LPVTools: A Toolbox for Modeling, Analysis, and Synthesis of Parameter Varying Control Systems ⋆

Arnar Hjartarson, **Peter Seiler**, **Andrew Packard** *

* *MUSYN Inc., Minneapolis, MN 55414, USA*
(e-mail: arnar496@gmail.com, peter.j.seiler@gmail.com, andrew.packard60@gmail.com).

Abstract: This paper describes the LPVTools software suite developed by MUSYN Inc. LPVTools is a MATLAB toolbox for simulation, analysis, and design of parameter dependent control systems using the Linear Parameter-Varying (LPV) framework. LPVTools contains data structures to represent both LFT and gridded (Jacobian-linearization) types of LPV systems. In addition it contains a collection of functions and tools for model reduction, analysis, synthesis and simulation of LPV systems. Finally, the toolbox is fully documented and contains several demonstration examples. The software is freely available for use by the community.

Keywords: Linear Parameter-Varying, Software Tools.

1. INTRODUCTION

LPVTools is a MATLAB toolbox for modeling and design of Linear Parameter-Varying (LPV) systems. The toolbox contains data structures to represent LPV systems in both the LFT and gridded (Jacobian-linearization) framework. The core of the toolbox is a collection of functions for model reduction, analysis, synthesis and simulation of LPV systems. Finally, the toolbox contains full documentation and several demonstration examples. LPVTools was developed by MUSYN, Inc. (G. Balas and the authors) but has been made freely available to the community. The toolbox can is available for download at:

`www.aem.umn.edu/~SeilerControl/software.shtml`

LPV theory is closely related to gain scheduling via interpolation of point designs, a standard method used in industry to develop full-envelope flight control laws. LPV theory provides a mathematically rigorous approach to the design of gain-scheduled controllers. This includes powerful guarantees on the performance and robustness in the presence of time-varying dynamics and uncertainty. LPVTools was developed as a result of increased interest in the LPV approach to modeling and design in aerospace applications. Specifically, the tooblox was developed to aid the design of aeroservoelastic control laws for lightweight, flexible Unmanned Aerial Vehicles (UAVs). Current efforts include flight control law development for the Mini-MUTT UAV at the University of Minnesota, Pfifer et al. (2015), and the X-56A by Lockheed Martin and NASA Armstrong Flight Research Center.

This paper describes the LPVTools software suite. It is not meant as a survey of LPV theory and results, but is instead focused on the core data structures in LPVTools and a description of their capabilities. For in an in depth look at LPV theory and recent advances in the field, the reader is referred to recent books and survey papers on the topic, e.g. Briat (2014); Mohammadpour and Scherer (2012); Lovera et al. (2011) The remainder of the paper is divided into five sections. First, an overview of the LPV framework is given in Section 2. Next, the core data structures are introduced in Section 3. The capabilities of LPVTools are detailed in Section 4. Finally, a small example that illustrates the toolbox usage is provided in Section 5. Conclusions are given in Section 6.

The functionality in this initial release of LPVTools does not comprehensively cover the existing results in the LPV literature. In addition, LPVTools does not interact with other existing software for gain-scheduled control design, e.g. the `hinfgs` command in LMILab. Despite these limitations the basic infrastructure in LPVTools should enable development of new algorithms and comparison of various techniques.

2. THE LINEAR PARAMETER-VARYING FRAMEWORK

LPV systems are time-varying, state-space models of the form:

$$\begin{bmatrix} \dot{x}(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} A(\rho(t)) & B(\rho(t)) \\ C(\rho(t)) & D(\rho(t)) \end{bmatrix} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \tag{1}$$

where $\rho \in \mathbb{R}^{n_\rho}$ is a vector of measurable parameters, $y \in \mathbb{R}^{n_y}$ is a vector of measurements, $x \in \mathbb{R}^{n_x}$ is the state vector, $u \in \mathbb{R}^{n_u}$ is a vector of control inputs, and $A : \mathcal{P} \to \mathbb{R}^{n_x \times n_x}$, $B : \mathcal{P} \to \mathbb{R}^{n_x \times n_u}$, $C : \mathcal{P} \to \mathbb{R}^{n_y \times n_x}$ and $D : \mathcal{P} \to \mathbb{R}^{n_y \times n_u}$ are continous matrix valued functions.

The LPV system depends on a set of time-varying parameters ρ. The trajectories of the parameters are assumed to take on values in a known compact set $\mathcal{P} \subseteq \mathbb{R}^{n_\rho}$, and to

⋆ This work was supported by a NASA Small Business Innovation Research contract from NASA Armstrong Flight Research Center, contract #NNX12CA14C. Contract Monitor is Dr. Martin J. Brenner.

have known bounds on their derivatives with respect to time: $\overline{\nu} \leq \dot{\rho} \leq \underline{\nu}$, where $\overline{\nu}$ and $\underline{\nu} \in \mathbb{R}^{n_\rho}$. A trajectory is said to be "rate unbounded" if $\overline{\nu} = \infty$ and $\underline{\nu} = -\infty$.

