\}}$ (28) for each $\varsigma_i \in \Sigma$
numCombPart:=0
numCheckedPart:=0
childNode.setAssdPart:=node.setAssdPart
for all $\varsigma \in \Sigma$ in increasing combinability metric order **do**
    if $|S^*| = 1$ **or** $n_{\texttt{LMI}}$=param.$\texttt{nLMI}_{max}$ **or**
     (numCombPart=param.$\texttt{sLVL}_{max}$ **or** numCheckedPart=param.n_k) **then**
        return S^*
    end if
    if isNewPartition(ς,childNode.setAssdPart) **then**
        numCheckedPart:=numCheckedPart+1
        childNode.setAssdPart:=childNode.setAssdPart $\cup \{\varsigma\}$
        $n_{\texttt{LMI}}$:=$n_{\texttt{LMI}}$+1
        if isCombinablePartition(ς,SysData) **then**
            numCombPart:=numCombPart+1
            childNode.$\mathcal{S}$:=ς
            if |childNode.$\mathcal{S}$| < $|\mathcal{S}^*|$ **then**
                S^*:= childNode.$\mathcal{S}$
            end if
            $\mathcal{S}^*$:= SearchMinimalPartition(childNode, $\mathcal{S}^*$, SysData, param)
        end if
    end if
end for
return S^*

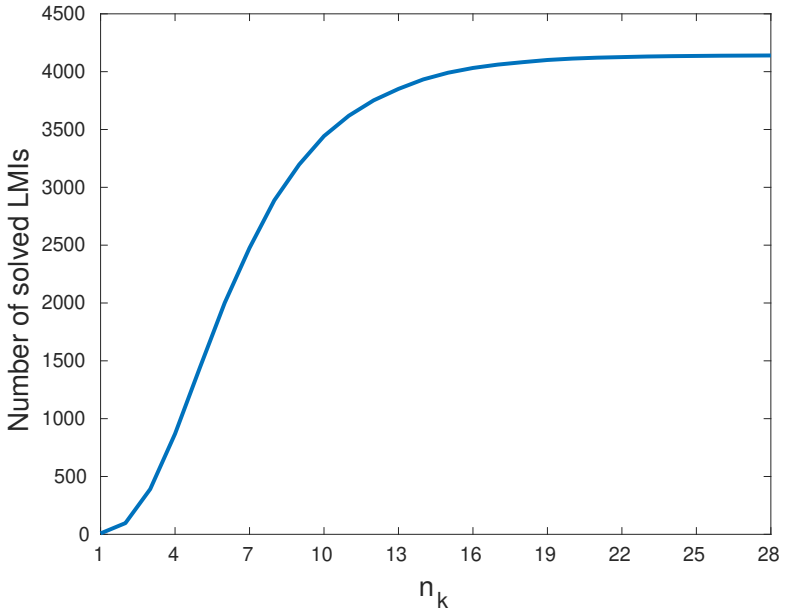

Figure 2.: Worst-case complexity dependence on n_k for $n_v = 8$

5. Illustrative examples

Three examples are considered in this section in order to assess the efficiency of Algorithm 3. First, a numerical example is used to describe in detail each step of Algorithm 3 for illustrative purposes. Second, it is shown that the number of controller gains for the TRMS, which is a complex system with 256 vertices, can be reduced by applying the proposed algorithm. Finally, a two-link robotic manipulator is used to demonstrate the applicability of the proposed approach to cases where the LPV system contains a parameter-varying input matrix.

5.1. Academic example

Consider a system that has 3 states, 2 inputs and 5 vertices with vertex state matrices $\mathbf{A}_v$ and input matrix $\mathbf{B}$ as follows

$$\mathbf{A}_1 = \begin{bmatrix} 6.63 & -2.01 & 3.14 \\ 6.07 & 0.54 & 2.56 \\ -8.80 & -1.66 & -4.16 \end{bmatrix} \qquad \mathbf{A}_2 = \begin{bmatrix} -1.37 & -6.66 & -6.04 \\ -9.69 & -7.88 & -0.21 \\ 9.68 & -2.55 & -3.21 \end{bmatrix}$$

$$\mathbf{A}_3 = \begin{bmatrix} 9.03 & 4.75 & 0.96 \\ 8.41 & -4.62 & 8.85 \\ -8.95 & -1.54 & -1.65 \end{bmatrix} \qquad \mathbf{A}_4 = \begin{bmatrix} 9.66 & 3.33 & 3.33 \\ -3.97 & 0.78 & -6.44 \\ 4.02 & 3.96 & -7.44 \end{bmatrix}$$

$$\mathbf{A}_5 = \begin{bmatrix} 9.98 & 1.22 & -6.19 \\ -6.58 & 7.64 & -2.62 \\ -9.35 & 3.38 & -0.79 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0.96 & 0.29 \\ -0.69 & -0.25 \\ 0.71 & -0.62 \end{bmatrix}.$$

