ENHANCED LFR-TOOLBOX FOR MATLAB AND LFT-BASED GAIN SCHEDULING

S. Hecker*, A. Varga*, J.F. Magni◇

† German Aerospace Center (DLR), Institute of Robotics and Mechatronics
D-82230 Wessling, Germany

◇ Office National d'Études et de Recherches Aérospatiales (ONERA)
2, av. Edouard Belin, 31055 Toulouse, France

Key words: linear fractional transformation(LFT), scheduled control, MATLAB toolbox

Abstract. *We describe recent developments and enhancements of the LFR-Toolbox for MATLAB for building LFT-based uncertainty models and for LFT-based gain scheduling. A major development is the new LFT-object definition supporting a large class of uncertainty descriptions: continuous- and discrete-time uncertain models, regular and singular parametric expressions, more general uncertainty blocks (nonlinear, time-varying, etc.). By associating names to uncertainty blocks the reusability of generated LFT-models and the user friendliness of manipulation of LFR-descriptions have been highly increased. Significant enhancements of the computational efficiency and of numerical accuracy have been achieved by employing efficient and numerically robust Fortran implementations of order reduction tools via mex-function interfaces. The new enhancements in conjunction with improved symbolical preprocessing lead generally to a faster generation of LFT-models with significantly lower orders. Scheduled gains can be viewed as LFT-objects. Two techniques for designing such gains are presented. Analysis tools are also considered.*

1 INTRODUCTION

In modelling uncertainties in linear systems the *linear fractional transformation* (LFT) plays an important role. LFT-based representations (see Figure 1) are ready to be used in robust control applications like the structured singular value (also called μ) [17].

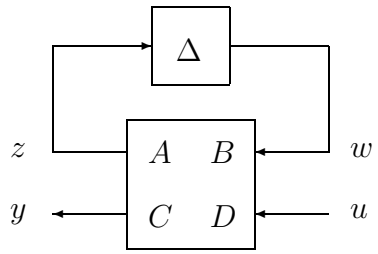

Figure 1: LFT-Representation

The LFT system equations are

$$\begin{aligned} z &= Aw + Bu \\ y &= Cw + Du \\ w &= \Delta z. \end{aligned} \tag{1}$$

For the partitioned matrix

$$M = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \in \mathbf{R}^{(p_1+p_2)\times(m_1+m_2)}$$

and $\Delta \in \mathbf{R}^{m_1 \times p_1}$, the *upper LFT* is defined as

$$\mathcal{F}_u(M, \Delta) = D + C\Delta(I - A\Delta)^{-1}B, \tag{2}$$

which represents the input/output mapping between u and y.

The LFR (Linear Fractional Representation) Toolbox is a MATLAB toolbox for the realization of LFT-representations for uncertain system models. With this toolbox LFT-representations can be directly obtained from symbolic expressions or via object oriented manipulation of LFT-objects (addition, multiplication, inversion,

column/row concatenation) [9]. The Version 1 of the LFR-toolbox has been implemented by the third author [10] and supports uncertain real or complex diagonal (scalar) blocks and uncertain rectangular real or complex full blocks. Elementary LFT-objects are realized with the function `lfrs` and the different blocks in the feedback matrix Δ are distinguished by the order of the initial realization.

The main goal of LFT-based uncertainty modelling is the generation of low order LFT-representations. The order of an LFT-representation is m_1, the row dimension of the $m_1 \times p_1$ block-diagonal matrix Δ. The generation of low order LFT-realizations is supported by the LFR toolbox in various ways. Version 1 of the LFR-toolbox offers special functions for symbolic preprocessing techniques as Mortons method [13] for affine uncertainty representations and the tree decomposition [4] for polynomial matrices to manipulate the symbolic system equations yielding LFT representations of low order. Furthermore numerical multidimensional order reduction and approximation methods [5, 9] for LFT-models are available. These algorithms use the MATLAB based functions `minreal` and `ctrbf` for order reduction.

In this paper we present recent developments and enhancements of the LFR-toolbox that are focused to improve the capabilities for low order LFT-modelling. With the definition of a new LFT-object, supporting also constant blocks in Δ [7], we circumvent the problem, that for the object oriented LFT-realization approach rational expressions like $1/p$ had to be symbolically normalized before performing the LFT-realization. This improvement generally leads to LFT-representations of lower orders. Furthermore, the new LFT-object definition is more transparent, user friendly and supports additional types of uncertainties to be directly compatible to other MATLAB toolboxes like the μ-Analysis and Synthesis toolbox, the LMI toolbox and the Robust Control toolbox. Significant enhancements of the computational efficiency and of numerical accuracy have been achieved by employing efficient and numerically robust FORTRAN implementations of order reduction tools via mex-function interfaces. The new enhancements in conjunction with improved symbolical preprocessing lead generally to a faster generation of LFT-models with significantly lower orders.

In parallel to the developments of Version 2 of the LFR-toolbox, Version 1 was improved by introducing control analysis and synthesis capabilities.