For control design in the LPV framework, it is further assumed that the trajectory of ρ is not known in advance, and that the parameter values are measured and available in real-time with sensors. The controller produced is itself a LPV system which is optimized for the parameter trajectories in $\rho \in \mathcal{P}$, subject to $\overline{\nu} \leq \dot{\rho} \leq \underline{\nu}$, and dependent on real-time measurements of the parameter.

Several methods have arisen for representing the parameter dependence in LPV models (Equation 1). LPVTools implements data structures for two of these: i) Linearizations on a gridded domain, e.g. as in Becker (1993) or Wu (1995); and ii) Linear Fractional Transformations (LFT), e.g. as in Packard (1994) or Apkarian and Gahinet (1995a,b). Linearizations on a gridded domain are obtained through Jacobian linearization at each grid point. Each linearization approximates the system's dynamics in the vicinity of a particular grid point, and the grid of linearizations captures the system's parameter dependence implicitly. Hence, linearization based LPV models do not require any special dependence on the parameter vector. LFT models, on the other hand, have state matrices that are rational functions of the parameter. Hence, their dependence on the parameter vector is modeled explicitly.

3. THE LPVTOOLS TOOLBOX

LPVTools is implemented in MATLAB, using object-oriented programming. The toolbox introduces several class-based data structures for modeling LPV systems. These data structures extend the functionality associated with standard MATLAB data structures from the Control Systems Toolbox and the Robust Control Toolbox into the LPV framework [1]. This is pictorially represented in Figure 1. The core LPVTools data structures are direct extensions of existing data structures, and provide a parameter dependent interpretation of the original object. Note that LPV systems are time varying, and as such do not have a valid frequency response interpretation. Hence, the parameter dependent frequency response objects (`pfrd` and `upfrd`) are simply a convenience to hold frequency response data at fixed parameter values, and do not imply a time-varying frequency response.

Each LPVTools object implements the capabilities of its corresponding standard object, wherever applicable. The motivation for this approach is to provide a seamless and intuitive interface between existing MATLAB data structures and the new LPVTools data structures. The standard MATLAB data structures become a special case of the LPV data structures, such that if the parameter dependence in a LPV data structure is eliminated it reverts back to a standard Linear Time-Invariant (LTI) MATLAB data structure.

3.1 Grid-based approach

The LPV system in Equation 1 is conceptually represented by a state-space system $S(\rho)$ that depends on a

[1] LPVTools requires MATLAB®, Simulink®, the Control Systems Toolbox®, and the Robust Control Toolbox®.

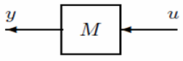
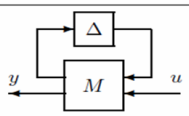
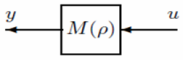
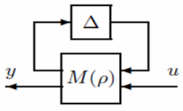
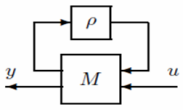
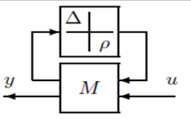
Object Type	Block	Matrix	System	Frequency Response
Nominal		`double`	`ss`	`frd`
Uncertain		`umat`	`uss`	`ufrd`
Nominal Gridded LPV		`pmat`	`pss`	`pfrd`
Uncertain Gridded LPV		`upmat`	`upss`	`upfrd`
Nominal LFT LPV		`pmatlft`	`psslft`	
Uncertain LFT LPV		`pmatlft`	`psslft`	

Fig. 1. Relation between MATLAB objects.

time-varying parameter vector ρ in some compact domain $\mathcal{P} \subset \mathbb{R}^{n_\rho}$. A grid-based LPV model of this system is a collection of linearizations on a gridded domain of parameter values. This is pictorially represented in Figure 2, for an example system that depends on two parameters (in this case Mach number and altitude). For general LPV systems this conceptual representation requires storing the state space system at an infinite number of points in the domain of ρ. The grid based LPV approach, as implemented in LPVTools, approximates this conceptual representation by storing the LPV system as a state space array defined on a finite, gridded domain. For each grid point $\hat{\rho}_k$ there is a corresponding LTI system $(A(\hat{\rho}_k), B(\hat{\rho}_k), C(\hat{\rho}_k), D(\hat{\rho}_k))$ which describes the dynamics of $S(\hat{\rho}_k)$ when $\hat{\rho}_k$ is held constant (note that $\hat{\rho}_k$, represents a constant vector corresponding to the k-th grid point, while ρ_i is later used to denote the i-th element of the vector ρ.) All the linearized systems on the grid have identical inputs u, outputs y and state vectors x. Together they form a LPV system approximation of $S(\rho)$. This approach is motivated by the traditional gain-scheduling framework in aircraft flight control, for which models are typically constructed as linearizations around various flight operating conditions.

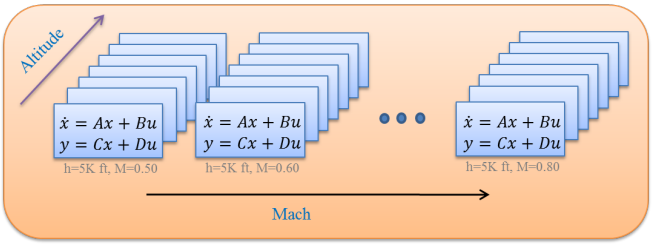


Fig. 2. LPV models defined on a rectangular grid.