The chosen performance specification is quadratic $\mathcal{D}-$stability of the closed-loop system with an LMI region $\mathcal{D}(\alpha, \gamma, r)$ composed by an α-stability region, a conic sector with angle γ and a disk of radius r and centre $(-q, 0)$ (Sanjuan et al., 2019). YALMIP (Lofberg, 2004) and SeDuMi (Sturm, 1999) have been used to model and solve the LMIs, respectively. For the remaining of the example, let us consider the case where the LMI region of interest is $\mathcal{D}(1, 0.6, 44.4)$, for which the matrices $\mathbf{X}$ and $\mathbf{\Gamma}_v$, returned in the non-combined vertices case, are the following

$$\mathbf{X} = \begin{bmatrix} 2.46 & -2.95 & -3.27 \\ -2.95 & 3.68 & 4.13 \\ -3.27 & 4.13 & 5.36 \end{bmatrix} \cdot 10^{-2},$$

$$\mathbf{\Gamma}_1 = \begin{bmatrix} -0.25 & 0.25 & 0.16 \\ -1.02 & 1.01 & 1.16 \end{bmatrix} \quad \mathbf{\Gamma}_2 = \begin{bmatrix} -0.59 & 0.70 & 0.74 \\ -1.10 & 1.27 & 1.89 \end{bmatrix} \quad \mathbf{\Gamma}_3 = \begin{bmatrix} -0.18 & 0.15 & 0.02 \\ -0.94 & 0.90 & 0.88 \end{bmatrix}$$
$$\mathbf{\Gamma}_4 = \begin{bmatrix} -0.23 & 0.22 & 0.15 \\ -0.50 & 0.43 & 0.79 \end{bmatrix} \quad \mathbf{\Gamma}_5 = \begin{bmatrix} -0.30 & 0.33 & 0.30 \\ -1.22 & 1.35 & 2.11 \end{bmatrix}.$$

Brute-force search has been used to evaluate for which partitions the LMIs are satisfied, determining that the ones with the lowest number of elements are: $\{\{1, 3, 4, 5\}, \{2\}\}$ and $\{\{1, 3, 5\}, \{2, 4\}\}$. It must be highlighted that the vertices are highly combinable in the selected region (Fig. 3), since 9 out of 10 level 4 candidate partitions are combinable. Note that the levels are numbered according to the cardinality of the corresponding candidate partitions, where each level corresponds to a call to Algorithm 3.

The behaviour of Algorithm 3 is detailed below when the following search parameters are set: $\texttt{sLVL}_{max} = 1$, $n_k = 2$, and $\texttt{nLMI}_{max} = 20$. The initial solution corresponds to the partition of the root node: $\mathcal{S}^* = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$. The nodes connected by a blue line in Fig. 3 correspond to the search sequence generated by these parameters.

The algorithm starts by evaluating the combinability metric of each vertex pair (steps 1 and 2) to determine the level 4 candidates ranking. Note that the combinability metrics are shown in the figure as a number located above the arrows. Thereafter, the algorithm selects the first candidate partition $\{\{1, 3\}, \{2\}, \{4\}, \{5\}\}$ (first iteration of the for loop, step 6, at level 4), for which the best combinability metric has been obtained (1.70), and computes the LMIs taking into account that vertices 1 and 3 use the same controller gain (step 14). In this case, the solver provides a feasible solution (denoted by a green background in Fig. 3), which means that a combinable partition has been found and the new level 3.A is created (step 20 - new call to Algorithm 3). Additionally, the current solution $\mathcal{S}^*$ is updated (step 18) since a combinable partition with fewer elements has been found.

After determining the level 3.A candidates ranking, the algorithm verifies that the LMIs for the first candidate partition $\{\{1, 3, 4\}, \{2\}, \{5\}\}$ provides a feasible solution (first iteration of the for loop, step 6, at level 3.A). Consequently, the solution is updated and level 2.A.A candidates set is generated. The algorithm determines that the first two partitions of level 2.A.A are not combinable partitions and backtracks to level 3.A (step 7), since $n_k = 2$. Moreover, taking into account that $\texttt{sLVL}_{max} = 1$, the algorithm further backtracks to level 4 and to the root node, since a combinable

partition has already been found at those levels (step 7). As a result, the search terminates, providing the current best solution: $\mathcal{S}^* = \{\{1,3,4\},\{2\},\{5\}\}$.

By setting $\texttt{sLVL}_{max} = 2$, a deeper search will be accomplished. The algorithm follows the search sequence indicated by the blue and orange lines in Fig.3. In this case, a second branch is explored after backtracking to level 3.A, updating the current solution to $\{\{1,3,5\},\{2,4\}\}$. Note that, the partition $\{\{1,2,4,3,5\}\}$ is not analysed, since level 1 was discarded when level 2.A.A partitions provided an infeasible solution (step 10). Later, when backtracking to level 4, a second branch is also explored, but no better solution is found. It must be highlighted that the first partition of level 3.B $\{\{1,3,4\},\{2\},\{5\}\}$ is discarded from the search (step 10), since it had been previously analysed in level 3.A. Finally, the search terminates, providing the solution: $\mathcal{S}^* = \{\{1,3,5\},\{2,4\}\}$. Thus, by changing a search parameter ($\texttt{sLVL}_{max}$, in this case) a better solution has been found, although at the cost of increasing the overall computational load. Note that this solution corresponds to one of the partitions with the minimal number of elements (optimal solution).