Modelling systems in LFT form is mostly useful when it is intended to use μ-analysis. But, if μ-analysis is considered for closed-loop properties it becomes necessary to assume that the controllers are independent of parameter variations or that they are frozen around selected operating points. This limitation is very troublesome especially in aeronautics where controllers are usually scheduled. In fact, scheduled gains can be viewed as LFTs. This remark justifies the development of new tools for designing feedback gain directly in LFT form. With systems and feedback gains in LFT form it was also natural to adapt classical analysis tools and to develop new specific analysis tools. The new tools for synthesis and analysis are briefly presented respectively in §6 and §7[1].

2 NEW DEVELOPMENTS

The current version of the LFR-Toolbox, Version 2, relies on a new LFT-object definition. The core function `lfr` to create an LFT-object is called inside almost all functions of the toolbox. This function has five input arguments: the first four input arguments specify the matrices A, B, C, D (see Figure 1) and the fifth argument describes the structure of Δ. The structure description argument is a structure with two fields: `names` and `desc`. For a given LFT-object `L` the five arguments can be recovered respectively by: `L.a`, `L.b`, `L.c`, `L.d`, `L.blk`.

As an example, the fields `names` and `desc` of the structure description argument of an LFT-object with $\Delta = \mathrm{diag}(p_1 I_2, p_2)$ are given by,

$$\texttt{names} = \{\, \texttt{p1name}, \texttt{p2name} \,\}$$

$$\texttt{desc} = \begin{bmatrix} 2 & 1 \\ 2 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ \vdots & \vdots \end{bmatrix} \begin{array}{ll} \leftarrow & \text{row-dimension} \\ \leftarrow & \text{column-dimension} \\ \leftarrow & \text{real(1)/complex(0)} \\ \leftarrow & \text{scalar(1)/full(0)} \\ \leftarrow & \text{linear(1)/nonlinear(0)} \\ \leftarrow & \text{time-invariant(1)/-varying(0)} \\ \leftarrow & \text{bound-information} \end{array}$$

where the field `names` is a cell-array of strings containing the names of all blocks in Δ. The names '$1/s$' and '$1/z$' are reserved for the integrator block I/s (continuous-time systems) and the delay block I/z (discrete-time systems), respectively. These blocks are included in Δ with the sole usage to represent standard linear time-invariant systems (continuous- or discrete-time) as an LFT-object. Furthermore the name '1' is reserved for a constant identity matrix block in Δ. This block plays a major role in representing singular parametric

[1] Illustrative examples: `http://www.cert.fr/dcsd/idco/perso/Magni/example2.html`

expressions as standard LFTs. An internal LFT-object reordering (function `reorderlfr`) is performed after each LFT-object manipulation where the constant block (if exists) is put on the first diagonal position of Δ, the integrator/delay block (if exists) is put on the second diagonal position followed by all the uncertainty blocks in ASCII dictionary order. Each block in Δ is clearly identified by its name, which makes the manipulation of LFT-objects flexible and transparent. For example, additional uncertainties can be added in any step of the LFT-realization and the names can be modified (e.g., by using the function `set`).

For each name in the field `names` there exists a corresponding column in the field `desc`, which describes the row/column dimensions and properties of this block. The LFT-object supports real or complex diagonal (scalar) blocks and real or complex full (rectangular) blocks. These blocks can have the properties linear/nonlinear and time-invariant/time-varying (in the case of nonlinear uncertainties the property time-invariant means memoryless). Furthermore, the field `desc` includes bound information for each uncertainty block, which can be described by min/max-values, a sector bound (for nonlinear uncertainties) or a SISO frequency dependent bound.

Using a standard LFT-representation it is not possible to directly represent parametric expressions like $1/p$ as an LFT-object. For such expressions a symbolic normalization of the parameters usually has to be performed before the LFT-realization. However, symbolic normalization tends to increase the order of the generated LFT-representations (see [4, 11] for examples). One way to avoid preliminary normalization is to use a general descriptor type LFT-representation as presented in [7]. In the Version 2 of the LFR-toolbox we support a so-called *generalized LFT-representation* which uses a constant identity matrix as the first diagonal block of Δ. With this simple extension it is possible to represent arbitrary rational parametric expressions as LFT models. The constant block in Δ can be considered as an additional dimension in a multidimensional system representation [3] and standard multidimensional order reduction methods [5, 9] can be applied to the generalized LFT-representation.

