Available online at www.sciencedirect.com

SCIENCE DIRECT®

Aerospace Science and Technology 9 (2005) 173–180

Aerospace Science and Technology

www.elsevier.com/locate/aescte

Enhanced LFR-toolbox for Matlab ☆

Erweiterte LFR-Toolbox für Matlab

Simon Hecker [a,*], Andras Varga [a], Jean-François Magni [b]

[a] *German Aerospace Center (DLR), Institute of Robotics and Mechatronics, 82230 Wessling, Germany*
[b] *Office National d'Études et de Recherches Aérospatiales (ONERA), 2, av. Edouard Belin, 31055 Toulouse, France*

Received 12 May 2004; received in revised form 15 September 2004; accepted 2 December 2004

Available online 8 January 2005

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-representations 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-representations with significantly lower orders.
© 2005 Elsevier SAS. All rights reserved.

Zusammenfassung

Dieser Aufsatz beschreibt aktuelle Entwicklungen und Erweiterungen der Matlab LFR-Toolbox zur Realisierung LFT-basierter Unsicherheitsmodelle und den Entwurf LFT-basierter Gain-Scheduling-Regler. Eine entscheidende Verbesserung der Toolbox wurde durch die Einführung eines neuen LFT-Objektes erreicht. Damit wird eine große Klasse von Unsicherheitsbeschreibungen unterstützt: zeitkontinuierliche und zeitdiskrete Modelle, reguläre und singuläre parametrische Modelle, nichtlineare sowie zeitvariante Modelle. Jeder Modellunsicherheit wird ein eindeutiger Name zugewiesen und somit die Wiederverwendbarkeit der LFT-Modelle und die Benutzerfreundlichkeit der Toolbox verbessert. Erhebliche Steigerungen der Rechengeschwindigkeit sowie der numerischen Genauigkeit, wurden durch die Anbindung effizienter und numerisch robuster Fortran Implementierungen von Ordungsreduktionsroutinen erreicht. All diese Verbesserungen, in Verbindung mit verbesserten symbolischen Vorverarbeitungswerkzeugen, ermöglichen eine effiziente Erzeugung von LFT-Modellen mit niedriger Ordnung.
© 2005 Elsevier SAS. All rights reserved.

Keywords: Robust control; Linear fractional transformation (LFT); Matlab toolbox; Linear fractional representation (LFR)

Schlüsselwörter: Robuste Regelung; Unsicherheitsmodellierung; Matlab Toolbox

1. Introduction

In modelling uncertainties in linear systems the *linear fractional transformation* (LFT) plays an important role. LFT-based representations of model uncertainties (see Fig. 1) are frequently used in modern robust control meth-

☆ This article was presented at ODAS 2004.
* Corresponding author. Phone: +49 (0)8153 28 3374, fax: +49 (0)8153 28 1441.
E-mail addresses: simon.hecker@dlr.de (S. Hecker), andras.varga@dlr.de (A. Varga), jean-francois.magni@cert.fr (J. Magni).

1270-9638/$ – see front matter © 2005 Elsevier SAS. All rights reserved.
doi:10.1016/j.ast.2004.12.001

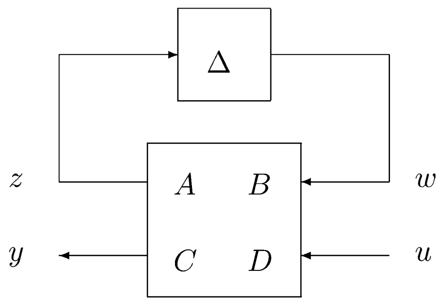

Fig. 1. LFT-representation.

ods like the structured singular value (also called μ) [20]. LFT-representations are also used for modelling, analysis and controller synthesis for multi-dimensional [3] or interconnected systems [6] (see also [5] presenting a toolbox for multi-dimensional systems). The applicability of LFT-based methods for control problems in the aerospace field has been shown for example in [14], where controllers for systems with large parameter variations (e.g., altitude and mass of an aircraft) and parametric uncertainties (e.g., center of gravity, aerodynamic coefficients) were successfully designed.

The LFT-representation in Fig. 1 is described by the system equations

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

where the feedback matrix Δ has usually a block diagonal structure. For a 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 corresponding to Fig. 1 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 order of this LFT-representation is m_1, the row dimension of the $m_1 \times p_1$ block-diagonal matrix Δ.

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) [16].

