Nonlinear symbolic LFT tools for modelling, analysis and design

A. Marcos[1*], D.G. Bates[2], and I. Postlethwaite[2]

[1] Deimos-Space S.L., Madrid, Spain. andres.marcos@deimos-space.com
[2] University of Leicester, United Kingdom dgb3,ixp@le.ac.uk

Abstract: In this chapter, a general nonlinear symbolic LFT modelling framework and its support LFT tools are presented. The modelling approach developed combines the natural modularity and clarity of presentation from LFT modelling with the ease of manipulation from symbolic algebra. It results in an exact nonlinear symbolic LFT that represents an ideal starting point to apply subsequent assumptions and simplifications to finally transform the model into an approximated symbolic LFT ready for design and analysis. The development of the framework is supported by a novel algebraic algorithm for symbolic matrix decomposition and two new LFT operations: nested substitution and symbolic differentiation.

1 Introduction

New control developments in the field of on-ground aircraft control are seen as a key technology to reduce the congestion of many airports while increasing safety during on-ground manoeuvres [10]. These new control developments necessitate of mathematical models that consider the aerodynamic forces but also, and more critically, the ground forces affecting the aircraft. The complexity of these ground forces make the modelling task challenging, with highly nonlinear effects and many uncertain parameters influencing the various phenomena considered. Consequently, it is necessary to develop modelling approaches capable of capturing all the on-ground aircraft complexity but which also allow simplification, in a systematic manner, of the high-fidelity model initially obtained down to control-oriented (on-ground aircraft) models.

Once the control laws have been synthesized and validated in the simplified mathematical models, it is required by public certification authorities that the resulting control system be also validated in the most complex models.

* First author was a post-doctoral fellow at the University of Leicester supported by EPSRC-UK research grant $GR/S61874/01$ when this research was performed.

With this certification objective in mind, many industries are recently making significant efforts to develop and apply nonlinear synthesis and analysis techniques - see for example, the work reported in [11] for recent progress in the aerospace industry. One approach to this certification problem, which has been very successful in practice, is to extend traditional linear design and analysis methods to address nonlinear problems. This is the basis of modern synthesis and analysis techniques such as gain scheduling [15], linear parameter varying (LPV) control [3] and integral quadratic constraints [19] among others.

It is noted that many of the above techniques work with models based on, or similar to, the linear fractional transformation (LFT) modelling paradigm [20]. Thus, it is further advantageous to develop a systematic nonlinear modelling framework based on LFT representations and offering the flexibility and modularity required to obtain the simplified models (used for control design) but also connecting them to the original & more complex models (used for certification).

In this chapter, such a modelling approach is presented. The basic idea of the approach is to rely on symbolic manipulations and LFT representations to provide the desired flexibility of modelling and ease of manipulation. In supporting such a modelling framework a novel algebraic matrix decomposition algorithm has been developed together with algorithmic formalizations for two new LFT operations: nested LFT substitution and symbolic LFT differentiation. The layout of the chapter is as follows: Section 2 presents the supporting LFT tools and algorithm, Section 3 covers the proposed nonlinear symbolic LFT modelling framework, and Section **??** shows examples of the application of the approach and supporting tools.

2 LFT and Symbolic Modelling Support Tools

In this section, the LFT tools developed in support of the proposed modelling framework are described.

The novel symbolic matrix decomposition algorithm, called Logic-Horner-Tree (LHT), is detailed first, including its extension to LFT modelling. Subsequently, the formalization, and proofs (which provide their algorithmic implementations), of two new LFT operations are given: nested substitution and symbolic differentiation.

2.1 Logic-Horner-Tree algorithm

Matrix manipulation is one of the basic cornerstones of many fields in engineering and mathematics. For example, polynomial/matrix representations and manipulations are ubiquitous in many areas of mathematics and most computer algebra systems such as *Mathematica* [26] and Maple [12] rely on them.

A typical objective (for example, in signal processing and control synthesis) when operating on matrices and polynomials, especially when these are multivariate, is to obtain an equivalent representation of reduced order (in terms of number of parameters and their repetitions). Ideally, the order should be minimal, but minimal representations are in general very difficult to obtain except for some simple cases.

In the case of LFT modelling, it is well-known that the problem of finding a minimal order representation is equivalent to a multidimensional realization [7] which remains an open problem. Indeed, most of the available LFT algorithms search for a minimal representation by exploiting the structure in the LFT models and performing algebraic factorizations on the multivariate matrices: for example, structured-tree decomposition [8], numerical matrix approaches [6], Horner factorizations [24] and symbolic linearizations [25] among others.

In this section an algorithm is proposed to achieve a lower order representation for multivariate polynomial matrices. This proposed decomposition algorithm is called Logic-Horner-Tree (LHT) to highlight its connection to previous approaches [8, 24, 25] and to emphasize the inclusion of decision logic. Indeed, the algorithm generalizes those approaches, as it follows the layout of the structured-tree decomposition approach but uses the Horner factorization operation which has now been extended for multivariate polynomial matrices.

The LHT algorithm is applicable to symbolic matrices whose coefficients are either constants or monomials/polynomials in one or several parameters, i.e. symbolic multivariate polynomial matrices. The symbolic parameters might represent highly complex functions dependent on many other parameters. Furthermore, it can be extended to symbolic rational expressions $P(\delta_i)$ using the generalized approach from [13] or the factorization result by [5, 16]:

$$P(\delta_i) = N(\delta_i)D(\delta_i)^{-1} = \tilde{D}(\delta_i)^{-1}\tilde{N}(\delta_i) \tag{1}$$

where $N(\delta_i)$,$D(\delta_i)$,$\tilde{D}(\delta_i)$ and $\tilde{N}(\delta_i)$ are polynomial matrices on the variables δ_i.

The algorithm is divided into three main routines: information management (IM), factorization (Horner and affine) and sum decomposition. The iterative combination of the last two routines yields a nested structure for the decomposition:

$$M = M_{B_1} + M_{A_1} = M_{B_1} + L_1\Big(\dots(M_{B_n} + L_n\bar{M}_{A_n}R_n)\dots\Big)R_1 \tag{2}$$

The matrices $L_{ii}, \bar{M}_{A_{ii}}, R_{ii}$ are obtained from the factorization of the primary matrix $M_{A_{ii}}$ which is obtained together with the secondary matrix $M_{B_{ii}}$ from the sum decomposition of $\bar{M}_{A_{ii-1}}$ (with $M_{A_1} = L_1\bar{M}_{A_1}R_1$).

Each of the algorithm steps is detailed next, starting by the definition of the metrics used and ending with the extension of the algorithm to LFT modelling.

Metrics

The main metrics used by the algorithm are the presence degree σ, the factor order fac, the reduction order red, and the possible reduction order red_{pos}.

The above metrics are based on the following standard monomial and polynomial definitions. Given n symbolic parameters $\delta_1, \delta_2, \dots, \delta_n$ a monomial m is defined as $m = c\delta_1^{\alpha_1}\delta_2^{\alpha_2}\dots\delta_n^{\alpha_n}$ where c is a non-zero constant coefficient and $\alpha_i \in \mathbb{Z}_+$ represents the corresponding power for the i-th parameter. The extension to negative powers is direct noting that $\delta_i^{-\alpha_i} = \hat{\delta}_i^{\alpha_i}$ where $\hat{\delta}_i = \delta_i^{-1}$ is considered a new symbolic parameter.

A polynomial p is given by a finite linear combination of k monomials, $p = \sum_{j=1}^{k} m_j$. The total degree of a monomial is $deg(m) = \sum_{i=1}^{n} \alpha_i$; the relative degree of a monomial defined with respect to a parameter and is given by $deg_{\delta_i}(m) = \alpha_i$; the degree of a polynomial is $deg(p) = max\, deg(m_j)$.

The presence degree $\sigma(\delta_i)$ is defined as the number of times, including powers, a parameter δ_i appears in an expression (symbolic monomial, polynomial or matrix). It can be viewed as a polynomial, or matrix, extension of the relative degree of a monomial. The total σ degree is the sum of the $\sigma(\delta_i)$ for all the symbolic parameters δ_i. The factor order of a parameter $fac(\delta_i)$ is the maximum power to which it can be factored out from an expression.

The reduction order for a parameter $red(\delta_i)$, is the largest reduction in the presence degree of an expression achievable through factorization of that parameter. Assuming there are k factorizable monomials (i.e. each monomial contains the parameter with a minimum order equal to $fac(\delta_i)$) in the expression, $red(\delta_i)$ is given by $(k-1)\,fac(\delta_i)$. The last metric, $red_{pos}(\delta_i)$, is similar to $red(\delta_i)$ but considering there are l non-factorizable monomials in the expression: $(k-l-1)\,fac(\delta_i)$.

Information Management (IM) routine

The IM routine condenses the logic of the algorithm and allows for a completely automated procedure without the need to apply combinatorics to the symbolic parameter vector ordering (a key issue for algebraic symbolic decomposition algorithms).

Its main objectives are (these are performed at different stages of the algorithm, see the algorithm pseudo-code in Appendix A):

i) to gather the proper information at each step
ii) to take a decision regarding operation and parameter ordering
iii) to prepare the decomposition matrix for the chosen operation.