The flexibility offered by using the generalized LFT-representation can be easily illustrated when performing LFT-manipulations involving system inversions (e.g., using functions like `mrdivide`, `rf2lfr` and `lf2lfr`). As an example, consider the LFT realization of a compound parametric matrix

$$[N(\Delta)\ D(\Delta)] = \mathcal{F}_u(\left[\begin{array}{c|cc} A & B_N & B_D \\ \hline C & D_N & D_D \end{array}\right], \Delta),$$

where $D(\Delta)$ is $p \times p$ and invertible. The function `lf2lfr` calculates an LFT-representation (M_{lf}, Δ_{lf}) such that

$$D^{-1}(\Delta)N(\Delta) = \mathcal{F}_u(M_{lf}, \Delta_{lf})$$

with $\Delta_{lf} = \mathrm{diag}(I_p, \Delta)$ and

$$M_{lf} = \left[\begin{array}{cc|c} D_D + I_p & C & D_N \\ B_D & A & B_N \\ \hline -I_p & 0 & 0 \end{array}\right].$$

By employing a constant block of order p, we can thus avoid the explicit inversion of D_D, and more importantly, we can represent the result even in the case when this matrix is not invertible.

To realize an LFT-representation for a rational parametric matrix, the toolbox supports the object oriented LFT-realization procedure, which was suggested in [9] and extended in [7] for descriptor-type LFT representations. This method is based on elementary LFT manipulations like addition/subtraction, multiplication, inversion, row/column concatenation. Furthermore conversions to LFT-objects of LTI-objects from the Control Toolbox, PCK-system representations from the μ-Synthesis Toolbox as well as constant matrices, are automatically performed via the core function `lfr`.

3 NORMALIZATION

To obtain finally a standard LFT-representation (without constant block in Δ) ready to be used in robust control applications (e.g. μ-Analysis/Synthesis) a normalization of parameters must usually be performed. The main advantage of using the generalized LFT-representation is that the normalization can be performed as the last step in the LFT-modelling. Thus there is no need for a preliminary symbolic normalization, which generally tends to increase the order of the resulting LFT-representation.

Let $\mathcal{F}_u(M, \Delta)$ be an LFT-representation with Δ having the structure

$$\Delta = \mathrm{diag}(I_c, I_d/s, \delta_1 I_{r1}, ..., \delta_k I_{rk}, \hat{\Delta}), \tag{3}$$

where I_c is a $c \times c$ identity matrix, I_d/s is a $d \times d$ integrator block (I_d/z for discrete time systems), followed by k real parametric uncertainty blocks and the block-diagonal $e_1 \times e_2$ matrix $\hat{\Delta}$, which consists of all uncertainty blocks that are not real parametric. By normalization each uncertain real parameter $\delta_i \in [\delta_{i,min}, \delta_{i,max}]$ is

replaced by $\delta_{i,n} + \delta_{i,sl}\overline{\delta}_i$, with $\delta_{i,n} := (\delta_{i,min} + \delta_{i,max})/2$ and $\delta_{i,sl} := (\delta_{i,max} - \delta_{i,min})/2$ such that $|\overline{\delta}_i| \leq 1$, for $i = 1, ..., k$. Thus, the normalization amounts to replace Δ by $\Delta_n + \Delta_{sl}\overline{\Delta}$ in $\mathcal{F}_u(M, \Delta)$, where

$$\Delta_n = \mathrm{diag}(I_c, 0_d, \delta_{1,n} I_{r1}, ..., \delta_{k,n} I_{rk}, 0_{e_1 \times e_2}) \quad (4)$$

$$\Delta_{sl} = \mathrm{diag}(0, I_d, \delta_{1,sl} I_{r1}, ..., \delta_{k,sl} I_{rk}, I_{e_1}) \quad (5)$$

$$\overline{\Delta} = \mathrm{diag}(0_c, I_d/s, \overline{\delta}_1 I_{r1}, ..., \overline{\delta}_k I_{rk}, \hat{\Delta}), \quad (6)$$

The following result shows that by normalization the constant block of a generalized LFT-representation can be eliminated.

Lemma 3.1 *Consider $\mathcal{F}_u(M, \Delta)$ with*

$$M = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

and Δ as given in (3). If Δ_n, Δ_{sl}, $\overline{\Delta}$ have the forms as in (4), (5) and (6) respectively and if $(I - A\Delta_n)$ is invertible, then

$$\mathcal{F}_u(M, \Delta) = \mathcal{F}_u(\overline{M}, \overline{\Delta}) = \mathcal{F}_u(\widetilde{M}, \widetilde{\Delta}),$$

where

$$\overline{M} = \left[\begin{array}{c|c} \overline{A} & \overline{B} \\ \hline \overline{C} & \overline{D} \end{array}\right] = \left[\begin{array}{cc|c} \overline{A}_{11} & \overline{A}_{12} & \overline{B}_1 \\ \overline{A}_{21} & \overline{A}_{22} & \overline{B}_2 \\ \hline \overline{C}_1 & \overline{C}_2 & \overline{D} \end{array}\right]$$

with

$$\begin{aligned} \overline{A} &= (I - A\Delta_n)^{-1} A\Delta_{sl} \\ \overline{B} &= (I - A\Delta_n)^{-1} B \\ \overline{C} &= C(\Delta_n (I - A\Delta_n)^{-1} A + I)\Delta_{sl} \\ \overline{D} &= C\Delta_n (I - A\Delta_n)^{-1} B + D, \end{aligned}$$

and

$$\widetilde{M} = \left[\begin{array}{c|c} \overline{A}_{22} & \overline{B}_2 \\ \hline \overline{C}_2 & \overline{D} \end{array}\right]$$

$$\widetilde{\Delta} = diag(I_d/s, \overline{\delta}_1 I_{r1}, ..., \overline{\delta}_k I_{rk}, \hat{\Delta}).$$

Proof. The calculation of $\overline{M}$ is straightforward and due to the particular structure of Δ_{sl} the submatrices $\overline{A}_{11}$, $\overline{A}_{21}$ and $\overline{C}_1$ of $\overline{M}$ are null.