The core data structure for grid-based LPV models is the `pss` object (denoting parameter-varying state space

model), which stores the LPV system as a state space array defined on a finite, gridded domain. As a simple example, consider an LPV system $S(\rho)$ that depends on a single scalar parameter ρ in the domain $\rho \in [a, b]$. The infrastructure requires the user to specify the domain with a finite grid, e.g. N points in the interval $[a, b]$. The toolbox contains an `rgrid` data object to facilitate the creation and manipulation of multivariable parameter domains. The user must also specify the values of the state space system $S(\rho)$ at each point in this gridded domain. The `pss` object stores the state-space array data using the MATLAB Control System Toolbox `ss` object. Thus the `pss` can be viewed as the parameter-varying extension of the standard `ss` object. To summarize, the LPV system $S(\rho)$ is represented by a `pss` data object which stores the gridded domain and the array that defines the state-space data at each point in the domain.

The notions of parameter-varying matrices and parameter-varying frequency responses arise naturally to complement the `pss` objects. LPV systems are time-varying and hence frequency responses can not be used to represent the system behavior as parameters vary. However frequency responses are useful to gain intuition about the system performance at fixed locations in the operating domain. LPVTools represents parameter varying matrices and frequency responses by `pmat` and `pfrd` data objects, respectively. These two data objects are both stored as a data array defined on a gridded domain. A `pmat` stores a `double` array, while a `pfrd` stores an array of frequency responses (`frd` object in the Control System Toolbox). Figure 1 shows the relation between the core LPVTools data objects (`pmat`, `pss`, `pfrd`) and existing MATLAB objects. The first row of the table ("Nominal") shows the basic MATLAB objects: matrices are `double` objects, state-space systems are `ss` objects, and frequency responses are `frd` objects. The third row of Table 1 ("Nominal Gridded LPV") shows the corresponding core LPV objects. The main point is that the (`pmat`, `pss`, `pfrd`) objects should be viewed as parameter-varying extensions of the standard MATLAB and Control Systems Toolbox objects (`double`, `ss`, `frd`).

The second row of Table 1 ("Uncertain") shows the equivalent objects used to represent uncertainty: uncertain matrices, state space systems, and frequency responses are represented by `umat`, `uss`, and `ufrd` objects, respectively. These objects are part of the MATLAB Robust Control Toolbox. The Robust Control Toolbox models the uncertainty as a perturbation Δ wrapped in feedback around a nominal part P, i.e. uncertainty is represented using a linear fractional transformation. Real parametric, complex parametric, and unmodeled dynamic uncertainties can be modeled. The fourth row of Table 1 ("Uncertain Gridded LPV") shows the corresponding parameter-varying objects with uncertainty: uncertain parameter-varying matrices, state space systems, and frequency responses are represented by `upmat`, `upss`, and `upfrd` objects, respectively. These objects enable the integration of uncertainty into LPV models.

Jacobian Linearization is the predominant method of constructing grid-based LPV models. In this case the `pss` object is formed by combining a `ss` array of linearizations with an `rgrid` object representing the grid of parameter values (e.g. the grid of Mach and altitude values in Figure 2). An alternative method of constructing grid-based LPV models in LPVTools is to use the `pgrid` atom (denoting a real scalar parameter defined on a grid of points).

3.2 LFT-based approach

An LPV model in Linear Fractional Transformation (LFT) form is an interconnection of a block that represents the plant's nominal dynamics (linear, time invariant), and a block that contains the time-varying parameters on which the system depends.

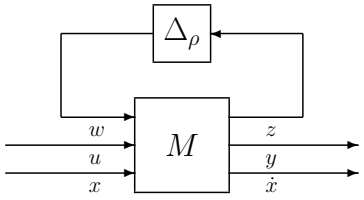


Fig. 3. An LPV system in LFT form.

In the LFT-based approach the LPV system in Equation 1 is expressed as the interconnection of the blocks M and Δ_ρ. $F_u(M, \Delta_\rho)$ denotes the upper LFT shown in Figure 3, where M is a constant matrix such that

$$\begin{bmatrix} z(t) \\ y(t) \\ \dot{x}(t) \end{bmatrix} = M \begin{bmatrix} w(t) \\ u(t) \\ x(t) \end{bmatrix} \tag{2}$$

and $\Delta_\rho := diag(\rho_1(t) I_{r_1}, \ldots, \rho_{n_\rho}(t) I_{r_{n_\rho}})$ is a diagonal matrix such that $w = \Delta_\rho z$. Where I_{r_i} indicates a r_i-by-r_i identity matrix, for positive integers $r_1, \ldots, r_{n_\rho}$, and $\{\rho_i\}_1^{n_\rho}$ represent the elements of ρ. Note that the LFT form can only be used to model LPV systems whose state matrices are rational functions of the parameters (see Cockburn and Morton (1997) for details on LFT modeling).