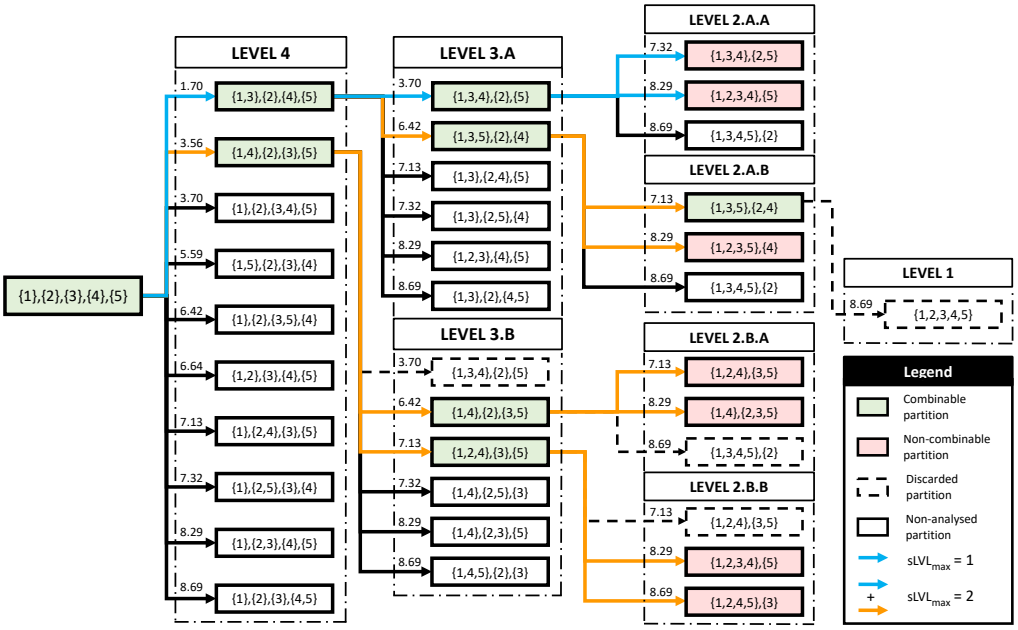

Figure 3.: Search sequences (Algorithm 3) for an LPV system with 5 vertices.

Fig. 4 depicts the cardinality of the minimal partition $|S^*_{bf}|$ obtained using brute-force search as a function of the radius r of the LMI region $\mathcal{D}(1, 0.6, r)$ (performance specification). A feasible solution of the control design problem for this academic example has been found for a radius of about 22.9. By increasing the radius, a lesser-fragmented partition can be further attained until reaching the case where a common controller gain can be applied to each vertex, which means that a robust controller gain exists for this performance specification (for a value of $r \approxeq 64$).

Given the results in Fig. 4, it can be concluded that the combinability of the elements of a partition is correlated with the performance specifications. Hence, when defining a small value for the parameter n_k, Algorithm 3 will carry out a lesser-exhaustive search, determining whether a combinable partition for a given performance specifications exists. Therefore, when assessing the combinable sets of the obtained partition, a

deeper search could be considered to further improve this result or if an attenuation of the performance specifications is required in case a more compact controller ought to be attained.

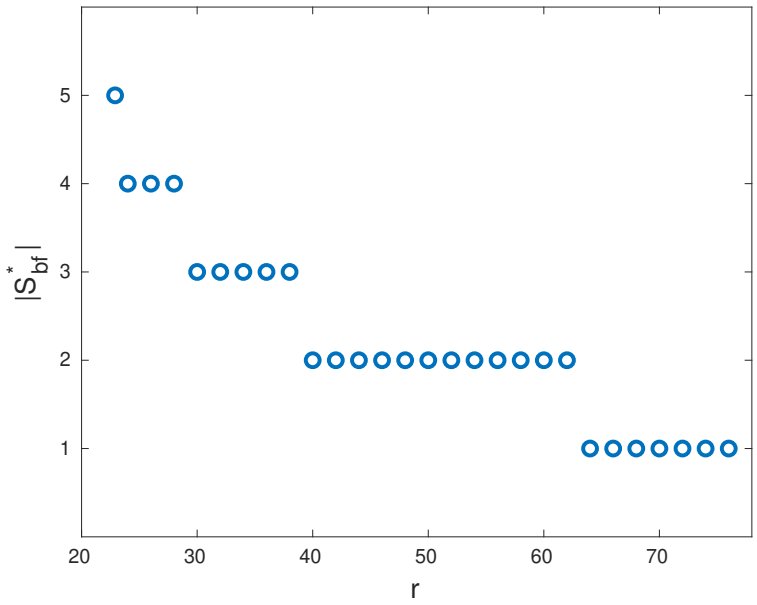

Figure 4.: Cardinality of the minimal partition as a function of the region of the performance specification $\mathcal{D}(1, 0.6, r)$

5.2. Twin-rotor MIMO system (TRMS)

The TRMS is an aero-dynamical system, which consists of two rotors located at the ends of a beam with perpendicular orientations between each other. This configuration allows to change the orientation of the pitch and yaw angles, thus emulating the behaviour of a helicopter.