To perform normalization, the LFR-toolbox offers the function `normalizelfr`, that allows to perform the normalization for a single parameter or for a selected set of parameters.

4 ENHANCED ORDER REDUCTION

LFT-models generated using an object oriented approach tend to be of considerable size (e.g., several hundreds) even for relatively simple practical applications (see Example 1). The high computational efforts resulted due to these large orders often prevent the applicability of available standard software tools for robust control design (e.g., convex optimization based approaches). Fortunately, these LFT-models are almost always non-minimal, and therefore using appropriate numerical tools to perform *exact* order reduction of LFT-models can alleviate the situation by producing models of lower order which allow the applicability of robust control methods like μ-Synthesis/Analysis.

Efficient and numerically reliable tools for order reduction of LFT-models are of primary importance to ease the usability of such models. To achieve efficiency of computation, numerical robustness and a high accuracy of results, the toolbox relies on Fortran based robust implementations of algorithms for basic computations related to order reduction. A language like Fortran allows to easily exploit all structural features of a computational problems with low additional computational effort and minimum memory usage. Fortran routines can be easily executed within the user friendly environment MATLAB via external functions, the so called *mex*-functions. Several *mex*-functions based on powerful Fortran routines from the LAPACK-based [1] public domain control library SLICOT [2] form the order reduction computational kernel of the LFR-Toolbox.

The LFR-Toolbox provides several order reduction tools for exact or approximative reduction of order. The *exact* 1-d order reduction technique [9] can be performed using the function `minlfr1` which is based on the

efficient ($O(n^3)$ complexity) SLICOT-based *mex*-function `ssminr` for the calculation of minimal realizations. Note that a pure MATLAB-based implementation using the MATLAB Control Toolbox function `minreal` would have a $O(n^4)$ worst-case complexity.

The *approximative* 1-d order reduction [15] can be performed using `redlfr1`, which is based on the collection of model reduction tools available in SLICOT [14], covering the balanced truncation, singular perturbation approximation and Hankel-norm approximation approaches. All these methods are implemented in a single *mex*-function `sysred` which is called by `redlfr1` to reduce 1-d (discrete-time) systems. With an appropriate scaling of the A matrix of the LFT-model (see Figure 1), this function can be also employed to perform *exact* order reduction.

The function `minlfr` can be used for n-d order reduction [5]. This function has been completely reimplemented to improve efficiency. The calculation of the n-d controllability/observability staircase forms relies on the $O(n^3)$ complexity SLICOT-based *mex*-function `sscof` to compute controllability/observability staircase forms using orthogonal transformations. Note that a pure MATLAB-based implementation using the MATLAB Control Toolbox function `ctrbf` would have a $O(n^4)$ worst-case complexity.

The SLICOT-based *mex*-function `balsys` is systematically called in all order reduction functions to perform a system scaling of the LFT-models as a preliminary operation within the order reduction routines. As the LFT-models resulting from the object oriented realization approach [9] can have matrices with a wide range of values this operation is essential before computing numerical sensitive controllability staircase forms.

The order reduction functions can be applied manually at any stage of the LFT-realization or can be executed automatically after each object oriented LFT-manipulation (e.g., multiplication, addition, etc.). To set global options (e.g., to perform or not automatic order reduction), the function `lfropt` can be used. This function basically defines a set of global variables to control the order reduction and to set the associated tolerances.

5 SYMBOLIC PREPROCESSING

The role of symbolic preprocessing of multivariate rational matrices is to convert individual elements, entire rows/columns or even the whole symbolic matrix to special decomposed forms which allow to immediately obtain a low order LFT-representation. Symbolic preprocessing oriented towards generating low order LFT-representations has been considered previously [4, 13, 15]. For the realization of a single multivariate rational function the Horner evaluation scheme and the "optimal operation count" based evaluation schemes have been employed in [15] as basis to generate lower order LFT-realizations. Alternatively, conversions to partial fraction form or continuous fraction form may be very efficient to obtain low order LFT-realizations. One of the most promising new techniques is the *variable splitting* (VS) based factorization technique which allows to express any scalar polynomial as an inner product of two vectors, each of them containing a disjoint set of parameters as indeterminates. The factors can be then efficiently realized using matrix oriented preprocessing (see next paragraph) to obtain low order realizations.