A key component of the LFT-based LPVTools infrastructure is the core LFT data structure object, referred to as a `tvreal` (denoting a time-varying parameter). The `tvreal` object is used to create a time-varying, real valued scalar object. The `tvreal` has a range, denoting the maximum and minimum value that the time-varying scalar can assume, and a rate-bound denoting the maximum and minimum rate of change of the time-varying scalar. The `tvreal` is used to model individual time-varying parameters, and construct parameter dependent LFT matrices and systems. LPVTools represents LFT-based parameter varying matrices and state-space systems by `plftmat` and `plftss` data objects, respectively. The `plftmat`, and `plftss` objects are constructed using `tvreal` elements, using a syntax that is a direct parallel to the `ureal` syntax that is used to define `umat` and `uss` objects in the Robust Control Toolbox.

Both `plftmat` and `plftss` objects are stored as uncertain objects (i.e. `umat` and `uss`, respectively) which in MATLAB consist of the LFT (M, Δ_ρ), with Δ_ρ being a block of `ureal` objects. Each `plftmat` and `plftss` object's constituent `tvreal` objects provide the additional information necessary to handle the Δ_ρ block like a block of time-varying parameters. Uncertainty can be integrated into the `plftmat`, and `plftss` objects, allowing these data

objects to model systems with, and without uncertainty. The `plftmat` and `plftss` objects should be viewed as LFT-based parameter-varying extensions of the standard MATLAB, Control System Toolbox, and Robust Control Toolbox objects `double`, `ss`, `umat`, and `uss`, as seen in rows five ("Nominal LFT LPV") and six ("Uncertain LFT LPV") in Table 1.

3.3 Transition Between Grid-based and LFT-based LPV Models

Transition between LFT-based (i.e. `plftmat`, or `plftss`) and grid-based (i.e. `pmat`, `pss`, `upmat`, and `upss`) LPV models is possible using the `grid2lft` and `lft2grid` functions in LPVTools. The `lft2grid` function transforms LFT-based LPV models into grid-based LPV models by evaluating the the LFT-based model at a grid of parameter values. The `grid2lft` function transforms a grid-based LPV model into a LFT-based LPV model, by approximating the parameter dependence of the underlying data and expressing it as a rational function of the parameter, which can then be rewritten in LFT form.

3.4 LPVTools Implementation

It is important to note that all LPV objects are being developed within MATLAB's object-oriented programming framework. A benefit of object-oriented programming is that key operations can be overloaded to provide seamless, consistent functionality across a variety of objects. For example, if `A` and `B` are `double` objects then the syntax `A*B` simply multiplies the matrices. If `A` and `B` are `pmat` objects then the syntax `A*B` multiplies the parameter-varying matrices at each grid point in the parameter domain. The manipulation of parameter-varying objects is facilitated by this extension of standard operations for MATLAB objects to meaningful, intuitive operations for LPV objects. In addition, standard MATLAB syntaxes, e.g. `M(i,j)` to index into the (i, j) element of an array, have been overloaded and extended for parameter-varying objects. Object-oriented programming enables this overloading of key functions and this enables a consistent interface between LPVTools and MATLAB, the Control System Toolbox, and the Robust Control Toolbox.

LPVTools provides a suite of LTI synthesis and analysis tools, in addition to the LPV specific functions described in Section 4. Each LPV object overloads functionality from the corresponding standard object in MATLAB, the Control System Toolbox, and the Robust Control Toolbox (see relationship between objects in Table 1). Overloaded methods include `lqr`, `loopsens`, and `loopmargin` from the Control Systems Toolbox, and `hinfsyn`, `wcgain`, and `robuststab` from the Robust Control Toolbox. These functions can be used to study the properties of the LPV system point-by-point in the parameter domain, i.e. the analysis (or synthesis) is performed on the Linear Time Invariant (LTI) system that is obtained when the time-varying parameter ρ is held at a fixed value. The resulting controllers are not LPV controllers (i.e. ones that satisfy the LPV analysis conditions), but a collection of LTI controllers − one for each grid point.

4. LPVTOOLS CAPABILITIES

LPVTools provides a set of tools for modeling, simulation, synthesis and analysis in the LPV framework. These tools include functions for synthesis of output feedback controllers, state-feedback controllers, and estimators. Tools are provided for analysis of the stability and input-to-output gain of LPV systems (with and without uncertainty). Tools are provided for performing model reduction on LPV models. And finally, tools are provided for simulating the time-domain response of an LPV system along user-supplied parameter trajectories.

4.1 Analysis

The LPV system in Equation 1 processes the inputs u linearly, but can depend nonlinearly on the time-varying parameter ρ. The analysis problem is to determine if the system is stable, and to quantify the input-to-output gain of the system. Denote the LPV system in Equation 1 by $S(\rho)$. Analysis in the LPV framework determines if $S(\rho)$ is internally exponentially stable, and whether the input/output map $S(\rho)$ from $u(t)$ to $y(t)$ has certain properties.