The information gathering pertains mainly to calculating the metrics, called 'METRICS' in the pseudo-code, presented above for all the specified symbolic parameters and also in identifying the polynomial coefficients. This information is used to decide on the appropriate operation and to prepare the matrix accordingly. For example, even if a high factor order $fac(\delta_k)$ is identified along any row or column, if a higher possible reduction order $red_{pos}(\delta_k)$ is also obtained the corresponding symbolic parameter can be scheduled for a different operation, see example 1:

Example 1. Given $M = \begin{bmatrix} \delta_3^2 & \delta_1 \\ \delta_3^2 & \delta_3^2 \\ \delta_1\delta_3 + \delta_2\delta_3^2 & \delta_3^3 \end{bmatrix}$, *we could start using a direct affine factorization of* δ_3, *since* $fac(\delta_3)_{2row} = 2$ *and* $fac(\delta_3)_{3row} = 1$ *which yield* $red(\delta_3) = (k_{2row} - 1) * fac(\delta_3)_{2row} + (k_{3row} - 1) * fac(\delta_3)_{3row} = 1*2 + 2*1 = 4$, *and which results in* M_{dec_1} *whose total presence degree is* $\sigma(M_{dec_1}) = \sigma(M) - red(\delta_3) = 11$*:*

$$M_{dec_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \delta_3^2 & 0 \\ 0 & 0 & \delta_3 \end{bmatrix} \begin{bmatrix} \delta_3^2 & \delta_1 \\ 1 & 1 \\ \delta_1 + \delta_2\delta_3 & \delta_3^2 \end{bmatrix}$$

Further manipulations on M_{dec_1} will yield a total $\sigma = 10$. On the other hand, noting that for the initial M the possible reduction order $red_{pos}(\delta_3)$ along the columns is 6, then an affine factorization of δ_3 can be performed along the 1st column followed by a sum decomposition to get:

$$M_{dec_2} = \left(\begin{bmatrix} 0 & \delta_1 \\ 0 & 0 \\ \delta_1 & 0 \end{bmatrix} + \begin{bmatrix} \delta_3 & 0 \\ \delta_3 & \delta_3^2 \\ \delta_2\delta_3 & \delta_3^3 \end{bmatrix} \right) \begin{bmatrix} \delta_3 & 0 \\ 0 & 1 \end{bmatrix}$$

The latter can be further manipulated (columns and row affine factorization of δ_3) to obtain a final total $\sigma = 8$. This shows that by postponing affine factorizations along a particular direction and using sum decomposition first, it might be possible to achieve larger reductions. □

The parameter ordering, 'SIGMA-ORDERING', and the matrix preparation steps depend on the symbolic operation selected by the IM:

1. For **Horner factorization**, the ordering of the symbolic vector is given by the 'HORNER-ORDERING' step, and the matrix undergoes a 'polynomial substitution' step 'POLY-SUBSTITUTION'.
   The symbolic vector 'HORNER-ORDERING' is based on the possible reduction order red_{pos} for each parameter along each matrix dimension (i.e. each parameter results in two values: the sum along the rows and the sum along the columns). This ordering might not give the largest order reduction *red* for a specific coefficient but will result in better matrix σ reduction, as it will be shown later.
   In the 'POLY-SUBSTITUTION' step (used before applying affine factorizations), the IM performs a substitution of the sub-polynomials found in the matrix by dummy parameters. The sub-polynomials are the polynomial remainders (or quotients) obtained in the Horner factorization. These substitutions will ease and speed up the task of recognizing which parameters (symbolic and dummies) to affine factorize. After the affine factorization, the IM back-substitutes the sub-polynomials and expands the matrix to prepare it for the sum decomposition step. Furthermore, these last steps allow the IM to appropriately update the metrics for all the symbolic parameters.
2. For the **sum decomposition**, the IM forms a 'DECOMPOSITION-LIST' that contains information for the sum decomposition routine, which attempts maximizing the reduction in the total σ degree for subsequent affine factorizations. The list is ordered in decreasing possible reduction order red_{pos} and when equal, subclassified by σ degree (and if necessary finally by lexicographic order). Each row in the decomposition list is formed by the parameter number, its position (which row or column), the required metrics, and cells containing the indexes for the factorizable and non-factorizable coefficients along the specified row/column – these include summands in polynomial coefficients. This splitting of the matrix coefficients indicates to the sum decomposition routine which coefficients should be assigned to the primary matrix $M_{A_{ii}}$ (i.e. factorizable coefficients) or to the secondary matrix $M_{B_{ii}}$ (i.e. the non-factorizable). There is a 'CONFLICT-ANALYSIS' step related to the sum decomposition and detailed later.

After the factorization and the sum decomposition have been performed, the IM evaluates, using the 'FULLY-DEC' and 'NONFULLY-DEC' steps, if the primary matrix $M_{A_{ii}}$ can be further decomposed or if the first of the non-decomposed $M_{B_{jj\leq ii}}$ secondary matrices should be decomposed. The priority is to decompose fully the primary matrix first, and subsequently to start with the secondary matrices (there might be more than one, as each time the primary matrix is passed through the decomposition scheme it will generate a secondary matrix). Note that as the secondary matrices are selected for the decomposition they become primary, see Figure 1.

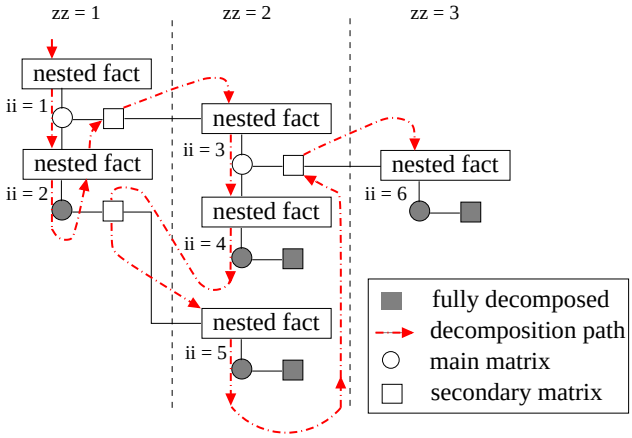


Fig. 1. Graph Logic Horner Tree (LHT) algorithm.

Factorization routine

This routine is composed of two main operations: 'HORNER-FACTORIZATION' and affine factorization 'AFF-FACTOR'. The former is always followed by the later and only occurs if there are polynomial coefficients.

The Horner factorization approach is based on the Horner simplification for polynomials. For the evaluation of a polynomial of degree k, it requires only k multiplications and k additions which is much less expensive than the number of multiplications for the expanded form [26]. It is also numerically more efficient and accurate. The general univariate case is given by: $\mathcal{P}(\delta) = a_n + \delta{\cdot}(a_{n-1} + \ldots \delta{\cdot}(a_1 + \delta{\cdot}a_o))$ For the multivariate case, an ordering of the parameters must be given. This ordering is not unique and will affect the nesting and the achievable σ reduction. Therefore, all the possible cases should be tested (for n parameters this means $n!$ which is very computationally expensive) unless some logic is used in the ordering.

A matrix extension of the polynomial Horner factorization is proposed based on the symbolic vector ordering given in the 'HORNER-ORDERING' step detailed before. In this manner, see example 2, the emphasis is placed on decreasing the reduction order *red* of the matrix and not for each individual polynomial coefficient:

Example 2. Given $M = \begin{bmatrix}\delta_1^3+\delta_1^3\delta_2^2\delta_3^2+\delta_2^2\delta_3^2 & \delta_2^2\delta_3^2\end{bmatrix}$*, assume it is desired to apply Horner factorization in the* $\{1,1\}$ *coefficient rather than sum decompositions (no affine factorization exists for M).*
If the maximal order reduction (per element) is used in $\{1,1\}$*, the correct element to factorize first is* δ_1 *and the resulting matrix is:*
$M1 = \begin{bmatrix}\delta_1^3(1+\delta_2^2\delta_3^2)+\delta_2^2\delta_3^2 & \delta_2^2\delta_3^2\end{bmatrix}$.
The logical next step is to sum decompose so that δ_1^3 *can be affine factorized (i.e. shift the monomial* $\delta_2^2\delta_3^2$ *in* $\{1,1\}$ *to another matrix, either together with* $\{1,2\}$ *or by itself). This yields a final total* $\sigma = 11$.
If Horner factorization is performed in M at system level (i.e. evaluating impact on $\{1,2\}$ *as well), the selected element is* δ_2^2 *or* δ_3^2 *(actually both but lets assume only one at a time), which yields:*
$M2 = \begin{bmatrix}\delta_2^2(\delta_1^3\delta_3^2+\delta_3^2)+\delta_1^3 & \delta_2^2\delta_3^2\end{bmatrix}$,
This can be sum decomposed pushing the monomial δ_1^3 *to the other summand-matrix, and using successive affine factorizations of* δ_2^2 *and* δ_3^2 *to get a final total* $\sigma = 10$. □

In the 'AFF-FACTOR' step, $M_{A_{ii}} = L_{ii}\bar{M}_{A_{ii}}R_{ii}$, parameter ordering is not important since there is no influence from the factorization of one parameter on the factorization of the others. Nevertheless, for each parameter it is important to identify the first direction along which to perform the factorization (i.e. left ≡ rows or right ≡ columns). Direction is important since, by virtue of the factorization nature, it is mutually exclusive (i.e. once factorization of a parameter along one dimension is performed, the possible order reduction red_{pos} for that parameter along the other direction is decreased). This 'AFFINE-DIRECTION' identification is based on the following classification logic (seven cases) which compares the total order reduction *red* and the total possible reduction red_{pos} along one direction with those along the other direction:

Cases 1-3: Factorize along rows :

$(1) red_{row} > red_{col} + red_{pos_{col}}$

$(2) red_{row} > red_{col}$ & $red_{pos_{row}} = red_{pos_{col}}$

$(3) red_{row} = red_{col} \neq 0$ & $red_{pos_{row}} > red_{pos_{col}}$

Cases 4-6: Factorize along columns :

$(4) red_{col} > red_{row} + red_{pos_{row}}$

$(5) red_{row} < red_{col}$ & $red_{pos_{row}} = red_{pos_{col}}$

$(6) red_{row} = red_{col} \neq 0$ & $red_{pos_{row}} < red_{pos_{col}}$

The 7^{th} case occurs when none of the previous cases is satisfied. In this case it will be very difficult and computationally expensive to ascertain along which direction to factorize. Hence, a 'preview' of the immediate effect the factorization along each direction has on the decomposition is performed (a 'preview' of only one step ahead is currently performed but this is a design decision). The classification logic is similar to the above but based on the 'future' affine factorizations along each direction: $M = Lf \cdot Mf_L$ (left) and $M = Rf \cdot Mf_R$ (right). This approach doubles the number of

calculations (as affine factorizations are performed in both directions and then only one is selected) but it maximizes the total order reduction *red* for the present and subsequent iteration.