The TRMS is a complex nonlinear system for which the direct transformation in a quasi-LPV system using the bounding box technique would lead to 2048 vertex systems (11 scheduling parameters) (Rotondo et al., 2013). However, as discussed in (Rotondo et al., 2013), three scheduling parameters can be approximated by a constant value because their variations have a relative small effect on the system, thus resulting into 256 vertex systems obtained from 8 scheduling parameters. In this section, the 256 vertex TRMS polytopic model has been used to apply the proposed methodology and determine a polytopic controller with a minimal number of controller gains. It is worth remarking that, for this complex system, there are no means to determine if the best combinable partition provided by Algorithm 3 corresponds to a possible optimal solution, since a brute-force search is computationally prohibitive.

The considered model of the TRMS is the continuous-time representation which is detailed in (Rotondo et al., 2013). The performance specifications for controller design is pole clustering in the region $\mathcal{D}(\alpha, \gamma, r)$ with $\alpha = 1$, $\gamma = 1$ and $r = 65$.

A search following Algorithm 3 with $\texttt{sLVL}_{max} = 1$, $n_k = 15$, and $\texttt{nLMI}_{max} = 300$ returns a combinable partition with 3 elements, which means that the number of controller gains to be implemented has been reduced from 256 to 3.

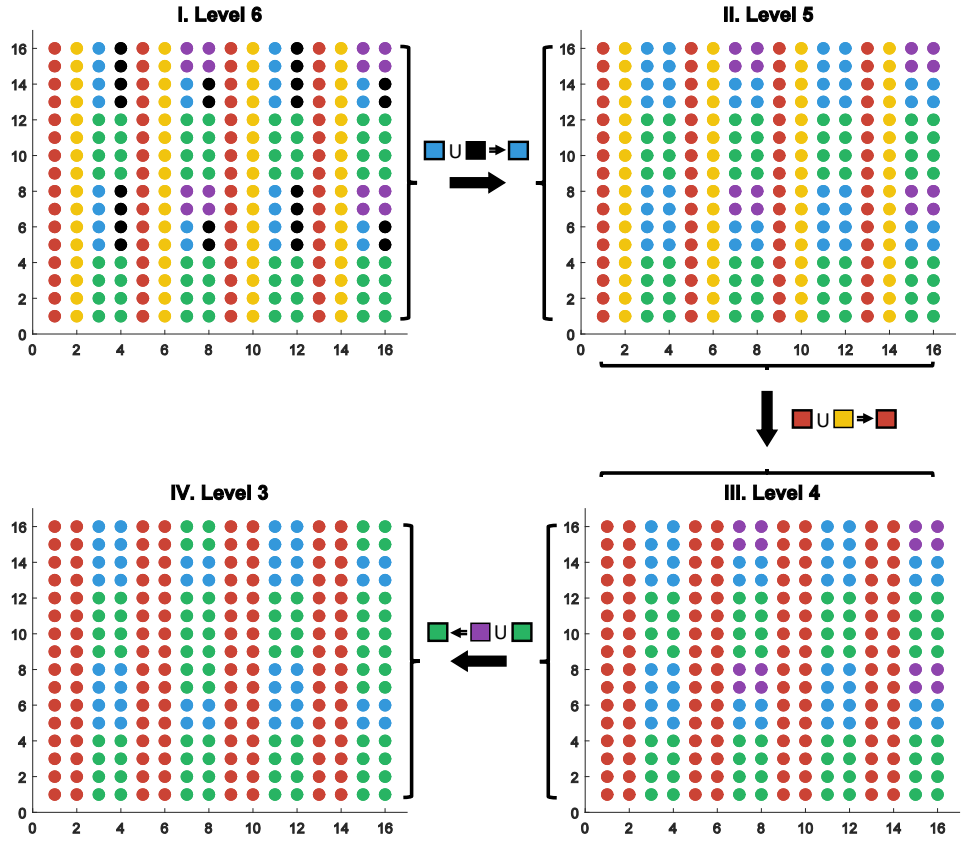

Figure 5.: Graphical representation of the last iterations of Algorithm 3 (TRMS).