LPVTools implements two methodologies for synthesis and analysis in the LPV framework. The two methodologies differ in their formulation of the input/output map $S(\rho)$. The first methodology formulates this input/output map in terms of the induced L_2 norm (gain) of the system:

$$\|S(\rho)\|_{2\to 2} = \max_{\substack{\rho\in\mathcal{P}\\ \underline{\nu}\le\dot{\rho}\le\overline{\nu}}} \max_{\substack{u\in L_2\\ \|u\|_2\neq 0}} \frac{\|S(\rho)u\|_2}{\|u\|_2} \tag{3}$$

The second methodology formulates the input/output map in terms of the stochastic LPV bound on $S(\rho)$:

$$stoch\,(S(\rho)) = \lim_{T\to\infty} \max_{\substack{\rho\in\mathcal{P}\\ \underline{\nu}\le\dot{\rho}\le\overline{\nu}}} E\left\{\frac{1}{T}\int_0^T y^T(t)y(t)dt\right\} \tag{4}$$

which describes the variance of y when the input u is a zero mean, white-noise processes with unit intensity.

The following theorem from Wu (1995) describes the LPV analysis problem when it is formulated in terms of the induced L_2 norm of $S(\rho)$ and the rate-bounds $(\underline{\nu}, \overline{\nu})$ of the parameter are taken into account.

Theorem 1. Given an LPV system $S(\rho)$, described by Equation 1, with $\rho \in \mathcal{P} \subseteq \mathbb{R}^{n_\rho}$ continuously differentiable, $\overline{\nu} \le \dot{\rho} \le \underline{\nu}$, and $x(0) = 0$. If there exists a differentiable matrix function $X : \rho \to \mathbb{R}^{n_x\times n_x}$ such that $X(\rho) = X^T(\rho) > 0$, and

$$\begin{bmatrix} A^TX + XA + \sum_{i=1}^{n_\rho}\left(\underline{\nu}_i\frac{\partial X}{\partial\rho_i}\right) & XB & C^T \\ B^TX & -\gamma I_{n_u} & D^T \\ C & D & -\gamma I_{n_y}\end{bmatrix} < 0 \tag{5}$$

where $\gamma \in \mathbb{R}^+$, and $\underline{\nu}_i$ stands for the fact that the inequality in Equation 5 must hold over the $n_\rho-$dimensional polytope of possible $\dot{\rho}$ values: $[\underline{\nu}_1, \overline{\nu}_1] \times [\underline{\nu}_2, \overline{\nu}_2] \times \ldots \times [\underline{\nu}_{n_\rho}, \overline{\nu}_{n_\rho}]$. Then $S(\rho)$ is internally exponentially stable and $\|S(\rho)\|_{2\to 2} < \gamma$ for all $\rho \in \mathcal{P}$ subject to $\overline{\nu} \le \dot{\rho} \le \underline{\nu}$.

The shorthand notation in Equation 5 represents 2^{n_ρ} inequalities (all the vertices of the convex hull representing

possible $\dot{\rho}$ values) which must hold for all $\rho \in \mathcal{P}$. Theorem 1 implies that the stability of $S(\rho)$ can be determined, and its input-to-output gain quantified, by solving for γ and the matrix function $X(\rho)$ in these $2^{n_\rho}+1$ convex feasibility conditions. The conditions described by Equation 5 are infinite dimensional, since $A(\rho)$, $B(\rho)$, $C(\rho)$, $D(\rho)$ and $X(\rho)$ are all continuous functions of the parameter ρ. LPVTools computes an approximate solution to the infinite dimensional feasibility conditions by converting them into a finite-dimensional set of Linear Matrix Inequalities (LMIs). This is accomplished by the following proceedure:

(1) Grid the set $\mathcal{P}$ into a set of n_r parameter values: $\{\hat{\rho}_1, \hat{\rho}_2, ...\hat{\rho}_{n_r}\}$.
(2) Pick a basis for $X(\rho)$ so that $X(\rho) = \sum_{k=1}^{n_b} f_k(\rho) X_k$, where n_b is the number of basis functions used to construct $X(\rho)$, the scalar functions $f_1, \ldots, f_{n_b} : \rho \to \mathbb{R}$ are the chosen basis functions, and $X_1, \ldots, X_{n_b} \in \mathbb{R}^{n_x \times n_x}$ are constant matrices to be determined.
(3) Solve for γ and $X_1, \ldots, X_{n_b}$, subject to the $(n_r 2^{n_\rho}+1)$ LMIs formed at the grid points by Equation 5, and the condition $X(\rho) > 0$.
(4) Validate the LMI solution on a denser gridding of $\mathcal{P}$.

This analysis approach is a direct extension of the grid-based modeling approach, where the set $\mathcal{P}$ is gridded as a matter of course during the modeling process, such that $S(\rho)$ is evaluated at a discrete set of parameter values.

The LPV analysis problem is different when the rate-bounds are neglected, when its formulated in terms of the stochastic LPV bound, and when the system is modeled as a LFT. The analysis conditions for these cases can be found in the literature, e.g. Wu (1995) and Apkarian and Gahinet (1995a,b). However, the above formulation illustrates the core issues in the LPV analysis problem: Convex constraints, computational complexity grows $O(2^{n_\rho})$, and the necessity for choosing basis functions and representing them in software.