Sum decomposition routine

The 'SUM-DECOMP' step decomposes the matrix $\bar{M}_{A_{ii}}$ into two summand-matrices, $\bar{M}_{A_{ii}} = M_{A_{ii+1}} + M_{B_{ii+1}}$, following the 'DECOMPOSITION-LIST' and using a conflict logic 'CONFLICT-ANALYSIS' to form the primary and secondary matrices.

The 'CONFLICT-ANALYSIS' operates as follows. For the first row in the list, the sum decomposition assigns the factorizable coefficients (for that parameter and specified matrix row or column) to $M_{A_{ii+1}}$ and the non-factorizable coefficients to $M_{B_{ii+1}}$. Subsequently, it moves through the list from top to bottom evaluating if there is any conflict, i.e. new factorizable coefficients already placed in $M_{B_{ii+1}}$ or non-factorizable coefficients in $M_{A_{ii+1}}$. If there is no conflict it distributes the new coefficients correspondingly and moves to the next row in the list. Otherwise, the possible reduction order red_{pos} for the parameter / position being evaluated is re-calculated after removing the conflicting coefficients. If the new red_{pos} is better or equal than that for the next row in the list, the sum decomposition is performed with the updated coefficients, otherwise the list is re-ordered to account for the new resulting row and the routine proceeds to the next row in the list, see the example:

Example 3. Given the matrix, $M = \begin{bmatrix} \delta_1^2 & \delta_1^3\delta_2^2 & \delta_2^2 \\ 0 & \delta_1\delta_2+\delta_3 & 0 \end{bmatrix}$*, after analyzing the possible reduction order* red_{pos} *along the rows and columns for the two symbolic parameters, the following decomposition list is obtained:*

symb.	**pos**	red_{pos}	**fact indx.**	**non-fact indx.**
δ_1	1st row	2	[1,2]	[3]
δ_2	1st row	2	[2,3]	[1]
δ_1	2nd col	1	[1,2(1)]	[2(2)]
δ_2	2nd col	1	[1,2(1)]	[2(2)]

After the first row in the list is used, the status of the sum decomposition is $M = M'_{A_1} + M'_{B_1}$*:*

$$\begin{bmatrix} \delta_1^2 & \delta_1^3\delta_2^2 & \delta_2^2 \\ 0 & \delta_1\delta_2+\delta_3 & 0 \end{bmatrix} = \begin{bmatrix} \delta_1^2 & \delta_1^3\delta_2^2 & 0 \\ * & * & * \end{bmatrix} + \begin{bmatrix} 0 & 0 & \delta_2^2 \\ * & * & * \end{bmatrix}$$

The second row in the list indicates the subsequent decomposition of the parameter δ_2 *along the first row of M. It is immediately noticed that one of the factorizable indexes conflicts with an already assigned coefficient in* M_{B_1}*. Hence, no sum-decomposition is performed for this parameter and the routine passes to the next row in the table which indicates that the* $\{1,2\}$ *coefficient and the 1st summand of*

the $\{2,2\}$ *go to* M'_{A_1}*:*

$$\begin{bmatrix} \delta_1^2 & \delta_1^3\delta_2^2 & \delta_2^2 \\ 0 & \delta_1\delta_2+\delta_3 & 0 \end{bmatrix} = \begin{bmatrix} \delta_1^2 & \delta_1^3\delta_2^2 & 0 \\ 0 & \delta_1\delta_2 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & \delta_2^2 \\ 0 & \delta_3 & 0 \end{bmatrix}$$

Now, performing an affine factorizations on M_{A_1} *yields a final total reduction order of* 8*, which is the best reduction achievable for* M*.* □

LFT modelling application

A paradigm shift in the modelling of dynamic systems occurred in the 1980's with the introduction of modern robust control theory and its associated modelling framework, the linear fractional transformation [20]. A LFT is generally used to represent an uncertain system using two operators in a linear feedback interconnection: $\mathcal{F}_U(M,\Delta) = M_{22} + M_{21}\Delta(I - M_{11}\Delta)^{-1}M_{21}$ where M is the nominal, known part and Δ represents the system's uncertainty, see Figure 2:

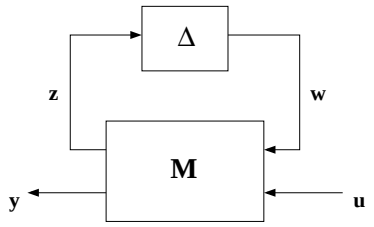


Fig. 2. Linear fractional transformation $LFT(M,\Delta)$.

In the case of parametric uncertainty (the case of interest in most aircraft applications), the resulting structure is $\Delta = diag(\delta_1 I_1,\ \delta_2 I_2, \ldots \delta_n I_n)$ where I_i represents an identity matrix of dimension equal to the number of repetitions of the i^{th} parameter. The order of the LFT is then said to be the dimension of Δ, and is an important consideration for the control synthesis and analysis methods currently available in robust control. Many realistic robustness analysis problems, for example, easily result in very high order LFTs and therefore it is vital to have efficient (and automated) tools which can compute minimal, or at least close to minimal, representations of these systems.

It is noted that the order of an LFT derived from a symbolic expression (where the symbolic parameters are considered uncertain parameters) is equal to the total presence degree σ of the expression [16]. Furthermore, a very important property of LFT systems is that their interconnection results in another LFT.

The extension of the LHT decomposition algorithm to LFT modelling is direct using the previous property and, in particular, the symbolic 'object-oriented' realization of LFTs as proposed in [16]. The 'object-oriented' realization takes the point of view that LFTs are feedback interconnections of matrices and as such are subject to standard matrix operations: addition and multiplication of matrices are equivalent to

parallel and series connection of LFTs (see the formulae given in [14, 16]). Furthermore, the order of the uncertainty matrix Δ for the resulting LFT is equal to the sum of the orders of the individual uncertainty matrices.

Hence, each of the individual matrices resulting from the proposed algorithm, see equation 2, can be transformed into a LFT with a diagonal uncertainty matrix of order equal to the respective matrix total presence degree. The thus obtained LFTs can be manipulated, following the resulting structure of the decomposition (2), to obtain a final LFT whose uncertainty matrix Δ is a diagonal matrix of order equal to the sum of the total presence degrees.

The applicability of this algorithm to dynamical systems is straight-forward recalling that a LFT is basically a generalization of the notion of state-space where the dynamic system is written as a feedback interconnection of a constant matrix and a diagonal element containing the integrator terms '$1/s$' and delays [2]. This is valid as well for exact nonlinear modelling, developed in Section 3, where the non-linearities are considered as symbolic parameters to be 'pulled out' into the Δ matrix.

LHT benchmarking

In reference [4] a comparison of other LFT algorithms is given for three benchmark examples:

1. A model of an F16 near a stall bifurcation with 9 states, 5 control inputs and 10 outputs that can be scheduled using a mix of airspeed V and flight path angle γ.
2. The longitudinal motion of a missile [21], a 2 degrees of freedom with one input and scheduled on angle of attack α and Mach M.
3. A generic physics-based model (no more details are included) given by a polynomial of matrices each multiplied by a monomial formed by three uncertain parameters (x, y, z) of different powers.

Specifically, the compared algorithms are the symtreed [8] implementation found in ONERAs LFR toolbox [16], the NASAs numerical approximation from [5] and the LFT algorithm available in the new MATLAB robust control toolbox [1].

The benchmarking of the LHT algorithm is given in Table 1 for the case of the direct application of the algorithms, i.e. no application of multidimensional reduction techniques (except the flag 'basic-reduction' for the MATLAB toolbox -otherwise the comparison will be unfair) [3].

From the above comparison it is observed that the LHT algorithm performs much better and actually seems to attempt minimizing the standard deviation between parameters. The latter characteristic is actually a wonderful advantage when there is no information about which parameter is more critical. For example, for the F16 model if it happens that the parameter V can be removed later on, then the effect on the relative order of the LHT algorithm is larger since both parameters have closer values than for the other methods.

[3] For the LHT algorithm, an initial pre-processing of the system matrices using the variable-splitting operation from [13] has been performed due to the particular structure of the examples (without this pre-processing similar values as the ONERA toolbox where obtained).