The version 1 of the LFR-toolbox has been implemented by the third author [11] and supports structured (parametric) or unstructured uncertainties of real or complex type. The generation of low order LFT-realizations is supported in various ways. Special functions for symbolic preprocessing techniques as Mortons method [15] for affine uncertainty representations and the tree decomposition [4] for polynomial matrices are provided. Furthermore numerical multidimensional order reduction and approximation methods [7,16] for LFT-representations are available. These algorithms rely on standard minimal realization tools available in the Control Toolbox of Matlab.

In this paper we present version 2 of the LFR-toolbox representing recent developments and enhancements of version 1, which are mainly focused to improve the capabilities for low order LFT-modelling. With the definition of a new LFT-object, supporting also constant blocks in Δ [10] (equivalent behavioral LFT-representations are presented in [8]), we can realize arbitrary rational expressions in LFT-form. In this way, we circumvent the problem, that for the object oriented LFT-realization approach rational expressions like $1/p$ had to be symbolically normalized before realizing the LFT-representation. 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. Version 2 includes also new LFT-based analysis and controller synthesis capabilities.

The organization of the paper is as follows. In Section 2 the new LFT-object definition, based on a generalized LFT-representation, is introduced and the object oriented manipulation based generation of LFT-models is described. The normalization of generalized LFT-representation and the corresponding normalization function are presented in Section 3. The improved symbolic preprocessing and numerical post-processing (order reduction) capabilities of version 2 of the toolbox are presented in Sections 4 and 5, respectively. Functions dedicated to analysis of LFT-representations are presented in Section 6.[1] In Section 7 we present two examples illustrating the capabilities of the toolbox to generate low order LFT-representations.

2. Object oriented LFT-realization

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. An LFT-object `L` can be created with the command

```
L = lfr(A,B,C,D,blk);
```

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

where the first four input arguments specify the matrices A, B, C, D (see Fig. 1) and the fifth argument `blk` describes the block-diagonal structure of Δ. The argument `blk` is a structure array with two fields, `names` and `desc`, containing, respectively, the names associated to the diagonal blocks of Δ and the corresponding uncertainty type description. The five input arguments can be recovered from the object `L` as the fields `L.a`, `L.b`, `L.c`, `L.d`, and `L.blk`, respectively.

As an example, the fields `names` and `desc` of the structure description argument of an LFT-object with $\Delta = \operatorname{diag}(p_1 I_2, p_2)$ can be specified as

```
blk.names = {'p1','p2'};
blk.desc = [ 2 1     % row-dimensions
                     % of blocks
             2 1     % column-dimensions
                     % of blocks
             1 1     % real(1) / complex(0)
                     % block types
             1 1     % scalar(1) / full(0)
                     % block types
             1 1     % linear(1) / nonlinear (0)
                     % block types
             1 1     % time-inv.(1) /
                     % time-var.(0) block types
             1 1     % min/max(1) / sector(2) /
                     % freq. dependent(>2) bounds
             2 2     % min/max(2) / sector(1) /
                     % freq. dependent(>2) bounds
            -1 -1    % minumum values of bounds
             1 1 ] ; % maximum values of bounds
```

where `blk.names` is a cell-array of two strings containing the names `'p1'` and `'p2'` given to the two diagonal blocks of Δ, and the values in each column of the real array `blk.desc` specifies the corresponding information describing each diagonal block (see below).

Each block in Δ is uniquely identified by its name, which makes the manipulation of LFT-objects flexible and transparent. For example, additional uncertainties can be easily added to an LFT-object and the already defined block names can be modified (e.g., by using the function `set`). The special 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 can be included in Δ to represent standard linear time-invariant systems (continuous- or discrete-time) as LFT-objects. Furthermore the special name `'1'` is reserved for a constant identity matrix block I in Δ. This block plays a major role in representing singular parametric expressions as LFT-representations. 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 a lexicographic order.

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.

When using only *standard LFT-representations* (i.e., without a constant block in Δ) it is generally not possible to represent arbitrary rational expressions in LFT-form. For example, to construct LFT-realizations for expressions containing *singular* factors or terms like $1/p$, a symbolic normalization of the respective parameters usually has to be performed before determining the LFT-representation. However, since symbolic normalization tends generally to increase the order of the generated LFT-representations (see [4,12] for examples), it is highly desirable to avoid any preliminary normalization when building LFT-representations. One way for that is to employ alternative LFT-representations, as for example, the *descriptor LFT-representation* proposed in [10]. Besides its ability to represent arbitrary rational expressions in LFT-forms, this representation allows to represent even generalized systems described by algebraic-differential equations (so-called descriptor systems) as LFT-based realizations. In 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. This simple extension of the standard LFT-representation allows to represent arbitrary rational parametric expressions as LFT-representations, albeit descriptor systems can not be directly represented using just a constant block (a transformation to a special coordinate form is necessary). A major advantage of this representation is that the constant block in Δ can be considered as an additional dimension in a multidimensional system representation [3] and all standard multidimensional LFT manipulation techniques (e.g., the order reduction methods of [7,16]) can be applied to the generalized LFT-representation without any modification.