During the search, 265 LMIs are solved (i,e, 265 candidates partitions are evaluated) and 254 levels are created until the best solution is found. Fig. 5 presents the last four updates of the current solution (step 18 of Algorithm 3), which occur at levels 6, 5, 4 and 3. A circle located in the i-th row and j-th column represents the $16(i-1)+j$-th vertex, to which a controller gain is assigned based on the represented color. At level 6 (Fig. 5-I), the algorithm determines that black and blue vertices can be combined. Thus, the number of controllers is reduced from six to five at Fig. 5-II. Next, in the following iteration, the algorithm combines yellow and red vertices and so on, until reaching level 3. At this point, the search terminates since the LMIs for the candidate partitions at level 2 are not feasible.

In order to show how hard is to find combinable partitions for this system, we considered 265 random partitions (corresponding to the number of LMIs that the proposed algorithm has computed) with a fixed number of elements and verified whether these random partitions were combinable. The results are presented in Fig. 6. For instance, when the number of elements is set to 255, about 48% of the 265 candidates correspond to combinable partitions, whereas this number reduces to only about 1% if the number of elements is set to 250. Hence, according to Fig. 6, reducing the number of controller gains is not an easy task for the TRMS example, and Algorithm 3 is a

useful tool to select efficiently which vertices can share a common controller gain.

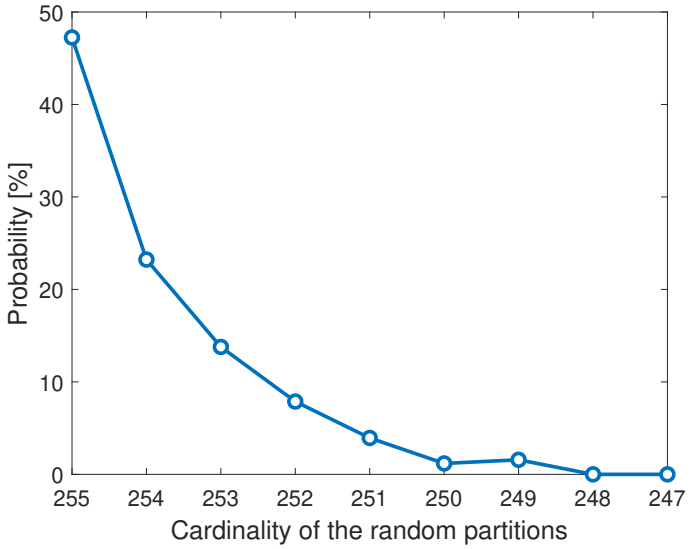

Figure 6.: Probability of generating randomly a combinable partition for the TRMS system

5.3. Two-link robotic manipulator

The two-link robotic manipulator example is used to demonstrate the efficiency of Algorithm 3 even in cases where the input matrix is not constant. This system consists of two rigid links, joined by one end, that can rotate around the horizontal plane. As described in (Tseng, Chen, & Uang, 2001), the model can be represented by nine vertex matrices.

The LMIs used for controller design have been obtained using the method proposed in (Sala & Arino, 2007), that allows handling a polytopic input matrix and is based on the application of Polya's theorem on definite quadratic forms. The procedure to build the LMIs is as follows:

i. Select the value of p. The parameter $p \in \mathbb{N}$ $(p \geq 2)$ defines a trade-off between conservativeness and computational burden. A larger value of p means that less conservative conditions are obtained, but the computational burden will be harder.
ii. Define the tuple of indices $\mathbb{I}_p^+$ as

$$\mathbb{I}_p^+ = \{(i_1, ..., i_p) \mid i_w \in \mathcal{V},\ i_w \leq i_{w+1}\}. \tag{32}$$

iii. Build the LMIs

$$\sum_{\mathbf{z} \in \mathcal{P}(\mathbf{i})} \mathbf{F}_{z_1 z_2} \leq 0 \qquad \forall \mathbf{i} \in \mathbb{I}_p^+, \tag{33}$$

where $\mathbf{F}_{z_1 z_2} := \mathbf{F}\left(\mathbf{A}_{z_1}, \mathbf{B}_{z_1}, \mathbf{X}, \mathbf{\Gamma}_{z_2}\right)$ and $\mathcal{P}(\mathbf{i})$ is the set of permutations of $\mathbf{i}$.

According to (33), the i-th LMI is the sum of $\mathbf{F}_{z_1 z_2}$, where the z_1 and z_2 correspond to the first and the second position, respectively, of an element of $\mathcal{P}(\mathbf{i})$.

Pole placement in $\mathcal{D}(1, 0.6, 9)$ has been selected as performance specification for the controller design and $p = 3$ in (32) to build the LMIs. The vertex reduction has been performed using Algorithm 3 with parameters $\texttt{sLVL}_{max} = 1$, $n_k = 3$, and $\texttt{nLMI}_{max} = 25$, concluding that a polytopic controller can be designed with five gains, corresponding to the following partition of vertices: $\{\{1, 9\}, \{2, 8\}, \{3, 7\}, \{4, 6\}, \{5\}\}$.