	F16			Missile			Generic			
	$\sum$	V	γ	$\sum$	α	M	$\sum$	x	y	z
NASA	86	31	55	12	4	8	94	9	64	21
ONERA	72	24	48	11	6	5	110	9	24	77
MATLAB	235	106	129	16	8	8	579	268	173	138
LHT	57	24	33	9	4	5	46	18	15	13

Table 1. LFT modelling benchmarking: No reduction cases

If 1D or n-D reduction methods [9], and available in the ONERAs toolbox, ([1] has its own methods) are applied, the results obtained are shown in Table 2:

	F16			Missile			Generic			
	$\sum$	V	γ	$\sum$	α	M	$\sum$	x	y	z
NASA	55	22	33	9	4	5	45	9	24	12
ONERA	56	23	33	9	4	5	46	9	24	13
MATLAB	53	22	31	14	8	6	45	9	24	12
LHT	56	23	33	8	4	4	45	18	15	12

Table 2. LFT modelling benchmarking: best of 1D or n-D reduction methods

Again, it is noted that the LHT algorithm has equivalent or better performance than the other algorithms but with the additional advantage of minimizing at the same time the standard deviation (if it is preferred to emphasize reduction of one parameter over the others that is also possible in the LHT algorithm). Also, it is remarkable that the effect of the multidimensional reduction approaches is minimal on the LHT (i.e. the difference before and after their application is negligible in comparison to the other techniques).

Finally, it is noted that similarities in the performance of the algorithms is only one side of the problem, since the question of which model is better for design or analysis is determined only partially by the total number of parameters (indeed, this is an open problem).

2.2 Symbolic LFT differentiation & nested LFT substitution operations

The LFT extension of the LHT algorithm has been shown to rely on standard LFT operations such as concatenation, addition and multiplication to name a few [14, 16]). Together with these standard operations, the modelling framework proposed in Section 3 will require also of two special operations: symbolic LFT differentiation and nested LFT substitution.

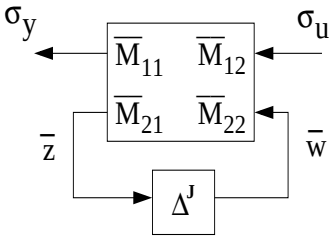
The first operation enables connecting the modelling framework with linear modelling, design and analysis techniques which are based on linear time invariant (LTI) state-space models. The second operation, and due to the diagonal structure of the Δ matrix, will allow substituting symbolic parameters by approximations or more detailed descriptions (all given also in LFT format) with minor modelling effort.

The lemmas focus on lower LFT representation but they can be straight-forwardly adapted for upper LFTs.

Symbolic LFT differentiation

This first LFT operation is based on the LFT differentiation idea found in [16] but adds an additional step which allows for a more complete automatization of the procedure. Indeed, the operation can be viewed as a three-step approach where in the first step a LFT is formed for the system under consideration using as inputs u both the system inputs d and also the symbolic parameters ρ. The second step is in charge of symbolically differentiating with respect to u the the part of the LFT containing the Δ matrix operator. And the third step performs a LFT of the expression from the second step and combines the resulting matrices with those from the first step.

Lemma 1. *Consider a symbolic well-posed lower LFT $y = \mathcal{F}_l(M,\Delta)u$ where $M = [M_{11}\ M_{12};M_{21}\ M_{22}]$, $\Delta = diag(\Delta_1,\Delta_2(\rho))$ and $u = [\rho\ \ d]^\top$. Its symbolic lower LFT linearization, see Figure 3, is given by $\sigma_y = \mathcal{F}_l(\bar{M},\Delta^J)\sigma_u$ where σ_y,σ_u are deviation variables with respect to an equilibrium point (y_{eq},u_{eq}), e.g. $\sigma_y = y - y_{eq}$; the coefficient matrix $\bar{M}$ is given by equation (3):*



$$\bar{M} = \begin{bmatrix} M_{11} + M_{12}M_{11}^J & M_{12}M_{12}^J \\ M_{21}^J & M_{22}^J \end{bmatrix} \tag{3}$$

Fig. 3. Symbolic linearized LFT.

The matrices $M^J = [M_{11}^J\ M_{12}^J;M_{21}^J\ M_{22}^J]$ and Δ^J are respectively the coefficient and uncertain components from the lower LFT of $\mathcal{L} = \mathcal{F}_l(M^J,\Delta^J)$, and $\mathcal{L}$ is given by:

$$\mathcal{L} = \frac{\partial\Big((I - \Delta M_{22})^{-1}\Delta M_{21}u\Big)}{\partial u}\Big|_{eq} \tag{4}$$

where Δ^J is obtained by selecting as symbolic variables the terms $\Delta_1|_{eq}$, $\Delta_2(\rho)|_{eq}$, u_{eq} and the symbolic derivative $\frac{\partial\Delta_2(\rho)}{\partial u}\Big|_{eq}$.

Proof: The proof is given in Appendix B. It represents an algorithmic implementation for the LFT operation.

Example 4. The following example based on the chemical reactor model from reference [22], page 287, illustrates the symbolic LFT linearization approach.
The component balance model of a chemical reactor (CSTR) is given by:

$$\dot{c}_A = \frac{c_{A_f} q}{V} - \frac{c_A q}{V} - k_1 c_A$$
$$\dot{c}_B = -\frac{c_B q}{V} + k_1 c_A - k_2 c_B$$

where q is the feed-flow rate, c_A is the concentration of reactant A, c_B is the concentration of the intermediate product B and V is the reactor volume. k_1 and k_2 are constant steady-state values for the feed-flow rate and c_{A_f} is the final concentration for the reactant A.
Introducing deviation variables $\sigma_{c_A} = c_A - c_{A_{eq}}$, $\sigma_{c_B} = c_B - c_{B_{eq}}$, $\sigma_q = q - q_{eq}$ and symbolically linearizing the original nonlinear system yields:

$$\dot{\sigma}_{c_A} = -(k_1 + \frac{q_{eq}}{V_{eq}})\sigma_{c_A} + (\frac{c_{A_{f\,eq}}}{V_{eq}} - \frac{c_{Aeq}}{V_{eq}})\sigma_q$$
$$\dot{\sigma}_{c_B} = k_1\sigma_{c_A} - (k_2 + \frac{q_{eq}}{V_{eq}})\sigma_{c_B} - \frac{c_{Beq}}{V_{eq}}\sigma_q$$

Using the symbolic LFT approach on the linearized equations, a lower symbolic LFT for the linearized system is obtained, see Figure 4 (note that this step is automatic using the LHT algorithm):

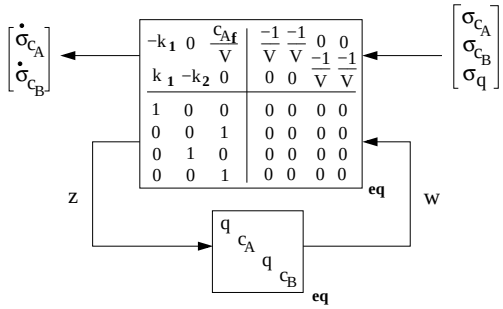


Fig. 4. Example: symbolic LFT for linearized CSTR.

Alternatively, using the proposed nonlinear symbolic LFT approach, a lower nonlinear symbolic LFT can be obtained for the original nonlinear system, see Figure 5:

Applying the symbolic linearization LFT operation from Lemma 1 to this nonlinear symbolic lower LFT, the input-output mapping from equation (16) is obtained:

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = (I - \Delta M_{22})^{-1}\Delta M_{21} u = \left(I - \begin{bmatrix} c_A & 0 \\ 0 & c_B \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\right)^{-1}\begin{bmatrix} c_A & 0 \\ 0 & c_B \end{bmatrix}\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} c_A \\ c_B \\ q \end{bmatrix} = \begin{bmatrix} c_A q \\ c_B q \end{bmatrix}$$

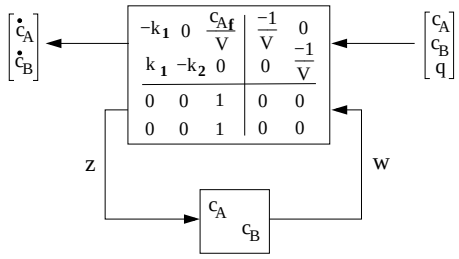


Fig. 5. Example: symbolic LFT for nonlinear CSTR.

Performing a symbolic partial derivative on this input-output mapping with respect to $u = [c_A, c_B, q]$ *yields the term* $\mathcal{L}$*:*

$$\begin{bmatrix}\sigma_{w_1}\\ \sigma_{w_2}\end{bmatrix} = \mathcal{L}\,\sigma_u == \begin{bmatrix}q & 0 & c_A\\ 0 & q & c_B\end{bmatrix}\Big|_{eq}\begin{bmatrix}\sigma_{c_A}\\ \sigma_{c_B}\\ \sigma_q\end{bmatrix}$$

A lower symbolic LFT for $\mathcal{L}$*, i.e.* $\mathcal{F}_l(M^J, \Delta^J)$*, can be obtained assuming* c_{Aeq}*,* c_{Beq}*,* q_{eq} *are the symbolic variables to be "pulled out", see Figure 6:*

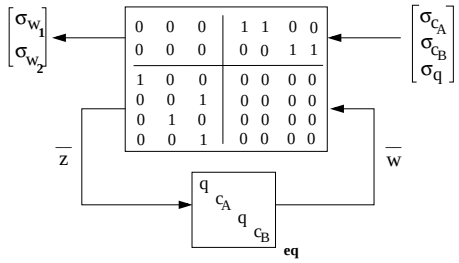


Fig. 6. Example: symbolic LFT for CSTR $\mathcal{L}$ function.