The choice of basis functions is facilitated by the `basis` data structure object in LPVTools. A `basis` object stores the value of a chosen basis function $f_k(\rho)$ at each of the grid points in the domain $\{\hat{\rho}_1, \hat{\rho}_2, ...\hat{\rho}_{n_r}\}$. The `basis` object also stores the values of the partial derivatives of $f_k(\rho)$ (i.e. $\frac{\partial f_k}{\partial \rho_1}, \ldots, \frac{\partial f_k}{\partial \rho_{n_\rho}}$) at each of the grid point.

4.2 Analysis Functions

Analysis in the LPV framework is implemented through two main functions: `lpvnorm` and `lpvwcgain`. `lpvnorm` computes an upper bound on the input-output gain of a LPV system when it has no uncertainty while `lpvwcgain` computes it for an uncertain LPV system. For grid-based LPV objects `lpvnorm` implements both a computation of the induced L_2 norm (as described above) and of the stochastic LPV bound, based on the derivation by Becker (1993) and Wu (1995). In the LFT case, `lpvnorm` implements a computation of the induced L_2 norm only based on the derivation by Apkarian and Gahinet (1995a,b). `lpvwcgain` computes the upper bound on the induced L_2 norm of an uncertain LPV system (grid- or LFT-based). Its implementation is based on Pfifer and Seiler (2014), and Veenman and Scherer (2014).

4.3 Synthesis Functions

The LPV analysis conditions described above are used to derive conditions for controller synthesis. LPVTools implements LPV controller synthesis for both the LFT-based LPV framework and the grid-based LPV framework. The synthesis functions generate controllers which optimize the performance of the closed-loop system while taking into account the permissible parameter trajectories: $\rho \in \mathcal{P}$, subject to $\overline{\nu} \leq \dot{\rho} \leq \underline{\nu}$.

Several LPV synthesis functions are provided. `lpvsyn`, `lpvncfyn`, `lpvmixsyn`, and `lpvloopshape` are used to synthesize LPV output-feedback controllers. `lpvsfsyn` is used to synthesize LPV state-feedback controllers, and `lpvestsyn` is used to generate LPV estimators.

In the grid-based LPV framework these functions can be used to generate controllers and estimators to minimize either the induced L_2 norm (based on results by Becker (1993) and Wu (1995), with pole-constrained synthesis based on the derivation by Lee (1997)) or the stochastic LPV bound (based on results by Wu (1995)). In the LFT-based LPV framework only `lpvsyn` is provided, and it implements an algorithm which minimizes the induced L_2 norm (based on results by Packard (1994), and Apkarian and Gahinet (1995a,b)).

4.4 Modeling and Simulation

The LPV data structures listed in Table 1 provide the means to model LPV systems in MATLAB. LPVTools provides a suite of tools to support the modeling effort. A range of functions is provided to ease the user's interaction with the LPV models (e.g. interpolation, sampling). `connect` and `feedback` from the Control Systems Toolbox, and `sysic` from the Robust Control Toolbox are overloaded to work with the LPV data structures and enable the assembly of arbitrary interconnections of LPV and LTI systems. The function `lpvgram` provides the ability to compute the controllability and observability Gramians of a LPV system. `lpvbalreal` provides the ability to create balanced realizations of LPV models. And the functions `lpvbalancmr`, and `lpvncfmr` enable model reduction of both stable and unstable LPV systems (based on the work of Wood et al. (1996)).

LPVTools includes command line tools to perform both LTI and LPV time-domain simulations. These functions are implemented for both grid-based and LFT-based objects. Simulation functions (e.g. `step`. `impulse`, `initial`, and `lsim`) from the Control Systems Toolbox are overloaded to work with the LPV data structures. They simulate the time-domain response at each grid point, and return a unique simulation response for each grid point. LPV simulations are implemented in `lpvstep`, `lpvimpulse`, `lpvinitial`, and `lpvlsim`. These functions simulate the time-domain response of the LPV system along a user supplied parameter trajectory: $\rho(t)$ for $t \in [t_{initial}, t_{end}]$. LPVTools also includes Simulink blocks for simulation and real-time implementation. These allow the LPVTools data structures to be implemented directly in Simulink.

5. APPLICATION: BENCHMARK ACTIVE CONTROL TECHNOLOGY WING

The model of the NASA Langley Research Center's Benchmark Active Control Technology (BACT) wing will be used to demonstrate the functionality of LPVTools. The BACT model has been used to design and test many types of flutter suppression control strategies including LPV designs, e.g. work in Barker and Balas (2000). In this context flutter denotes a phenomenon where aerodynamic forces acting on the wing excite unstable elastic modes in the wing structure, causing an unstable oscillation. This section briefly describes the modeling, design, and analysis of an LPV active flutter control using the LPVTools software tools. The focus in this section is on demonstrating the functionality and syntax of the LPV software toolbox. Additional details on the design choices can be found in Barker and Balas (2000).

The reduced-order BACT model is defined by state space models on a 4-by-6 grid of Mach and dynamic pressure, $\bar{q}$ (kPa). The reduced order model has a total of six states at each flight condition. Each model has two pair of complex poles which correspond to the plunge and pitch modes of the wing. These complex poles are the only poles near the imaginary axis and the only poles that change significantly with operating condition. As the dynamic pressure increases, one complex pair moves further into the left-half plane, while the other moves towards the right-half plane and has a positive real part above the flutter dynamic pressures, as seen in Figure 5.