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

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

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 $\Delta_{lf} = \text{diag}(I_p, \Delta)$ 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 [16] and extended in [10] 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

$$\Delta = \text{diag}(I_{n_c}, I_{n_d}/\lambda, \delta_1 I_{r_1}, \ldots, \delta_k I_{r_k}, \widehat{\Delta}), \tag{3}$$

where I_{n_c} denotes an $n_c \times n_c$ identity matrix, I_{n_d}/λ is a $n_d \times n_d$ integrator block in continuous-time case ($\lambda = s$) or a delay block for discrete-time systems ($\lambda = z$), $\delta_j I_{r_j}$, $j = 1, \ldots, k$, are k real parametric uncertainty blocks, and $\widehat{\Delta}$ is a block-diagonal $\hat{n}_1 \times \hat{n}_2$ matrix, 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 $\delta_{i,n} + \delta_{i,sl}\bar{\delta}_i$, with $\delta_{i,n} := (\delta_{i,min} + \delta_{i,max})/2$ and $\delta_{i,sl} := (\delta_{i,max} - \delta_{i,min})/2$ such that $|\bar{\delta}_i| \leqslant 1$, for $i = 1, \ldots, k$. Thus, the normalization amounts to replace Δ by $\Delta_n + \Delta_{sl}\overline{\Delta}$ in $\mathcal{F}_u(M, \Delta)$, where

$$\Delta_n = \text{diag}(I_{n_c}, 0_{n_d \times n_d}, \delta_{1,n} I_{r_1}, \ldots, \delta_{k,n} I_{r_k}, 0_{\hat{n}_1 \times \hat{n}_2}), \tag{4}$$

$$\Delta_{sl} = \text{diag}(0_{n_c \times n_c}, I_{n_d \times n_d}, \delta_{1,sl} I_{r_1}, \ldots, \delta_{k,sl} I_{r_k}, I_{\hat{n}_1}), \tag{5}$$

$$\overline{\Delta} = \text{diag}(0_{n_c \times n_c}, I_{n_d}/\lambda, \bar{\delta}_1 I_{r_1}, \ldots, \bar{\delta}_k I_{r_k}, \widehat{\Delta}), \tag{6}$$

where $0_{n \times m}$ denotes the $n \times m$ null matrix.

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

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

and

$$\widetilde{M} = \left[\begin{array}{c|c} \overline{A}_{22} & \overline{B}_2 \\ \hline \overline{C}_2 & \overline{D} \end{array}\right],$$
$$\widetilde{\Delta} = \text{diag}(I_{n_d}/\lambda, \bar{\delta}_1 I_{r_1}, \ldots, \bar{\delta}_k I_{r_k}, \widehat{\Delta}).$$

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. 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 [4,15,18]. 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 [18] as basis to generate lower order LFT-representations. Alternatively, conversions to partial fraction form or continuous fraction form may be very efficient to obtain low order LFT-representations.

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 multivariate Laurent polynomials of the indeterminates. Additionally, further enhancements were obtained by integration of Morton's method [15] in the ETD algorithm and by extending the VS factorization 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.

5. Enhanced order reduction

LFT-representations 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-representations are almost always non-minimal, and therefore using appropriate numerical tools to perform *exact* order reduction of LFT-representations 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-representations 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 [16] 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 [18] can be performed using `redlfr1`, which is based on the collection of model reduction tools available in SLICOT [17], 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-representation (see Fig. 1), this function can be also employed to perform *exact* order reduction.

The function `minlfr` can be used for n-d order reduction [7]. In version 2 of the LFR-toolbox 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-representations as a preliminary operation within the order reduction routines. As the LFT-representations resulting from the object oriented realization approach [16] can have matrices with a wide range of values this operation is essential before computing numerically 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.

6. Analysis of systems in LFT-form

The main applications of LFT-representations are in performing stability and performance robustness assessment using the μ-analysis. Two complementary tools, `wp_rad` and `ns_rad`, are available to check the well-posedness of LFT-representations by computing the so-called *well-posedness* and *non-singularity* radii, respectively. Both functions rely on computing the structured singular value of certain matrices.