Finally, using the coefficient matrices from Figures 5 and Figure 6 together with the formulae from Lemma 1, the coefficient matrix $\bar{M}$ *is obtained:*

$$\bar{M}_{11} = (M_{11} + M_{12}M_{11}^J)|_{eq} = \begin{bmatrix}-k_1 & 0 & \frac{c_{A_f}}{V}\\ k_1 & -k_2 & 0\end{bmatrix} + \begin{bmatrix}-\frac{1}{V} & 0\\ 0 & -\frac{1}{V}\end{bmatrix}\begin{bmatrix}0&0&0\\0&0&0\end{bmatrix} = \begin{bmatrix}-k_1 & 0 & \frac{c_{A_f}}{V}\\ k_1 & -k_2 & 0\end{bmatrix}\Big|_{eq}$$

$$\bar{M}_{12} = (M_{12}M_{12}^J)|_{eq} = \begin{bmatrix}-\frac{1}{V} & 0\\ 0 & -\frac{1}{V}\end{bmatrix}\begin{bmatrix}1&1&0&0\\0&0&1&1\end{bmatrix} = \begin{bmatrix}-\frac{1}{V} & -\frac{1}{V} & 0 & 0\\ 0 & 0 & -\frac{1}{V} & -\frac{1}{V}\end{bmatrix}\Big|_{eq}$$

$$\bar{M}_{21} = M_{21}^J = \begin{bmatrix}1&0&0\\0&0&1\\0&1&0\\0&0&1\end{bmatrix}$$

$$\bar{M}_{22} = M_{22}^J = 0_{4\times 4}$$

The above coefficient matrix together the uncertain matrix Δ^J from Figure 6 yield a lower symbolic LFT $\mathcal{F}_l(\bar{M},\Delta^J)$ which is the same as that from Figure 4. □

In order to use linear analysis and design techniques, the numerical form of the linear symbolic LFT must be used. Actually, the latter form is only necessary for the final constant matrix M_p, and for those terms in Δ_p not required to be symbolic (e.g. some of the parameters might be left symbolic for robust control synthesis or worst-case μ-analysis). As all the symbolic parameters are parameterized by (or independent of) the general equilibrium point (y_{eq}, u_{eq}), a simple numerical substitution of the chosen equilibrium point suffices to find the required linear system model.

Nested LFT substitution

This second operation has the aim of facilitating substitution of a (symbolic) parameter in the Δ matrix operator by another LFT that represents either an approximation or a more detailed expression for that parameter. The term 'nested' refers to the generalization of the substitution operation whereby subsequent substitutions can be performed recursively in the newly obtained Δs.

Lemma 2. *Consider a lower LFT, $y = \mathcal{F}_l(M,\Delta(\rho))u$, where $M = [M_{11}\, M_{12}; M_{21}\, M_{22}]$ and $\Delta(\rho) = diag(\Delta_1(\rho),\Delta_2(\rho))$, as shown in Figure 7 (a).*

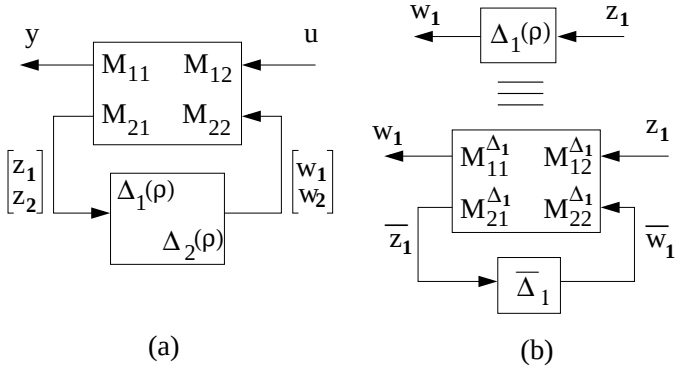


Fig. 7. Nested LFT: initial lower LFTs.

Assume $w_1 = \Delta_1(\rho)z_1$ can be represented as another LFT, $\Delta_1(\rho) = \mathcal{F}_l(M^{\Delta_1},\bar{\Delta}_1)$ where $M^{\Delta_1} = [M_{11}^{\Delta_1}\, M_{12}^{\Delta_1}; M_{21}^{\Delta_1}\, M_{22}^{\Delta_1}]$, as shown in Figure 7 (b).
The nested substitution *of the $\Delta_1(\rho)$ lower LFT into $\mathcal{F}_l(M,\Delta(\rho))$ yields another lower LFT $\mathcal{F}_l(\bar{M},\bar{\Delta})$, of order equal to the sum of the orders for Δ_1 and Δ_2, defined by:*

$$\bar{\Delta} = \begin{bmatrix} \bar{\Delta}_1 & 0 \\ 0 & \Delta_2(\rho) \end{bmatrix} \tag{5}$$

$$\bar{M} = \begin{bmatrix} \mathcal{F}_l(M, \bar{M}_{11}^{\Delta}) & M_{12}(I - \bar{M}_{11}^{\Delta} M_{22})^{-1} \bar{M}_{12}^{\Delta} \\ \bar{M}_{21}^{\Delta}(I - M_{22}\bar{M}_{11}^{\Delta})^{-1} M_{21} & \mathcal{F}_u(\bar{M}^{\Delta}, M_{22}) \end{bmatrix} \tag{6}$$

$$\bar{M}^{\Delta} = \begin{bmatrix} \bar{M}_{11}^{\Delta} & \bar{M}_{12}^{\Delta} \\ \bar{M}_{21}^{\Delta} & \bar{M}_{22}^{\Delta} \end{bmatrix} = \begin{bmatrix} 0_{dim(w)\times dim(z)} & I_{dim(w)\times dim(\bar{w}_1+w_2)} \\ I_{dim(\bar{z}_1+z_2)\times dim(z)} & 0_{dim(\bar{z}_1+z_2)\times dim(\bar{w}_1+w_2)} \end{bmatrix} \tag{7}$$

where $\bar{M}^{\Delta}$ is composed of zero and identity matrices with elements $\bar{M}_{11}^{\Delta}(ii,ii) = M_{11}^{\Delta_1}$, $\bar{M}_{12}^{\Delta}(ii,ii) = M_{12}^{\Delta_1}$, $\bar{M}_{21}^{\Delta}(ii,ii) = M_{21}^{\Delta_1}$, $\bar{M}_{22}^{\Delta}(ii,ii) = M_{22}^{\Delta_1}$. The index (ii,ii) is given by the position of $\Delta_1(\rho)$ in $\Delta(\rho)$.

Proof: The proof, which represents as well an algorithmic implementation of the operation, is given in Appendix C.

Example 5. Assume that the angle of attack aircraft equation of motion is given in nonlinear symbolic form by (see example 6 for details on the symbolization):

$$\dot{\alpha} = q - \frac{\overline{q}\, S}{m\, V_{TAS}} C_L + \frac{g}{V_{TAS}} (s_\alpha s_\theta + c_\theta c_\alpha) = q - c_1\rho_8\rho_9 + \rho_8\rho_{10} \tag{8}$$

Furthermore, assume that its corresponding symbolic LFT is $\mathcal{F}_l(M,\Delta)$ where M:

$$M_{11} = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad M_{12} = \begin{bmatrix} -c_1 & 1 & 0 & 0 \end{bmatrix}$$

$$M_{21} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \qquad M_{22} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

and $w = \Delta z = diag(\rho_8, \rho_8, \rho_9, \rho_{10})z$ with $w = [w_1\ w_2\ w_3\ w_4]^\top$, $z = [z_1\ z_2\ z_3\ z_4]^\top$. Now assume the symbolic parameter in $w_4 = \rho_{10} z_4$ is approximated by $\rho_{10} = c_2(\rho_4\rho_6 + \rho_5\rho_7)$ which is represented as the symbolic LFT $w_4 = \mathcal{F}_l(M^{\Delta_1}, \bar{\Delta}_1) z_4$, see Figure 8:

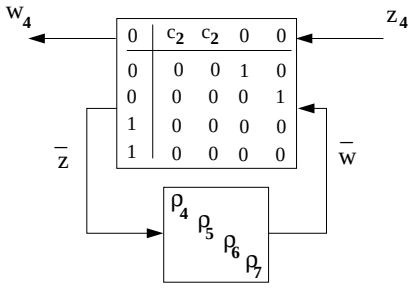


Fig. 8. Example symbolic substitution: $w_4 = \rho_{10} z_4 = \mathcal{F}_l(M^{\Delta_1}, \bar{\Delta}_1) z_4$.