The BACT model has two inputs: the control input is the trailing edge flap deflection (rad) and a disturbance input is fed into a Dryden turbulence model. There are two measurements available for control: trailing and leading edge accelerations (cm/sec^2). The BACT model data has been saved in a 4-by-6 array of state space models (an `ss` called `SSArray`) with 2-inputs and 2-outputs. The commands below create a `pss` object for the LPV BACT model.

```
>> size(SSArray)
4x6 array of state-space models.
Each model has 2 outputs, 2 inputs, and 6 states.

% Create rgrid object for parameter domain
>> machvec = [0.5 0.7 0.78 0.82];
>> qvec = [5.99 6.51 7.18 8.38 9.58 10.77];
>> Mach = pgrid('Mach',machvec,[-0.02 0.02]);
>> qbar = pgrid('qbar',qvec,[-0.3 0.3]);
>> Domain = rgrid(Mach,qbar);

% Create pss BACT model
>> BACT = pss(SSArray,Domain)
PSS with 6 States, 2 Outputs, 2 Inputs.
 Mach: Gridded real, 4 points in [0.5,0.82].
 qbar: Gridded real, 6 points in [5.99,10.8].
```

The Mach number and dynamic pressure values are represented by `pgrid` atom objects. The first argment to `pgrid` is the name of the parameter, the second is a vector of grid points, and the third is a vector containing the upper and lower limits on the parameter's rate of variation. In this case $|\dot{M}| \leq 0.02$ and $|\dot{\bar{q}}| \leq 0.03$ kPa/sec where M denotes Mach. The two `pgrid` objects are combined to form the `rgrid` object `Domain`, which describes the two dimensional grid of Mach number and dynamic pressure values. The `rgrid` is combined with the BACT `ss` model array to form a parameter varying state space model (`pss`).

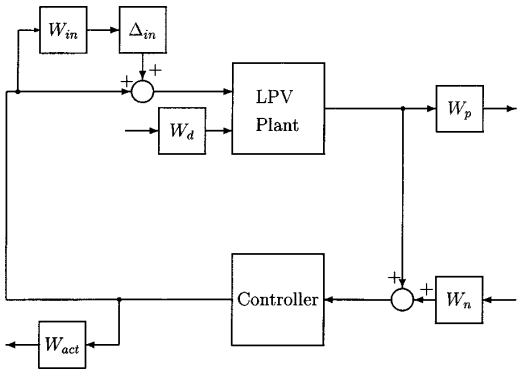


Fig. 4. Generalized Plant for BACT Control Design

Next, the weighted interconnection, shown in Figure 4, is constructed for the LPV design. The weights and weighted interconnection used in the design are taken from Barker and Balas (2000). The weighted interconnection is constructed using the commands below.

```
>> Win   = 0.1*tf([1/2 1],[1/200 1]);
>> Wp    = eye(2)*(1/12500);
>> Wact  = 12/pi;
>> Wn    = [250 0; 0 250];
>> Wd    = 1;

>> systemnames = 'Win Wact Wd G Wp Wn BACT';
>> inputvar  = '[ w; dist; noise{2}; flap_cmd]';
>> outputvar = '[ Win; Wp; Wact; BACT+Wn]';
>> input_to_BACT  = '[G+w; Wd]';
>> input_to_Win   = '[G]';
>> input_to_Wact  = '[G]';
>> input_to_Wd    = '[dist]';
>> input_to_G     = '[ flap_cmd ]';
>> input_to_Wp    = '[BACT]';
>> input_to_Wn    = '[ noise ]';
>> H=sysic;
```

Notice that the standard `sysic` interconnection command has been overloaded to allow interconnections of `pss` objects and standard MATLAB system objects. `H` is a `pss` object that specifies the weighted design interconnection, i.e. the generalized plant, over the (Mach,$\bar{q}$) domain.

Next, a LPV controller was synthesized under the assumption that the parameter rate of variation is bounded. This control synthesis is performed using `lpvsyn` which solves a set of parameterized LMIs using parameter varying Lyapunov functions. The Lyapunov functions are assumed to be parameter dependent with basis functions: $f_1 = 1$, $f_2 =$ Mach, and $f_3 = \bar{q}$. The following commands define the basis functions:

```
>> f1 = basis(1,0);
>> f2 = basis(Mach,1);
>> f3 = basis(qbar,1);
>> Xb = [f1;f2;f3]; Yb=Xb;
```

The first argument to `basis` specifies the value of the basis function, the second specifies the partial derivative with respect to the parameter. The controller is synthesized using `lpvsyn`:

```
>> [Krb,Gamma,Info] = lpvsyn(H,2,1,Xb,Yb);
```

The first input to `lpvsyn` is the `pss` representing the weighted generalized plant, the second and third are the number of measurmentes and control inputs available to the controller, and the fourth and fifth are the basis functions to be used in the Lyapunov functions.