By performing a deeper search, which means increasing $\texttt{sLVL}_{max}$ from 1 to 2, the following combinable partition with fewer vertices is found: $\{\{1, 9\}, \{2, 4, 6, 8\}, \{3, 7\}, \{5\}\}$. Note that, using a brute-force search, it has been verified that this solution corresponds to the partition with the minimal number of elements. It must be also pointed out that there exists just one out of 7770 possible four element partitions that is feasible, and Algorithm 3 finds this partition in only nine attempts (i.e. the feasibility of the LMIs have been assessed for only nine candidate partitions before the above solution is found).

6. Conclusions

In this paper, a heuristic algorithm (Algorithm 3) has been developed to reduce the number of gains of a polytopic LPV state-feedback controller, whose performance specifications are represented by a set of parametrized LMIs. The proposed algorithm combines two elements of a partition recursively, in such a manner that at each iteration a lesser-fragmented combinable partition that preserves the original performance specifications is found. It must be highlighted that the developed algorithm can be used for vertex reduction in any LMI-based design, e.g. the design of an observer-based fault estimator.

The combinability metric is the core of the algorithm because it increases the chance of selecting a combinable partition. For this reason, two innovations have been introduced to decrease the conservativeness of the metric: the evaluation of the metric using the Frobenius norm and the inclusion of the input matrix in the definition of the LMIs perturbation matrix. The combination of both innovations, which corresponds to Algorithm 2, improves the performance of the combinability ranking, in the sense of placing the combinable partitions/vertex pairs in the top positions of the ranking. It must be highlighted that the proposed vertex reduction algorithm depends strongly on the specific value of the decision variables, so that its performance could depend on the specific LMI solver/parser as well as on the initial seeds provided to the solvers. In spite of this fact, it was shown that the proposed algorithm was able to perform effectively a reduction of the total number of vertex gains.

The number of possible partitions of a set, which corresponds to the Bell number, grows exponentially with the cardinality of the set itself. This fact prevents the development of an optimal search algorithm due to the computational complexity of the problem. To overcome such complexity, an algorithm (Algorithm 3) that introduces some constraints to the search has been developed and assessed in an academic example. As a result, this algorithm finds a combinable partition with the minimal (or close to the minimal) number of elements much more efficiently than through a brute-force

search.

The efficiency of the algorithm has been analysed further with two realistic applications. First, a polytopic controller has been designed for the TRMS and, by applying the developed algorithm, the number of gains has been reduced from 256 to 3. Second, using a two-link robotic manipulator, it has been shown that the proposed methodology can also be applied to systems with a parameter-varying input matrix.

Appendix A. Perturbation matrix definition: $\mathcal{H}_\infty$ performance specification

In this section, the perturbation matrices $\Delta\mathbf{F1}$ and $\Delta\mathbf{F2}$ will be provided for the case when a $\mathcal{H}_\infty$ performance specification is chosen. Consider the continuous time LPV system

$$\begin{cases} \dot{\mathbf{x}}(t) = \sum_{v=1}^{n_v} \alpha_v(\boldsymbol{\theta}(t)) \left(\mathbf{A}_v\mathbf{x}(t) + \mathbf{E}_v\mathbf{w}(t)\right) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{z}_\infty(t) = \mathbf{C}_w\mathbf{x}(t) + \mathbf{D}_w\mathbf{w}(t) \end{cases}, \tag{A1}$$

where $\mathbf{w}(t) \in \mathbb{R}^{n_w}$ is an exogeneous input, $\mathbf{E}_v \in \mathbb{R}^{n_x \times n_w}$ and $\mathbf{D}_w \in \mathbb{R}^{n_z \times n_w}$ are the disturbance distribution matrices, and $\mathbf{z}_\infty(t) \in \mathbb{R}^{n_z}$ denotes the controlled outputs.

The LMIs which allow to design an LPV controller guaranteeing quadratic H_∞ performance (Apkarian et al., 1995) are described by

$$\mathbf{F}(\mathbf{A}_v, \mathbf{B}, \mathbf{E}_v, \mathbf{X}, \boldsymbol{\Gamma}_v) = \begin{pmatrix} (\mathbf{A}_v\mathbf{X} + \mathbf{B}\boldsymbol{\Gamma}_v)^T + \mathbf{A}_v\mathbf{X} + \mathbf{B}\boldsymbol{\Gamma}_v & \mathbf{E}_v & (\mathbf{C}_w\mathbf{X})^T \\ \mathbf{E}_v^T & -\gamma\mathbf{I} & \mathbf{D}_w^T \\ \mathbf{C}_w\mathbf{X} & \mathbf{D}_w & -\gamma\mathbf{I} \end{pmatrix} \prec 0 \quad \forall\, v = 1, ..., n_v, \tag{A2}$$

with $\mathbf{X} \succ 0$.