Following the proof of Lemma 2, the previous LFT $\mathcal{F}_l(M^{\Delta_1}, \bar{\Delta}_1)$ is augmented to contain the input-output mappings $w_1 = \rho_8 z_1$, $w_2 = \rho_8 z_2$ and $w_3 = \rho_9 z_3$, yielding another symbolic lower LFT $w = \mathcal{F}_l(\bar{M}^{\Delta}, \Delta)$, see Figure 9:

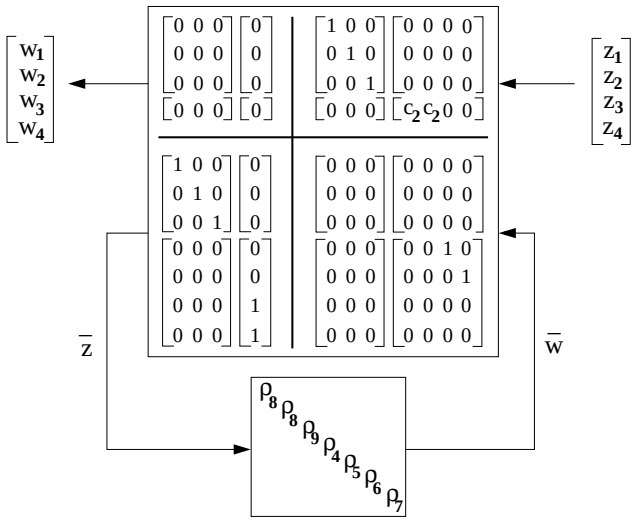


Fig. 9. Example symbolic substitution: $w = \mathcal{F}_l(\bar{M}^{\Delta}, \bar{\Delta})z$.

Finally, using the the coefficient matrices M and $\bar{M}^{\Delta}$ together with equations (24-28), the final coefficient matrix $\bar{M}$ is obtained:

$$\bar{M}_{11} = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad \bar{M}_{12} = \begin{bmatrix} -c_1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\bar{M}_{21} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \qquad \bar{M}_{22} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The uncertain matrix $\bar{\Delta}$ is the same as that shown in Figure 9. This final LFT $\mathcal{F}_l(\bar{M}, \bar{\Delta})$ can be easily shown to be the lower symbolic LFT corresponding to the angle of attack equation $\dot{\alpha} = q - c_1\rho_9\rho_8 + c_2\rho_8(\rho_4\rho_6 + \rho_5\rho_7)$.
Note that the same procedure could also be followed in order to substitute $\rho_9 = \rho_1\rho_2\rho_3$ into the LFT. Furthermore, due to the LFT property that interconnections of LFTs yield another LFT, the same procedure could be followed for the pitch rate equation and the resulting LFT added to that for the angle of attack to yield the complete aircraft short-period motion model in LFT form. □

3 Nonlinear symbolic LFT modelling approach

The basic idea of the proposed modelling approach is to combine symbolic and LFT tools to represent the complex ordinary differential equations (ODEs), which define the nonlinear system. In this manner, an exact nonlinear symbolic LFT where the structured Δ matrix contains the nonlinear, time-varying or uncertain terms as symbolic parameters can be obtained. Once this exact nonlinear symbolic LFT is obtained, it is more straightforward and flexible to simplify the model down to a manageable size while retaining sufficient precision for successful control design.

Indeed, an advantage of the proposed modelling approach is that it results in a highly structured representation of the nonlinearities, which facilitates their analysis and ameliorates the effect that inappropriate simplifications and approximations might have on the overall modelling process. This advantage arises due to the LFT nature of the framework together with the diagonal structure of the symbolic nonlinearities in $\Delta(\rho)$.

Furthermore, once the control system is synthesized using the simplified LFT model it can be gradually validated using increasingly complex models up to the full nonlinear model given by the exact nonlinear symbolic LFT. This gradual validation has the advantage of possibilitating the identification of the problematic model terms or controller shortcomings in greater detail, providing specific feedback if re-design of the controller is necessary (while automatically updating the model if required).

Class of nonlinear systems considered

It is assumed that the class of nonlinear systems is defined as follows:

$$\dot{x} = f(x,u) = f_1(x)\,x + f_2(x)\,u + f_3(x) \tag{9}$$

$$y = g(x,u) = g_1(x)\,x + g_2(x)\,u + g_3(x) \tag{10}$$

where the nonlinear functions $f_i(x), g_i(x)$ are given by a polynomial mix of analytic expressions and tabular data. The first-order derivative condition for the states is without loss of generality (higher-order derivatives can be substituted by new state variables to transform the system into a first-order).

The main structural restriction for this class of systems is the linear dependency of the nonlinear functions on the input vector u, e.g. $f(x,u)$ is a function of $f_1(x)x$, $f_2(x)u$ and $f_3(x)$. This assumption is indeed quite general and standard for mechanical systems, nevertheless an extension to systems with nonlinear dependency on the inputs can also be considered [18]. The inclusion of the functions $f_3(x)$, $g_3(x)$ which represent those terms (nonlinear, time-varying or constant) that cannot be represented as linear in the states, significantly expands the set of nonlinear systems that can be considered. For example, these extra functions often arise in aerospace systems, where their consideration is critical [23].

Exact symbolic LFT modelling approach

It is highlighted that although seemingly trivial, the proposed approach has not been possible until recent development of specialized symbolic algebra software, as the LHT algorithm or those in [16, 13], and the previous LFT tools –and as such it represents a novel and powerful systematic modelling methodology. The basic steps of the modelling approach are:

1. Represent the nonlinear ODEs as a nonlinear state-space 2×2 block matrix using, if needed, fictitious signals $u_f = 1\ \forall\, t$ to include those nonlinear terms not affine on the inputs u or the states x:

$$\begin{bmatrix}\dot{x}\\ y\end{bmatrix} = \left[\begin{array}{c|cc} f_1(x) & f_2(x) & f_3(x)\\ \hline g_1(x) & g_2(x) & g_3(x)\end{array}\right]\begin{bmatrix}x\\ u\\ u_f\end{bmatrix} \tag{11}$$

2. Declare as symbolic parameters ρ_k all the uncertain, nonlinear or time-varying terms (including physical parameters that vary with operational condition, e.g. discrete switches). The guiding principle proposed at this stage is *to select everything that is not a known constant c_j as a symbolic parameter ρ_k*:

$$f_i(x) = f_i(\rho_1^{n1},\ldots,\rho_k^{nk},c_1,\ldots,c_j) \tag{12}$$

   where $n1,n2,\ldots,nk$ indicate the number of repetitions for each parameter.
3. Transform the resulting nonlinear symbolic 2×2 matrix into a nonlinear symbolic LFT where all the symbolic parameters (and their repetitions) are placed in the $\Delta(\rho)$ matrix. This step is automatically carried out using available algebraic LFT modelling software as the proposed LHT algorithm of those found in [16, 13].

   Remark: *Note that the three steps above result in an LFT representation which is identical to the original nonlinear system given by equations (9-10).*

4. Taking advantage of the diagonal structure of $\Delta(\rho)$ arising from the previous LFT modelling process it is possible now to carry out [18]: *i*) simplifications, *ii*) model reduction, *iii*) approximations, and *iv*) uncertainty characterization, in order to obtain a manageable LFT model for design and analysis. Furthermore, the modularity afforded by the symbolic LFT allows easy updating of the model if any assumption needs to be corrected.

Example 6. Assume that the nonlinear angle of attack equation is given by the compact nonlinear equation:

$$\dot{\alpha} = q - \frac{\bar{q}S}{mV_{TAS}}C_L + \frac{g}{V_{TAS}}(s_\alpha s_\theta + c_\alpha c_\theta)$$

The known constants are assumed to be the wing surface $c_1 = S$ and the Earth gravity $c_2 = g$; everything else is treated as a symbolic parameter: dynamic pressure $\rho_1 = \bar{q}$, lift coefficient $\rho_2 = C_L$, aircraft mass $\rho_3 = \frac{1}{m}$, and trigonometric relationships $\rho_4 =$

s_α, $\rho_5 = c_\alpha$, $\rho_6 = s_\theta$, $\rho_7 = c_\theta$. *The inverse of the true airspeed is also considered as a symbolic parameter* $\rho_8 = \frac{1}{V_{TAS}}$. *Hence, the nonlinear equation is transformed into:*

$$\dot{\alpha} = q - c_1\rho_1\rho_2\rho_3\rho_8 + c_2\rho_8(\rho_4\rho_6 + \rho_5\rho_7)$$

Now, for simplicity in the manipulations of subsequent examples, substitute $\rho_9 = \rho_1\rho_2\rho_3$ *and* $\rho_{10} = c_2(\rho_4\rho_6 + \rho_5\rho_7)$ *in the previous system. Introducing a fictitious input* δ_f, *the following nonlinear symbolic system is obtained:*

$$\dot{\alpha} = q - c_1\rho_9\rho_8\delta_f + \rho_8\rho_{10}\delta_f$$

The corresponding LFT obtained using the LHT algorithm is given in Figure 10:

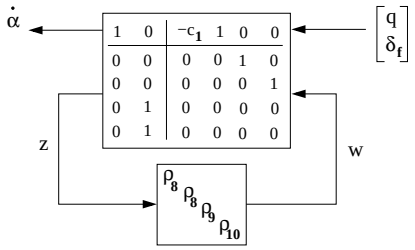


Fig. 10. Example: Nonlinear Symbolic LFT for an aircraft angle of attack.