An efficient technique applicable to multivariate polynomial matrices is the *tree-decomposition* (TD) based approach proposed in [4]. This approach can be also employed to rational matrices represented in polynomial fractional forms. The LFR-toolbox includes an enhanced implementation of the TD technique, called ETD, which directly applies to rational matrices where the elements are polynomials in integer powers (positive or negative) of the indeterminates. Additionally, further enhancements were obtained by integration of Morton's method [13] in the ETD algorithm and by extending the VS method to rows or columns of matrices.

All these methods for decomposition of multivariate rational functions and matrices are supported by the function `sym2lfr` of the toolbox. To increase the efficiency of the symbolic preprocessing, many of the core functions are directly implemented in MAPLE and called via the Extended Symbolic Toolbox of MATLAB.

6 LFT APPROACHES TO GAIN SCHEDULING

In the previous sections, this paper considers input/output LFT representation of dynamic systems. In this section we shall use the LFT form of state-space representations. First, it is shown how to obtain state-space models in LFT form from input/output representations. Then, two design techniques are briefly presented.

6.1 LFT-state-space models

The representation of Figure 1 or Equ. (2) is an input/output representation. For introducing the corresponding LFT-state-space model the matrix Δ of (3) is written

$$\Delta = \mathrm{diag}(I_d/s, \delta_1 I_{r1}, ..., \delta_k I_{rk}) = \mathrm{diag}(I_d/s, \Delta')$$

here we have removed the constant and full blocks. More precisely, in Δ' remain only the blocks corresponding to *real parameters that can be measured*, so, that can be used for gain scheduling. Partitioning the matrices A,

B, C of (2) in conformity with the above matrix Δ we obtain

$$y = \mathcal{F}_u \left(\left[\begin{array}{cc|c} A_{11} & A_{12} & B_1 \\ A_{21} & A_{22} & B_2 \\ \hline C_1 & C_2 & D \end{array} \right], \left[\begin{array}{cc} I_d/s & 0 \\ 0 & \Delta' \end{array} \right] \right) u \tag{7}$$

The state-space representation we look for is the transfer from

$$\left[\begin{array}{c} x \\ u \end{array} \right] \text{ to } \left[\begin{array}{c} \dot{x} \\ y \end{array} \right]$$

which is easily identified from (7) as

$$\left[\begin{array}{c} \dot{x} \\ y \end{array} \right] = \mathcal{F}_u \left(\left[\begin{array}{cc|c} A_{22} & A_{21} & B_2 \\ A_{12} & A_{11} & B_1 \\ \hline C_2 & C_1 & D \end{array} \right], \Delta' \right) \left[\begin{array}{c} x \\ u \end{array} \right] \tag{8}$$

This last equation is the LFT form of the state-space representation we look for. For simplifying notations we shall denote this representation as $(A_\Delta, B_\Delta, C_\Delta, D_\Delta)$ where for example

$$A_\Delta = \left[\begin{array}{cc} I & 0 \end{array} \right] \mathcal{F}_u \left(\left[\begin{array}{cc|c} A_{22} & A_{21} & B_2 \\ A_{12} & A_{11} & B_1 \\ \hline C_2 & C_1 & D \end{array} \right], \Delta' \right) \left[\begin{array}{c} I \\ 0 \end{array} \right]$$

similar for B_Δ, C_Δ and D_Δ. Two techniques for designing scheduled gains are now briefly presented..

6.2 First gain scheduling technique

This technique was introduced in [12] and illustrated in [6]. It consists of *transforming the problem of designing a scheduled law into a problem of robust control design.* First, a polynomial form of the scheduled gain must be chosen. Then the system (in LFT form) is augmented (function `lfr2bsys`). A robust control law is designed (using any design technique, for example multi-objective optimization [8]) relative to the augmented system. Finally, the augmented gain is transformed to an LFT gain corresponding to the original system (function `bfb2klfr`). The following example explains the underlying ideas.

Example of a polynomial form of the scheduled gain:

$$K_\Delta = K_0 + \delta_1 K_1 + \delta_1 \delta_2 K_2 \tag{9}$$

The augmented system corresponding to $(A_\Delta, B_\Delta, C_\Delta, D_\Delta)$ is

$$\left(A_\Delta, B_\Delta, \left[\begin{array}{c} C_\Delta \\ \delta_1 C_\Delta \\ \delta_1 \delta_2 C_\Delta \end{array} \right], \left[\begin{array}{c} D_\Delta \\ \delta_1 D_\Delta \\ \delta_1 \delta_2 D_\Delta \end{array} \right] \right) \tag{10}$$

Justification: Consider a feedback $u = K_\Delta y$

$$u = K_\Delta (C_\Delta x + D_\Delta u) =$$

$$[K_0 \;\; K_1 \;\; K_2] \left(\left[\begin{array}{c} C_\Delta \\ \delta_1 C_\Delta \\ \delta_1 \delta_2 C_\Delta \end{array} \right] x + \left[\begin{array}{c} D_\Delta \\ \delta_1 D_\Delta \\ \delta_1 \delta_2 D_\Delta \end{array} \right] u \right)$$

so, it is equivalent to apply K_Δ to $(A_\Delta, B_\Delta, C_\Delta, D_\Delta)$ or to apply $[K_0 \;\; K_1 \;\; K_2]$ to the augmented system (10). Designing $[K_0 \;\; K_1 \;\; K_2]$ is a robust control problem.