For a well-posed LFT-representation of the form (2) the invertibility of $I - A\Delta$ is tacitly assumed for all admissible values of parameters. The *well-posedness radius* is the maximum allowable size of the parameter variations for which the LFT-representation can be evaluated (i.e., $I - A\Delta$ can be inverted). For example, LFT-scheduled controller gains can be safely implemented, provided for all values between -1 and $+1$ of the normalized scheduling parameters $I - A\Delta$ is invertible. This means that the well-posedness radius must be generally larger than one. The *non-singularity radius* is a complementary tool related to invertibility of the LFT-model itself. Provided D in (2) is non-singular, the *non-singularity radius* is in fact the *well-posedness radius* related to the invertibility of $I - (A - BD^{-1}C)\Delta$.

More elementary analysis tools of LFT models based on parameter gridding are also available. The main function is `lfrview` based on the standard Matlab function `ltiview`. This function can be used for drawing families of pole/zero plots or frequency/time domain responses.

7. Examples

7.1. Example 1: order reduction

To illustrate the effect of preliminary normalization and the enhancements of order reduction tool, we generated an LFT-representation for the most complicated term a_{29} of the parametric state dynamics matrix $A(\delta)$ of the extended parametric RCAM [19]. This transport aircraft model, one of the most complicated parametric models documented in the literature, 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 $C_w = mg/((1/2)\rho V_A^2 S)$ and

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

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-representation of a_{29}, the resulting block structure for

$$\Delta = \mathrm{diag}(\delta m I_{r_1}, \delta X_{cg} I_{r_2}, \delta Z_{cg} I_{r_3}, \delta V_A I_{r_4})$$

has $\{r_1, r_2, r_3, r_4\} = \{31, 54, 27, 81\}$. 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 only on standard LFT-models. When using the generalized LFT-representation (including a constant block in Δ), we can avoid the preliminary normalization. The generated LFT-model for a_{29} has the uncertainty block dimensions $\{r_1, r_2, r_3, r_4\} = \{19, 18, 9, 69\}$ leading to a total order of $n_\Delta = 85$. This illustrates clearly that a preliminary normalization has often the effect to increase substantially (more then twice in this example) the order of the generated LFT-representations.

To illustrate the enhancements of order reduction capabilities of the toolbox, we performed on the 193th order model 1-d and n-d order reductions, using pure *m*-function based implementations (M) and *mex*-function based implementations (MEX) 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 Matlab 6.5 under Windows NT.

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

Reduction	Time [s]	$\{r_1, r_2, r_3, r_4\}$	n_Δ
1-d (M)	9.61	{5, 2, 9, 28}	44
1-d (MEX)	0.1	{5, 2, 4, 7}	18
n-d (M)	0.54	{5, 2, 3, 7}	17
n-d (MEX)	0.13	{5, 2, 3, 7}	17

It can be seen a significant reduction of computational times for both the 1-d reduction (almost 100 times faster!) and the n-d reduction (more than four times faster). Note also that for this example, the 1-d reduction performed using the *mex*-file based implementation led to a much smaller order than the pure *m*-file based implementation.

7.2. Example 2: symbolic preprocessing

To illustrate the effectiveness of symbolic preprocessing in obtaining low order LFT-realizations we applied the variable splitting technique in combination with the extended tree decompositions to the a_{29} element in Example 1. The resulting realization has order 11 and we conjecture that this LFT-representation is of minimal order. Compared to the least order 17 (see Example 1) obtained with numerical order reduction, one can clearly see the strengths of employing symbolic preprocessing tools.

Table 2
Orders of LFT-representations for the extended RCAM example

Symbolic preprocessing	Order	Order (numerically reduced)
None	400	262
Single element	260	158
TD	156	97
VS+ETD	71	65

The power of symbolic preprocessing can be best seen from Table 2, containing the resulting orders of the LFT-representations of the complete extended parametric RCAM [19]. For the parametric state-space system defined by quadruple of matrices $(A(\delta), B(\delta), C(\delta), D(\delta))$, we computed several LFT-representations of the parametric system matrix

$$S(\delta) = \begin{bmatrix} A(\delta) & B(\delta) \\ C(\delta) & D(\delta) \end{bmatrix}$$

by applying different symbolic preprocessing techniques followed by numerical n-D order reductions [7]. The corresponding resulting orders are presented in the successive columns of Table 2.