The software tools can be used to easily analyze and compare the open and closed loop systems. As mentioned above, the open-loop BACT wing becomes unstable above the flutter dynamic pressure. Figure 5 shows a plot of two of the open loop complex poles across all (Mach,$\bar{q}$) conditions. The figure also shows the closed-loop poles that appear in the same region of the complex plane. Notice that the controller stabilizes the poles at all (Mach,$\bar{q}$) conditions. The plot in Figure 5 is generated using the following commands:

```
>> CL = lft(Krb,BACT);
>> Pol = pole(BACT);
>> Pcl = pole(CL);
>> plot(real(Pol),imag(Pol),'bx')
>> plot(real(Pcl),imag(Pcl),'rs')
>> axis([-10 2 18 30])
```

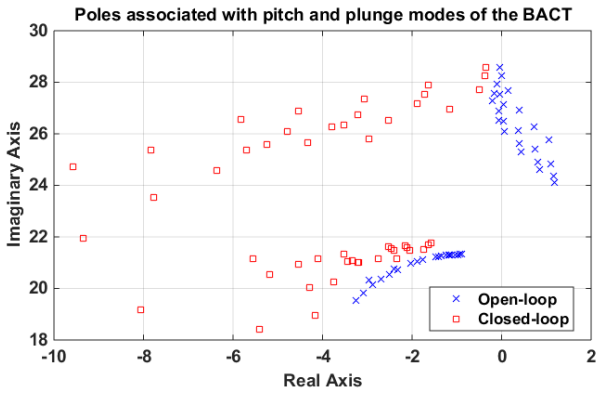


Fig. 5. Open loop (blue x) and closed loop (red square) poles associated with the plunge and pitch modes of the wing. Values for all (Mach,$\bar{q}$) values shown.

6. CONCLUSION

This paper describes LPVTools, a software suite for modeling, simulation, analysis and synthesis of LPV systems. A brief description of the LPV framework was provided, with a key LPV analysis result presented to illustrate the framework's potential. The paper then provides an overview of the core LPVTools data structures for modeling LPV systems, and lists the various capabilities of LPVTools. The toolbox has been made freely available to the community. It is hoped that this toolbox will encourage further advances in LPV systems.

ACKNOWLEDGEMENTS

This paper is dedicated to Dr. Gary Balas, president of MUSYN Inc., who passed away in November 2014. Dr. Balas was a strong advocate of Linear Parameter-Varying Control. He believed that reliable and easy-to-use software for LPV modeling, analysis and design would encourage wider use of this approach. Dr. Balas instigated and led the effort to develop LPVTools. The open-source release of LPVTools realizes his ambition of making these powerful tools available to the practicing engineer.

REFERENCES

Apkarian, P. and Gahinet, P. (1995a). A convex characterization of gain-scheduled H_∞ controllers. *IEEE Transactions on Automatic Control*, 40(5).

Apkarian, P. and Gahinet, P. (1995b). Erratum to "A convex characterization of gain-scheduled H_∞ controllers". *IEEE Transactions on Automatic Control*, 40(9).

Barker, J. and Balas, G. (2000). Comparing Linear Parameter-Varying Gain-Scheduled Control Techniques for Active Flutter Suppression. *Journal of Guidance, Control, and Dynamics*, 23(5).

Becker, G. (1993). *Quadratic Stability and Performance of Linear Parameter Dependent Systems*. Ph.D. Dissertation, University of California, Berkeley.

Briat, C. (2014). *LPV & Time-Delay Systems – Analysis, Observation, Filtering & Control*, volume 3. Springer-Heidelberg, Germany.

Cockburn, J.C. and Morton, B.G. (1997). Linear Fractional Representations of Uncertain Systems. *Automatica*, 33(7).

Lee, L.H. (1997). *Identification and Robust Control of Linear Parameter-Varying Systems*. Ph.D. Dissertation, University of California at Berkeley.

Lovera, M., Novara, C., dos Santos, P.L., and Rivera, D. (2011). Guest Editorial Special Issue on Applied LPV Modeling and Identification. *IEEE Transactions on Control Systems Technology*, 19(1), 1–4.

Mohammadpour, J. and Scherer, C.W. (eds.) (2012). *Control of Linear Parameter Varying Systems with Applications*. Springer US, Boston, MA.

Packard, A. (1994). Gain scheduling via linear fractional transformations. *Systems and Control Letters*, 22(2), 79–92.

Pfifer, H., Moreno, C.P., Theis, J., Kotikapuldi, A., Gupta, A., Takarics, B., and Seiler, P. (2015). Linear Parameter Varying Techniques Applied to Aeroservoelastic Aircraft: In Memory of Gary Balas. In *Submitted to the 1st IFAC Workshop on Linear Parameter Varying systems*.

Pfifer, H. and Seiler, P. (2014). Robustness analysis of linear parameter varying systems using integral quadratic constraints. *International Journal of Robust and Nonlinear Control*.

Veenman, J. and Scherer, C.W. (2014). Iqc-synthesis with general dynamic multipliers. *International Journal of Robust and Nonlinear Control*, 24(17), 3027–3056.

Wood, G.D., Goddard, P.J., and Glover, K. (1996). Approximation of Linear Parameter-Varying Systems. In *IEEE Conference on Decision and Control*, volume 1, 406–411.

Wu, F. (1995). *Control of linear parameter varying systems*. Ph.D. Dissertation, University of California at Berkeley.