This technique can be applied without the LFR Toolbox (as in [6]), however this toolbox offers some convenient features:

- for modelling the original system (8) in LFT form,
- for building the augmented system (10),
- for back transformation to the LFT/scheduled gain (9) and order reduction,
- for μ-analysis of the results because the system and its controller are in LFT form,
- the toolbox analysis tools for gridded LFR-objects presented in §7 can also be used.

6.3 Second gain scheduling technique

The second technique that is implemented is based on a very simple idea. Assuming that an algorithm $\mathcal{K}$ used for computing a feedback gain from a fixed state-space representation

$$(A, B, C, D) \rightarrow K = \mathcal{K}(A, B, C, D)$$

only involves matrix operations that exist also for LFT-objects, the same algorithm can be used to design directly a scheduled gain (in LFT form):

$$(A_\Delta, B_\Delta, C_\Delta, D_\Delta) \rightarrow K(\Delta) = \mathcal{K}(A_\Delta, B_\Delta, C_\Delta, D_\Delta)$$

Unfortunately, there are not so many standard feedback design algorithms that satisfy the above requirement. For example all techniques involving eigenvalue / eigenvector computation (the eigenvalues of an LFT object are not an LFT objects) or involving optimization (*e.g.* LMI's) cannot be considered. The eigenvector assignment technique satisfies the above requirement.

The eigenvector assignment approach to LFT-scheduled gain design has been validated in an industrial setting considering a military aircraft in all the subsonic flight domain. Theory and toy examples can be found in [11]. It is shown in this reference that for aircraft control, the scheduling parameters can be chosen as being the aerodynamic coefficients in addition to the speed, Mach number and so on, so, a generic aircraft controller can be derived. In addition, the poles that are assigned can also be viewed as parameters entering in the Δ matrix, which means that the performance of such LFT controllers can be tuned without any re-design.

However, the size of the LFT gain Δ matrices is usually quite large. This size cannot be efficiently reduced using the techniques discussed in §4 on account of the fact that these techniques ignore parameter commutativity ($ab - ba$ cannot be simplified to zero). So, it is necessary to define new techniques specific to LFT gain order reduction.

The main functions are `fb_sched` (LFT-scheduled feedback gains)and `ob_sched` (LFT-scheduled observers). These functions have several options, in particular one of these options consists of re-assigning the eigenstructure assigned by any given standard dynamic controller. In that way, scheduling propagates the nominal performance of the given controller to other operating points. It is also intended to introduce Q-parametrization in this framework in order to treat simultaneously scheduling for varying parameters and robustness (by tuning Q) for uncertain parameters.

7 ANALYSIS OF SYSTEMS IN LFT FORM

The system and its controller being in LFT form, μ-analysis (for example for robust stability and performance robustness analysis, limit cycles analysis, worst case delay margin computation) can be performed in one shot.

In addition to these natural applications of μ-analysis, two alternative tools also based on μ-analysis but more or less specific to LFT gains, are proposed:

- *Well-posedness* radius computation.
- *Non-singularity* radius computation.

An LFT model has the form of (2) in which there is an inversion of $(I - A\Delta)$. The *well-posedness radius* is the maximum size of the parameter variations for which an LFT can be computed, *i.e.* $(I - A\Delta)$ can be inverted. Considering an LFT-scheduled gain, this is an important issue because it must be computed on-line, therefore, the inversion must remain feasible for all the relevant values of the scheduling parameters. The well-posedness radius test is used as follows. After the LFT-scheduled gain is designed, the scheduling parameters are normalized between -1 and $+1$ (see §3). Then, the gain can be implemented only if its well-posedness radius is larger than one. The *non-singularity radius* is a complementary tool that can be used for example for checking the controllability / observability of a model in LFT form and also before each inversion in order to be sure that the inverted objects will be well-posed.

More elementary analysis tools based on LFT parameter gridding are also proposed. The main function is `lfrview` based on the standard MATLAB function `ltiview`. This function is used for drawing families of locus or frequency/time domain responses. By clicking one curve in a family of curves the corresponding model of the gridding is displayed to the screen.