The order of the symbolic nonlinear LFT based on the parameters ρ_8, ρ_9 *and* ρ_{10} *is 4 (two repeated parameters* ρ_8 *plus one for each of the other two parameters). Note, that if the substituted parameters* ρ_9 *and* ρ_{10} *are expanded, the order of the LFT will increase.* □

General practical considerations that should be considered when applying the proposed modelling approach are:

i. Initially, declare each physical parameter as symbolic and only group them as a new ρ if functional expression is complex, e.g. $\rho_1 = \sqrt{pq}$ but not $\rho_2 = pq$.
ii. The symbolic parameters are considered "independent" at this stage (e.g. $\rho_1 = s_\theta$, $\rho_2 = c_\theta$ and $\rho_3 = \theta$).
iii. The reciprocal of a parameter is considered also a parameter (e.g. $\rho_1 = m$ and $\rho_2 = m^{-1}$).
iv. Select carefully which parameter is extracted from a monomial.
v. In general, the selected exact LFT model is that with the lowest LFT order. This might not be appropriate if step 4 of the modelling approach is to be used afterwards.
vi. The symbolic parameters should be normalized only after all other manipulations have been performed on the symbolic LFT. By performing the normalization last, it is ensured that the physical meaning of the parameters is retained and

hence their effects on the system remain easier to understand and study. Furthermore, the diagonal structure of the Δ matrix means that the order of the LFT remains the same after normalization - see [17] for a flight dynamics example of the dramatic effect normalization of the parameters before the LFT process has on the overall LFT order. This problem has been mentioned before [8, 16], but it is noted that most of the available applications in the literature do not follow this guideline.

4 Conclusion

In this chapter a modelling framework based on symbolic algebra and LFT representations has been presented. Also, a novel LFT algorithm and the formalization of two additional LFT operations have been developed in support of the framework.

The proposed modelling approach might seem trivial but it has not been possible until recent development of specialized symbolic algebra algorithms, such as the LHT, and software implementation of LFT operations, such as the tools from [16]. It results in an exact nonlinear symbolic LFT that represents an ideal starting point to apply subsequent assumptions and simplifications to finally transform the model into an approximated symbolic LFT ready for design and analysis.

A clear advantage of the proposed modelling approach is that it results in a highly structured representation of the nonlinearities present in the system, which facilitates their analysis and ameliorates the effect that inappropriate simplifications and approximations might have on the overall modelling process. Furthermore, the approach allows to obtain a connected set of increasingly-simplified models that can be used to gradually validated the control design from the simplest of the models to the exact nonlinear symbolic LFT. This gradual validation has the advantage of possibilitating the identification of the problematic model terms or controller shortcomings in greater detail, providing specific feedback if re-design of the controller is necessary.

References

1. Balas, G.J., Chiang, R., Packard, A., and Safonov, M. Robust control toolbox v3.0, 2006.
2. Balas, G.J., Doyle, J.C., Glover, K., Packard, A., and Smith, R. μ-Analysis and Synthesis Toolbox, June 1998.
3. Becker, G. and Packard, A. Robust Performance of Linear Parametrically Varying Systems Using Parametrically Dependent Linear Dynamic Feedback. *Systems and Control Letters*, 23(3):205–215, 1994.
4. Belcastro, C., Khong, T.H., Shin. J.Y., Kwatny, H., Chang, B.C., and Balas, G. Uncertainty Modeling for Robustness Analysis of Aircraft Control Upset Prevention and Recovery Systems. In *AIAA Guidance, Navigation, and Control Conference*, San Francisco, CA, August 2005.
5. Belcastro, C.M. and Chang, B.C. On parametric uncertainty modeling for real parameter variations. In *IEEE Conference on Decision and Control*, December 1992.

6. Belcastro, C.M. and Chang, B.C. LFT Formulation for Multivariable Polynomial Problems. In *American Control Conference*, pages 1002–1007, Philadelphia, PA, June 1998.
7. Cockburn, J.C. Multidimensional realizations of systems with parametric uncertainty. In *Mathematical Theory of Networks and Systems*, Perpignan, France, June 2000.
8. Cockburn, J.C. and Morton, B.G. Linear Fractional Representations of Uncertain Systems. *Automatica*, 33(7):1263–1271, 1997.
9. D'andrea, R. and Khatri, S. Kalman decomposition of linear fractional transformation representations and minimality. In *American Control Conference*, pages 3557–3561, Alburquerque, NM, June 1997.
10. Duprez, J., Mora-Camino, F., and Villaume, F. Aircraft-on-ground lateral control for low speed manœuvers. In *16th IFAC symposium on automatic control in aerospace*, St. Petersburg, Russia, June 2004.
11. Fielding, C., Varga, A., Bennani, S., and Selier, M., editors. *Advanced Techniques for Clearance of Flight Control Laws*. Number 283 in Lecture Notes in Control and Information Sciences. Springer Verlag, 2002.
12. Heck, A. *Introduction to Maple*. Springer-Verlag, 1993.
13. Hecker, S., Varga, A., and Magni, J.F. Enhanced LFR-Toolbox for Matlab. *Aerospace Science and Technology*, 9(2), 2005.
14. Lambrechts, P., Terlouw, J., Bennani, S., and Steinbuch, M. Parametric Uncertainty Modeling using LFTs. In *American Control Conference*, pages 267–272, San Franciso, CA, June 1993.
15. Leith, D.J. and Leithead, W.E. Survey of Gain-Scheduling analysis and Design. *International Journal of Control*, 73(11):1001–1025, 2000.
16. Magni, J.F. Linear Fractional Representation Toolbox Modelling, Order Reduction, Gain Scheduling. Technical Report TR 6/08162 DCSD, ONERA, Systems Control and Flight Dynamics Department, Toulouse, France, January 2004.
17. Marcos, A., Bates, D.G., and Postlethwaite, I. Flight Dynamics Application of a New Symbolic Matrix Order-Reduction Algorithm. In *International Conference on Polynomial Symbolic Systems*, Paris, FR, November 2004.
18. Marcos, A., Bates, D.G., and Postlethwaite, I. Exact Nonlinear Modeling using Symbolic Linear Fractional Transformations. In *IFAC World Congress*, Praga, CH, June 2005.
19. Megretski, A. and Rantzer, A. System analysis via integral quadratic constraints: Part i. Technical Report LUTD2/TFRT-7531-SE, Department of Automatic Control, Lund Institute of Technology, April 1995.
20. Packard, A. and Doyle, J. The complex structured singular value. *Automatica*, 29(1):71–109, 1993.
21. Shamma, F. and Athans, M. Gain scheduling: Potential hazards and possible remedies. *IEEE Control Systems*, pages 101–107, June 1992.
22. Skogestad, S. and Postlethwaite, I. *Multivariable Feedback Control Analysis and Design*. Wiley, May 1996.
23. Stevens, B. and Lewis, F. *Aircraft Control and Simulation*. John Wiley & Sons, Inc., 2 edition, 1992.
24. Varga, A. and Looye, G. Symbolic and Numerical Software tools for LFT-based Low Order Uncertainty Modeling. In *IEEE Symposium on Computed Aided Control System Design*, Hawai'i, USA, August 1999.
25. Varga, A., Looye, G., Moormann, G., and Grubel, G. Automated Generation of LFT-Based Parametric Uncertainty Descriptions from Generic Aircraft Models. *Mathematical and Computer Modelling of Dynamical Systems*, 4(4):249–274, 1998.
26. Wolfram, S. *Mathematica:* a System for Doing Mathematics by Computer. Addison-Wesley, 1991.

Appendixes

A . LHT algorithm pseudo-code

The pseudo-code for the LHT algorithm implementation is given here. The description for each of the steps shown can be found in Section 2.1.

LHT($M(\delta)$)

```
1  i ← 1;  ok ← 1
2  while ok = 1
3    do M ← EXPAND(M)
4    if i = 1
5      then MA[1] ← SYMBOLIC-COEFF(M)
6           MB[1] ← CONSTANT-COEFF(M)
7    metrics ← METRICS(MA[i])
8    ordvect ← SIGMA-ORDERING(MA[i])
9    if EXIST-POLYCOEFF(MA[i]) = yes
10     then Hlist ← HORNER-ORDERING(MA[i], metrics)
11          MHA ← HORNER-FACTORIZATION(MA[i], Hlist)
12          MA[i], polylist ← POLY-SUBSTITUTION(MHA)
13          metrics ← METRICS(MA[i])
14          ordvect ← SIGMA-ORDERING(MA[i])
15   for j ← 1 to LENGHT[ordvect]
16     affdir ← AFFINE-DIRECTION(MA[i], metrics, ordvect[j])
17        ▷ if affdir = 0 ⇒ skip parameter affine factorization
18     L[i], MA[i], R[i] ← AFF-FACTOR(MA[i], affdir, ordvect[j])
19        ▷ short-hand: LMAR[i] ≡ L[i], MA[i], R[i]
20   if NOEMPTY(polylist)
21     then LMAR[i] ← POLY-BACKSUBS(LMAR[i], polylist)
22          LMAR[i] ← EXPAND(LMAR[i])
23          polylist ← empty
24   declist ← DECOMPOSITION-LIST(MA[i])
25   MA[i+1], MB[i+1] ← SUM-DECOMP(MA[i], [], declist[1])
26      ▷ short-hand: MAMB[i+1] ≡ MA[i+1], MB[i+1]
27   j ← 1
28   while NOEMPTY(declist)
29     do j ← j+1
30     declist ← CONFLICT-ANALYSIS(MAMB[i+1], declist[j])
31     MAMB[i+1] ← SUM-DECOMP(MAMB[i+1], declist[j])
32   M ← MA[i+1]
33   i ← i+1
34   if FULLY-DEC(M) and NONFULLY-DEC(MB[k])
35     then M ← MB[k]
36        ▷ k is special index to get correct MB
37   elseif FULLY-DEC(M) and FULLY-DEC(MB[k])
38     then ok ← 0
```

B . Symbolic jacobian LFT linearization

The following proof provides an algorithmic implementation of the symbolic LFT linearization operation from Lemma 1.