In this case, the definition of $\Delta\mathbf{F}1^{\{i,j\}}$ and $\Delta\mathbf{F}2^{\{i,j\}}$ will be as follows

$$\Delta\mathbf{F1}^{\{i,j\}} = \mathbf{F}\left(\mathbf{A}_j, \mathbf{B}, \mathbf{E}_j, \mathbf{X}, \boldsymbol{\Gamma}_l\right) - \mathbf{F}\left(\mathbf{A}_i, \mathbf{B}, \mathbf{E}_i, \mathbf{X}, \boldsymbol{\Gamma}_l\right), \tag{A3}$$

and

$$\Delta\mathbf{F2}^{\{i,j\}} = -\left(\mathbf{F}(\mathbf{A}_i, \mathbf{B}, \mathbf{E}_i, \mathbf{X}, \boldsymbol{\Gamma}_j) - \mathbf{F}(\mathbf{A}_i, \mathbf{B}, \mathbf{E}_i, \mathbf{X}, \boldsymbol{\Gamma}_i)\right). \tag{A4}$$

Note that $\mathbf{E}$ has the same index as the matrix $\mathbf{A}$, as it is a system matrix.

Then, the perturbation matrices for the specifications represented by the LMI (A2) are given by

$$\Delta\mathbf{F1}^{\{i,j\}} = \begin{pmatrix} ((\mathbf{A}_j - \mathbf{A}_i)\mathbf{X})^T + (\mathbf{A}_j - \mathbf{A}_i)\mathbf{X} & \mathbf{E}_j - \mathbf{E}_i & \mathbf{0} \\ \mathbf{E}_j^T - \mathbf{E}_i^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}, \tag{A5}$$

$$\Delta\mathbf{F2}^{\{i,j\}} = -\begin{pmatrix} (\mathbf{B}(\mathbf{\Gamma}_j - \mathbf{\Gamma}_i))^T + \mathbf{B}(\mathbf{\Gamma}_j - \mathbf{\Gamma}_i)) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}. \tag{A6}$$

Acknowledgements

This work has been funded by the Spanish State Research Agency (AEI) and the European Regional Development Fund (ERFD) through the project SCAV (ref. MINECO DPI2017-88403-R).

References

Abbas, H. S., & Werner, H. (2010). Frequency-weighted discrete-time LPV model reduction using structurally balanced truncation. *IEEE Transactions on Control Systems Technology*, *19*(1), 140–147.

Ahmadi, M., & Haeri, M. (2018). Multimodel control of nonlinear systems: An improved gap metric and stability margin-based method. *Journal of Dynamic Systems, Measurement, and Control*, *140*(8), 081013.

Apkarian, P., Gahinet, P., & Becker, G. (1995). Self-scheduled $\mathcal{H}_\infty$ control of linear parameter-varying systems: a design example. *Automatica*, *31*(9), 1251–1261.

Apkarian, P., & Tuan, H. D. (2000). Parameterized LMIs in control theory. *SIAM journal on control and optimization*, *38*(4), 1241–1264.

Blanchini, F., & Miani, S. (2003). Stabilization of LPV systems: state feedback, state estimation, and duality. *SIAM journal on control and optimization*, *42*(1), 76–97.

Bokor, J., & Balas, G. (2004). Detection filter design for LPV systems—a geometric approach. *Automatica*, *40*(3), 511–518.

Chilali, M., & Gahinet, P. (1996). $\mathcal{H}_\infty$ design with pole placement constraints: an LMI approach. *IEEE Transactions on automatic control*, *41*(3), 358–367.

El-Sakkary, A. (1985). The gap metric: Robustness of stabilization of feedback systems. *IEEE Transactions on Automatic Control*, *30*(3), 240–247.

Gőzse, I., Luspay, T., Péni, T., Szabó, Z., & Vanek, B. (2016). Model order reduction of LPV systems based on parameter varying modal decomposition. In *2016 ieee 55th conference on decision and control (cdc)* (pp. 7459–7464).

Hoffmann, C., & Werner, H. (2015). A survey of linear parameter-varying control applications validated by experiments or high-fidelity simulations. *IEEE Transactions on Control Systems Technology*, *23*(2), 416–433.

Ibáñez, B., Inthamoussou, F. A., & De Battista, H. (2019). Wind turbine load analysis of a full range LPV controller. *Renewable Energy*.

Jabali, M. B. A., & Kazemi, M. H. (2017). A new LPV modeling approach using PCA-based parameter set mapping to design a PSS. *Journal of advanced research*, *8*(1), 23–32.

Kwiatkowski, A., Boll, M.-T., & Werner, H. (2006). Automated generation and assessment of affine LPV models. In *Decision and control, 2006 45th ieee conference on* (pp. 6690–6695).

Kwiatkowski, A., & Werner, H. (2008). PCA-based parameter set mappings for LPV models with fewer parameters and less overbounding. *IEEE Transactions on Control Systems Technology*, *16*(4), 781–788.