8 EXAMPLES

8.1 Example 1: Order reduction

To illustrate some enhancements available in the Version 2 of the LFR Toolbox, we generated an LFT model for the most complicated term a_{29} of the matrix A of the extended parametric RCAM [16]. The RCAM is

one of the most complicated existing parametric benchmark models in the literature. The RCAM contains four uncertain parameters: the mass m, two components of the position of the center of gravity X_{cg} and Z_{cg} and the trimmed air speed V_A. The expression of a_{29} can be put into the form

$$a_{29} = 0.061601 \frac{\tilde{a}_{29}}{C_w V_A}$$

where $C_w = \frac{mg}{\frac{1}{2}\rho V_A^2 S}$ and

$$\begin{aligned}
\tilde{a}_{29} =& 1.6726 X_{cg} C_w^2 Z_{cg} - 0.17230 X_{cg}^2 C_w \\
&- 3.9324 X_{cg} C_w Z_{cg} - 0.28903 X_{cg}^2 C_w^2 Z_{cg} \\
&- 0.070972 X_{cg}^2 Z_{cg} + 0.29652 X_{cg}^2 C_w Z_{cg} \\
&+ 4.9667 X_{cg} C_w - 2.7036 X_{cg} C_w^2 \\
&+ 0.58292 C_w^2 - 0.25564 X_{cg}^2 - 1.3439 C_w \\
&+ 100.13 X_{cg} - 14.251 Z_{cg} - 1.9116 C_w^2 Z_{cg} \\
&+ 1.1243 X_{cg} Z_{cg} + 24.656 C_w Z_{cg} \\
&+ 0.45703 X_{cg}^2 C_w^2 - 46.850.
\end{aligned}$$

The uncertain parameters can be normalized as follows

$$\begin{aligned}
m &= 125000 + 25000\ \delta m \\
X_{cg} &= 0.23 + 0.08\ \delta X_{cg} \\
Z_{cg} &= 0.105 + 0.105\ \delta Z_{cg} \\
V_A &= 80 + 10\ \delta V_A
\end{aligned}$$

where δm, δX_{cg}, δZ_{cg}, δV_A are, respectively, the normalized uncertain parameters.

By performing first the normalization of parameters and then generating an LFT-realization of a_{29}, the resulting block structure for

$$\Delta = \mathrm{diag}(\delta m I_{n_1}, \delta X_{cg} I_{n_2}, \delta Z_{cg} I_{n_3}, \delta V_A I_{n_4})$$

has $\{n_1, n_2, n_3, n_4\} = \{31, 54, 27, 81\}$. The total order n_Δ of Δ is $n_\Delta = 193$. Note, that the expression of a_{29} is "singular" in parameters m and V_A, and therefore normalization is obligatory for generation techniques relying on standard LFT-models. When using the generalized LFT-representation (including a constant block), we can avoid the preliminary normalization. The generated LFT-model for a_{29} has the uncertainty block dimensions $\{n_1, n_2, n_3, n_4\} = \{19, 18, 9, 69\}$ leading to a total order of $n_\Delta = 85$. This illustrated that often a preliminary normalization has the effect to increase substantially (more then twice in this example) the order of the generated LFT-realizations.

To illustrate the enhancements in order reduction capabilities of the toolbox, we performed on the 193th order model 1-d and n-d order reductions, using the pure MATLAB-based implementations (ML) and *mex*-function based implementations of the order reduction tools. In Table 1 we give the computational times resulted on a PC with a 1.2 GHz AMD ATHLON processor running under MATLAB 6.5 under Windows NT.

Reduction	Time [s]	$\{n_1, n_2, n_3, n_4\}$	n_Δ
1-d (ML)	9.61	$\{5, 2, 9, 28\}$	44
1-d (mex)	0.1	$\{5, 2, 4, 7\}$	18
n-d (ML)	0.54	$\{5, 2, 3, 7\}$	17
n-d (mex)	0.13	$\{5, 2, 3, 7\}$	17

Table 1: Order reduction results for RCAM element a_{29}

In Table 1 we can see a significant reduction of computational time for the 1-d reduction (almost 100 times faster) and also for the n-d reduction (more than four times faster). Note also that the 1-d reduction implementations generated using the mex-file based implementation has a much smaller order than the pure MATLAB-based implementation.

8.2 Example 2: Symbolic Preprocessing

The effectiveness of symbolic preprocessing can be seen from the following table, where we put the resulting orders of the LFT-realizations of the whole (not only element a_{29}) extended parametric RCAM [16]. For each specific run consisting of different symbolic preprocessing computations we list in the columns the resulting corresponding orders without and with additional numerical n-D order reduction [5].

Symbolic Preprocessing	without reduction	with reduction
None	400	262
Single element enhancements	260	158
TD	156	97
VS+ETD	77	65

Table 2: Orders of LFT-realizations for the extended RCAM example.

The parametric matrices of the RCAM have only elements as polynomials with positive and/or negative powers in the indeterminates (see element a_{29} in Example 1). Thus, this model perfectly fits to the proposed ETD algorithm. Without symbolic preprocessing, an order of 262 can be achieved by using numerical order reduction. Using various symbolic techniques on single rational matrix elements followed by application of numerical n-D order reduction, an LFT representation of order 158 has been computed in [15]. The TD algorithm for a polynomially factorized representation as proposed in [4] yields an LFT-model of order 156, which can be reduced to order 97. With the proposed symbolic preprocessing tools (VS+ETD) implemented in the LFR-toolbox, we obtained an LFT-representation of the aircraft model with order 77 and we could exactly reduce this model to order 65, which is very close to the theoretical lower bound of 56. Note, using symbolic tools we obtained an LFT-representation of order 11 for the element a_{29}. Compared to order 17 (see example 1) with numerical order reduction, one can clearly see the capabilities of symbolic preprocessing.