The input-output mappings for the symbolic well-posed lower LFT $y = \mathcal{F}_l(M,\Delta)u$ where $M = [M_{11}\ M_{12};M_{21}\ M_{22}]$, $\Delta = diag(\Delta_1,\Delta_2(\rho))$ and $u = [\rho\ \ d]^\top$, are:

$$y = M_{11}u + M_{12}w \tag{13}$$

$$z = M_{21}u + M_{22}w \tag{14}$$

$$w = \Delta z \tag{15}$$

Combining equations (14) and (15) yields:

$$w = (I - \Delta M_{22})^{-1}\Delta M_{21}u \tag{16}$$

which exists due to the well-posedness assumption on the LFT $y = \mathcal{F}_l(M,\Delta)u$.

Since the coefficient matrix terms M_{11}, M_{12}, M_{21}, M_{22} are constants, the symbolic first-order Taylor approximations with respect to a general equilibrium point (y_{eq},u_{eq}) are:

$$\sigma_y = M_{11}\sigma_u + M_{12}\sigma_w \tag{17}$$

$$\sigma_z = M_{21}\sigma_u + M_{22}\sigma_w \tag{18}$$

$$\sigma_w = \frac{\partial\Big((I - \Delta M_{22})^{-1}\Delta M_{21}u\Big)}{\partial u}\Big|_{eq} = \mathcal{L} \tag{19}$$

Obtain a symbolic lower LFT for $\mathcal{L}$, i.e. $\sigma_w = \mathcal{F}_l(M^J,\Delta^J)\sigma_u$ where $M^J = [M^J_{11}\ M^J_{12};M^J_{21}\ M^J_{22}]\Big|_{eq}$ is constant and Δ^J contains all the 'fixed' (at the general equilibrium point y_{eq}, u_{eq}) symbolic uncertain variables $\Delta_1|_{eq}$, $\Delta_2(\rho)|_{eq}$, u_{eq} and $\frac{\partial\Delta_2(\rho)}{\partial u}\Big|_{eq}$ in diagonal format, see Figure 11:

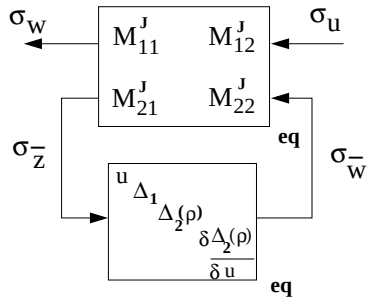


Fig. 11. Proof: symbolic linearized LFT for $\mathcal{L}$, $\sigma_w = \mathcal{F}_l(M^J,\Delta^J)\sigma_u$.

The corresponding input-output mappings of the LFT for $\mathcal{L}$ are:

$$\sigma_w = M_{11}^J \sigma_u + M_{12}^J \sigma_{\bar{w}} \tag{20}$$

$$\sigma_{\bar{z}} = M_{21}^J \sigma_u + M_{22}^J \sigma_{\bar{w}} \tag{21}$$

$$\sigma_{\bar{w}} = \Delta^J \sigma_{\bar{z}} \tag{22}$$

Substituting equation (20) into equation (17) yields:

$$\sigma_y = M_{11}\sigma_u + M_{12}(M_{11}^J \sigma_u + M_{12}^J \sigma_{\bar{w}}) = (M_{11} + M_{12}M_{11}^J)\sigma_u + M_{12}M_{12}^J \sigma_{\bar{w}} \tag{23}$$

Combining with equations (21-22), a new lower symbolic LFT $\sigma_y = \mathcal{F}_l(\bar{M}, \Delta^J)\sigma_u$ is obtained which corresponds to the symbolic lower LFT for the symbolic linearization of equations (13-16). The coefficient matrix $\bar{M}$ corresponds to the formulae from equation (3) and the uncertain matrix Δ^J is defined above.

C . Nested LFT substitution

As before, the following proof provides the algorithmic implementation for the nested LFT substitution of Lemma 2. First, the case for one uncertainty block is proved (this is in fact the well-known Red-Heffer product). This is extended to the case of a diagonal uncertainty matrix with two blocks, which generalizes the result due to the diagonal structure.

Given the lower LFT $\mathcal{F}_l(M, \Delta_1)$, where Δ_1 can also be represented by a lower LFT $\Delta_1 = \mathcal{F}_l(M^{\Delta_1}, \bar{\Delta}_1)$ and where the coefficient matrices M and M^{Δ_1} are partitioned in the standard 2×2 block format, i.e. Figure 7 assuming $\Delta_2 = 0$.

Substitute $w_1 = \Delta_1 z_1$ by $w_1 = \mathcal{F}_l(M^{\Delta_1}, \bar{\Delta}_1) z_1$ and use the Red-Heffer product [16] to obtain a new lower LFT $\mathcal{F}_l(\bar{M}, \Delta_1)$, see Figure 12:

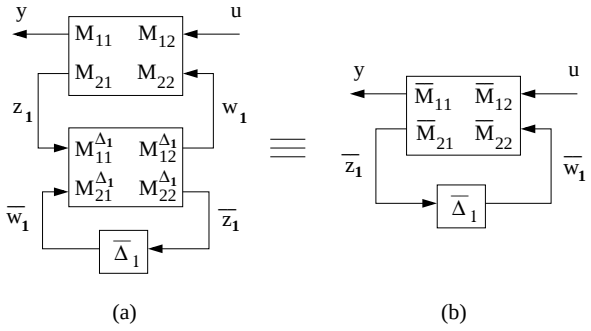


Fig. 12. Nested LFT proof: one block case - $\mathcal{F}_l(\bar{M}, \bar{\Delta}_1)$.

The new coefficient matrix $\bar{M} = [\bar{M}_{11}\ \ \bar{M}_{12};\ \bar{M}_{21}\ \ \bar{M}_{22}]$ is given by:

$$\bar{M}_{11} = \mathcal{F}_l(M, M_{11}^{\Delta}) = M_{11} + M_{12}\Delta_n^{-1}M_{11}^{\Delta}M_{21} \tag{24}$$

$$\bar{M}_{12} = M_{12}\Delta_n^{-1}M_{12}^{\Delta} \tag{25}$$

$$\bar{M}_{21} = M_{21}^{\Delta}(I + M_{22}\Delta_n^{-1}M_{11}^{\Delta})M_{21} = M_{21}^{\Delta}(I - M_{22}M_{11}^{\Delta})^{-1}M_{21} \tag{26}$$

$$\bar{M}_{22} = \mathcal{F}_u(M^{\Delta}, M_{22}) = M_{22}^{\Delta} + M_{21}^{\Delta}M_{22}\Delta_n^{-1}M_{12}^{\Delta} \tag{27}$$

$$\Delta_n = I - M_{11}^{\Delta}M_{22} \tag{28}$$

Note that in the one-block case, $\bar{M}^{\Delta}$ from equation (7) is equal to M^{Δ_1}.

The case for two uncertain blocks $\mathcal{F}_l(M, diag(\Delta_1, \Delta_2))$ where it is desired to substitute $\Delta_1 = \mathcal{F}_l(M^{\Delta_1}, \bar{\Delta}_1)$, is similar to the one-block case but with an intermediate step. The intermediate step augments the coefficient matrix M^{Δ_1} and the uncertain matrix $\bar{\Delta}_1$ with appropriate channels and terms in order to diagonally include the uncertain block Δ_2, thus forming a new lower LFT $\mathcal{F}_l(\bar{M}^{\Delta}, \bar{\Delta})$ with $\bar{\Delta} = diag(\bar{\Delta}_1, \Delta_2)$. As the original uncertain matrix is diagonal (i.e. $w_i = \Delta_i z_i$), the row/column augmentation and uncertainty inclusion is straight-forward, see Figure 13:

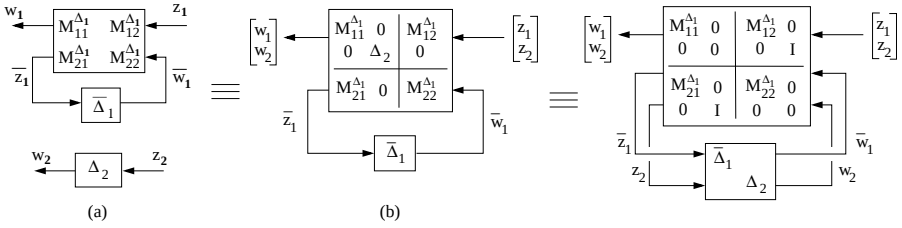


Fig. 13. Nested LFT proof: two block case - (augmenting $\mathcal{F}_l(M^{\Delta_1}, \bar{\Delta}_1)$ with Δ_2) $\equiv \mathcal{F}_l(\bar{M}^{\Delta}, \bar{\Delta})$.

The augmented matrix $\bar{M}^{\Delta}$ is now in the appropriate 2×2 block format to perform a Red-Heffer product with the coefficient matrix M, using equations (24-28), to obtain the desired result: $\mathcal{F}_l(\bar{M}, \bar{\Delta})$ from equations (5-6).

The generalization of this result is direct, noting that the augmented matrix $\bar{M}^{\Delta}$ consists of blocks formed by identity and zero matrices of appropriate dimensions, see equation (7), with those terms in the position (ii, ii) (corresponding to the position in Δ of the uncertain block to be substituted, e.g. Δ_1 is in the $(ii, ii) = (1, 1)$ position in Δ) equal to the terms from the coefficient matrix to be substituted, i.e. M^{Δ_1}.