Lofberg, J. (2004). YALMIP: A toolbox for modeling and optimization in MATLAB. In *Computer aided control systems design, 2004 ieee international symposium on* (pp. 284–289).

López-Estrada, F. R., Ponsart, J.-C., Theilliol, D., Zhang, Y., & Astorga-Zaragoza, C.-M. (2016). LPV model-based tracking control and robust sensor fault diagnosis for a quadrotor UAV. *Journal of Intelligent & Robotic Systems*, *84*(1-4), 163–177.

Marcos, A., & Balas, G. J. (2004). Development of linear-parameter-varying models for aircraft. *Journal of Guidance, Control, and Dynamics*, *27*(2), 218–228.

Matz, J., Mourllion, B., & Birouche, A. (2018). On parametric model order reduction based on projections. In *European control conference (ecc)* (pp. 2977–2982).

Pfifer, H., & Seiler, P. (2015). Robustness analysis of linear parameter varying systems using integral quadratic constraints. *International Journal of Robust and Nonlinear Control*, *25*(15), 2843–2864.

Poussot-Vassal, C., & Roos, C. (2012). Generation of a reduced-order LPV/LFT model from a set of large-scale MIMO LTI flexible aircraft models. *Control Engineering Practice*, *20*(9), 919–930.

Quarteroni, A., Sacco, R., & Saleri, F. (2010). *Numerical mathematics* (Vol. 37). Springer Science & Business Media.

Rizvi, S. Z., Abbasi, F., & Velni, J. M. (2018). Model reduction in linear parameter-varying models using autoencoder neural networks. In *2018 annual american control conference (acc)* (pp. 6415–6420).

Rizvi, S. Z., Mohammadpour, J., Tóth, R., & Meskin, N. (2016). A kernel-based PCA approach to model reduction of linear parameter-varying systems. *IEEE Transactions on Control Systems Technology*, *24*(5), 1883–1891.

Robert, D., Sename, O., & Simon, D. (2009). An $\mathcal{H}_\infty$ LPV Design for Sampling Varying Controllers: Experimentation With a T-Inverted Pendulum. *IEEE Transactions on Control Systems Technology*, *18*(3), 741–749.

Rotondo, D., Cristofaro, A., Johansen, T. A., Nejjari, F., & Puig, V. (2019). Robust fault and icing diagnosis in unmanned aerial vehicles using LPV interval observers. *International Journal of Robust and Nonlinear Control*, *29*(16), 5456–5480.

Rotondo, D., Nejjari, F., & Puig, V. (2013). Quasi-LPV modeling, identification and control of a twin rotor MIMO system. *Control Engineering Practice*, *21*(6), 829–846.

Rotondo, D., Nejjari, F., & Puig, V. (2014). A virtual actuator and sensor approach for fault tolerant control of LPV systems. *Journal of Process Control*, *24*(3), 203–222.

Rotondo, D., Puig, V., Nejjari, F., & Romera, J. (2015). A fault-hiding approach for the switching quasi-LPV fault-tolerant control of a four-wheeled omnidirectional mobile robot. *IEEE Transactions on Industrial Electronics*, *62*(6), 3932–3944.

Rotondo, D., Puig, V., Nejjari, F., & Witczak, M. (2015). Automated generation and comparison of Takagi–Sugeno and polytopic quasi-LPV models. *Fuzzy Sets and Systems*, *277*, 44–64.

Rotondo, D., Sánchez, H. S., Nejjari, F., & Puig, V. (2019). Analysis and design of linear parameter varying systems using LMIs. *Revista Iberoamericana de Automatica e Informatica Industrial*, *16*(1), 1–14.

Sala, A., & Arino, C. (2007). Asymptotically necessary and sufficient conditions for stability and performance in fuzzy control: Applications of Polya's theorem. *Fuzzy Sets and Systems*, *158*(24), 2671–2686.

Sanjuan, A., Rotondo, D., Nejjari, F., & Sarrate, R. (2019). An LMI-based heuristic algorithm for vertex reduction in LPV systems. *International Journal of Applied Mathematics and Computer Science*, *29*(4), 725–737.

Sturm, J. F. (1999). Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization methods and software*, *11*(1-4), 625–653.

Sun, X.-D., & Postlethwaite, I. (1998). Affine LPV modelling and its use in gain-scheduled helicopter control. In *Ukacc international conference on control.*

Tseng, C.-S., Chen, B.-S., & Uang, H.-J. (2001). Fuzzy tracking control design for nonlinear dynamic systems via TS fuzzy model. *IEEE Transactions on fuzzy systems*, *9*(3), 381–392.

Vizer, D., & Mercere, G. (2014). H_∞-based LPV model identification from local experiments with a gap metric-based operating point selection. In *European control conference (ecc)*

(pp. 388–393).
Zribi, A., Chtourou, M., & Djemal, M. (2016). A systematic determination approach of model's base using gap metric for nonlinear systems. *Journal of Dynamic Systems, Measurement, and Control*, *138*(3), 031008.