The main feature of RCAM is that all its parametric system matrices have only rational elements which can be assimilated with multivariate Laurent polynomials (see element a_{29} in Example 1). Thus, this model perfectly fits to the 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 [18]. The TD algorithm for a polynomially factorized representation as proposed in [4] yields an LFT-model of order 156, which can be further reduced to order 97. With the symbolic preprocessing tools (VS+ETD) available in the LFR-toolbox, we obtained an LFT-representation of the aircraft model with order 71 and we could exactly reduce this model to order 65, which is very close to the theoretical least order bound of 56.

8. 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 in LFT-form. Since no preliminary normalization of parameters is necessary, the resulting LFT-representations have usually lower orders than those obtained when employing only standard LFT-representations based approaches. A new LFT-object has been defined, in which each diagonal block in the uncertainty matrix Δ is identified by a unique name. This improves significantly the flexibility and user-friendliness of the current version of toolbox. To ensure compatibility with other Matlab toolboxes (e.g. μ-Analysis/Synthesis) new uncertainty properties (e.g., nonlinear, time-varying) have been included in the LFT-object definition. The calculation of reduced order LFT-representations relies on efficient and numerically reliable algorithms for basic system order reductions (minimal realization, staircase controllability/observability forms, model reduction). These algorithms are implemented in the Fortran 77 library SLICOT [2] and accessed via *mex*-functions. 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.

The presented new LFT-based realization and analysis capabilities available in version 2 of the LFR toolbox are mainly intended to perform LFT-based robust control design (e.g., using μ-synthesis). However, the available tools can be also employed to assess the robustness of controllers designed by any other method. An interesting domain is the design and assessment of gain scheduling controllers. These controllers are traditionally designed as a bank of linear feedback compensators, whose gains are interpolated depending on the values of some measurable physical parameters. Recently proposed approaches addresses the design problem of scheduled control laws either transforming the problem into a robust control design problem [9,13], or using symbolic design approaches. An LFT-based eigenstructure assignment approach has been proposed in [12]. The resulting scheduled gains can be expressed easily in LFT-forms and employed in robustness analysis.

References

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

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

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

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

[5] R. D'Andrea, Software for modeling, analysis, and control design for multidimensional systems, in: Proc. of the IEEE International Symposium on Computer-Aided Control System Design, 1999, pp. 24–27.

[6] R. D'Andrea, G.E. Dullerud, Distributed control design for spatially interconnected systems, IEEE Trans. Automatic Control 48 (9) (2003).

[7] R. D'Andrea, S. Khatri, Kalman decomposition of linear fractional transformation representations, in: Proc. of the American Control Conference, Albulquerque, NM, 1997, pp. 3557–3561.

[8] R. D'Andrea, F. Paganini, Interconnection of uncertain behavioral systems for robust control, in: Proc. of the 32nd Conference on Decision and Control, San Antonio, TX, 1993, pp. 3642–3647.

[9] C. Döll, Y. Le Gorrec, G. Ferreres, 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).

[10] S. Hecker, A. Varga, Generalized LFT-based representation of parametric uncertain models, European J. Control 4 (2004).

[11] J.F. Magni, Presentation of the Linear Fractional Representation Toolbox (LFRT), in: Proc. CACSD'2002 Symp., Glasgow, Scotland, 2002.

[12] J.F. Magni, Linear Fractional Representations with a Toolbox: modelling order reduction, gain scheduling, Version 1.1, Department of Systems Control and Flight Dynamics, ONERA-CERT, Toulouse, France, January 2004.

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

[14] J.F. Magni, S. Bennani, J. Terlouw, Robust Flight Control – A Design Challenge, Lecture Notes in Control and Information Sciences, vol. 224, Springer, Berlin, 1997.

[15] B. Morton, New applications of mu to real.parameter variation problems, in: Proc. Conference on Decision and Control, Fort Lauderdale, FL, 1985, pp. 233–238.

[16] J. Terlouw, P. Lambrechts, S. Bennani, M. Steinbuch, Parametric uncertainty modeling using LFTs, in: Proc. American Control Conference, 1993, pp. 267–272.

[17] A. Varga, Model reduction software in the SLICOT library, in: B.N. Datta (Ed.), Applied and Computational Control, Signals and Circuits, in: The Kluwer International Series in Engineering and Computer Science, vol. 629, Kluwer Academic, Boston, 2001, pp. 239–282.

[18] A. Varga, G. Looye, Symbolic numerical software tools for LFT-based low order uncertainty modeling, in: Proc. CACSD'99 Symp., Kohala Coast, Hawaii, 1999.

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

[20] K. Zhou, J.C. Doyle, K. Glover, Robust Optimal Control, Prentice Hall, Englewood Cliffs, NJ, 1996.