9 CONCLUSION

We presented the new developments and enhancements available in Version 2 of the LFR-Toolbox. The introduction of a generalized LFT-object allows to realize arbitrary rational parametric matrices as LFTs. No preliminary normalization of parameters is necessary, which generally yields LFT-representations of lower order than expected with using standard LFT-realization approaches. In the new LFT-object each block in the feedback matrix Δ is clearly identified by a name, which improves the flexibility and user-friendliness of the toolbox. To be directly compatible with other MATLAB toolboxes (e.g. μ-Analysis/Synthesis) the uncertainty properties nonlinear and time-varying are now supported. The calculation of reduced order LFT-realizations relies on efficient and numerically reliable *mex*-functions for basic system order reductions (minimal realization, staircase controllability/observability forms, model reduction). Version 2 of the LFR-Toolbox offers improved symbolic preprocessing capabilities, which are very efficient for low order LFT-realization. By means of the RCAM example we illustrated some of the main enhancements.

We also presented the new control analysis and synthesis capabilities of version 1 of the toolbox. These tools will be ported to version 2 after validation. The tools for LFT-scheduled gain design proposed in §6 are intended to permit the designer to apply μ-analysis for non-fixed gains. However, the problem of LFT-gain order reduction must be addressed because it cannot be treated with the classical reduction tools presented in §4. The feasibility of on-line computation of LFT-scheduled gain is also considered, this is an important issue because the computation of an LFT involves the inversion of a matrix depending on scheduling parameters.

References

[1] E. Anderson, Z. Bai, J. Bishop, J. Demmel, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK User's Guide, Second Edition.* SIAM, Philadelphia, France, 1995.

[2] P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, and A. Varga. *SLICOT - A Subroutine Library in Systems and Control Theory.* In B. N. Datta, Ed., Applied and Computational Control, Signals and Circuits. Birkhäuser, 1998.

[3] N. K. Bose. *Applied Multidimensional Systems Theory.* Van Nostrand Reinhold Company, 1982.

[4] J.C. Cockburn and B.G. Morton. Linear fractional representation of uncertain systems. *Automatica*, 33(7):1263–1271, 1997.

[5] R. D'Andrea and S. Khatri. Kalman decomposition of linear fractional transformation representations and minimality. In *Proc. of the American Control Conference*, pages 3557–3561, Albulquerque, New Mexico, 1997.

[6] C. Döll, Y. Le Gorrec, G. Ferreres, and J.F. Magni. Design of a robust self-scheduled missile autopilot by multi-model eigenstructure assignment. *Control Engineering and Practice, Special Issue on Defense*, 2001.

[7] S. Hecker and A. Varga. Generalized LFT-based representation of parametric uncertain models. *European Journal of Control (accepted)*, 2004.

[8] H.D. Joos. A methodology for multi-objective design assessment and flight control synthesis tuning. *Aerospace Science and Technology*, 3(3):161–176, 1999.

[9] P. Lambrechts, J. Terlouw, S. Bennani, and M. Steinbuch. Parametric uncertainty modeling using LFTs. In *Proc. American Control Conference*, pages 267–272, San Francisco, CA, 1993.

[10] J. F. Magni. Presentation of the Linear Fractional Representation Toolbox (LFRT). In *Proc. CACSD'2002 Symposium, Glasgow, Scotland*, 2002.

[11] J. F. Magni. *Linear Fractional Representations with a Toolbox: Modelling, order reduction, gain scheduling. Version 1.1.* Departement of Systems Control and Flight Dynamics, ONERA–CERT, Toulouse, France, January 2004.

[12] J.F. Magni. Multimodel eigenstructure assignment in flight-control design. *Aerospace Science and Technology*, 3(3):141–151, 1999.

[13] B. Morton. New applications of mu to real.parameter variation problems. In *Proc. Conference on Decision and Control*, pages 233–238, Fort Lauderdale, Florida, 1985.

[14] A. Varga. Model reduction software in the SLICOT library. In B. N. Datta, editor, *Applied and Computational Control, Signals and Circuits*, volume 629 of *The Kluwer International Series in Engineering and Computer Science*, pages 239–282. Kluwer Academic Publishers, Boston, 2001.

[15] A. Varga and G. Looye. Symbolic and numerical software tools for LFT-based low order uncertainty modeling. In *Proc. CACSD'99 Symposium, Kohala Coast, Hawaii*, 1999.

[16] A. Varga, G. Looye, D. Moormann, and G. Grübel. Automated generation of LFT-based parametric uncertainty descriptions from generic aircraft models. *Mathematical and Computer Modelling of Dynamical Systems*, 4:249–274, 1998.

[17] K. Zhou, J. C. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice Hall, 1996.