A Symbolic Matrix Decomposition Algorithm for Reduced Order Linear Fractional Transformation Modelling [⋆]

Andrés Marcos [*], Declan G. Bates, Ian Postlethwaite

Dept. of Engineering, University of Leicester, University Road, Leicester, LE1 7RH, U.K.

Abstract

In this paper an algorithm that provides an equivalent, but of reduced order, representation for multivariate polynomial matrices is given. It combines ideas from computational symbolic algebra, polynomial/matrix algebraic manipulations and information logic. The algorithm is applied to the problem of finding minimal linear fractional transformation models. Statistical performance analysis of the algorithm reveals that it consistently outperforms currently available algorithms.

Key words: linear fractional transformation, symbolic multivariate polynomial matrices

1 Introduction

Matrix manipulation is one of the basic cornerstones of many fields in engineering and mathematics. For example, in the field of control the basis of state-space theory is the representation of a system as a two-by-two block matrix and its subsequent use for synthesis and analysis Skogestad, S. and Postlethwaite, I. (1996). Indeed, modern robust control theory is built on the concept of linear fractional transformations (LFT) which allows the representation of an uncertain system in terms of nominal and uncertain components given by matrices, Balas, G.J., Doyle, J.C., Glover, K., Packard, A., and Smith, R. (1998). Similarly, polynomial representations and manipulations are also ubiquitous in many areas of mathematics – note, for example, that most computer algebra systems such as *Mathematica* Wolfram, S. (1991) and Maple Heck, A. (1993) rely on them.

A typical objective (for example, in signal processing and control synthesis) when operating on parametric matrices and polynomials is to obtain an equivalent representation of reduced order (defined as the 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, Cockburn, J.C. (2000), which remains an open problem and is usually approached from an algebraic algorithmic perspective. Indeed, most of the available LFT order-reduction algorithms search for a minimal representation by exploiting the structure in the LFT models and performing algebraic operations on the multivariate matrices: e.g. structured-tree decomposition Cockburn, J.C. and Morton, B.G. (1997), numerical matrix approaches Belcastro, C.M. and Chang, B.C. (1998), and Horner factorizations Varga, A. and Looye, G. (1999) amongst others.

In this paper, an algorithm is proposed that builds on previous symbolic techniques to achieve a lower order representation for multivariate polynomial matrices. The main difference with previous algorithms is the use of an information management scheme which uses a set of specialized metrics and decision logic to evaluate the best (in terms of achievable order reduction) parameter ordering and matrix operation. It is especially useful for LFT modelling, and hence, the application of the algorithm to this problem is described in detail. The algorithm software implementation is freely available from the first author.

2 Theoretical Background

The basic structure of the proposed algorithm is a combination of the structured-tree decomposition Cockburn, J.C. and Morton, B.G. (1997), and the Horner factorization Varga, A. and Looye, G. (1999) approaches, extended by an information management module with special metrics and appropriate decision logic. Most of

⋆ Research supported under EPSRC research grant $GR/S61874/01$. Article presented at the IFAC World Congress on Automatic Control, Prague, 2005.

* Corresponding author is now a Senior Engineer in the Simulation & Control Section at Deimos Space S.L., Madrid, Spain. Tel: +34-91-806-3462; Fax: +34-91-806-3451;

Email addresses: andres.marcos@deimos-space.com (Andrés Marcos), dgb3@le.ac.uk (Declan G. Bates), ixp@le.ac.uk (Ian Postlethwaite).

the basic operations (i.e. sum decomposition, affine and polynomial Horner factorizations) used in the proposed algorithm were developed in the two previous algorithms, which also introduced the idea of a reduction metric. In this section, these metrics and symbolic operations are reviewed and additional ones are introduced.

2.1 Multivariate Matrices

The proposed 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. Almost all engineering systems and applications can be represented as multivariate polynomial matrices – a very representative case is the state-space models used in control design.

The algorithm can be extended to symbolic rational expressions $P(\delta_i)$ using the generalized approach from Hecker, S. and Varga, A. (2004) or the factorization result by Belcastro, C.M. (1998); Magni, J.F. (2004):

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

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

2.2 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 $\delta_1, \delta_2, \ldots, \delta_n$ a monomial m is defined as $m = c\delta_1^{\alpha_1}\delta_2^{\alpha_2}\ldots\delta_n^{\alpha_n}$ 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 minimal 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: $(k-l-1)fac(\delta_i)$.

2.3 Tree Decomposition & Horner Factorization

The structured-tree decomposition is an iterative approach that decomposes any matrix A_0 into simpler (i.e. of smaller σ) matrices by means of two basic operations, affine factorizations $(L_1A_1R_1)$ and sum decompositions $(A_2 + A_3)$: $A_0 = L_1A_1R_1 + (A_2 + A_3)$. The name comes from its graph-nature as each simpler matrix is in turn sum-decomposed or affine-factorized until it cannot be decomposed further. The graph forms a tree where the nodes are one of the two basic operations and the leaves the matrices as they are reduced. A third operation, weighted-sum decomposition, uses information similar to the above metrics (i.e. maximal factor order) when no sum decompositions or affine factorizations exist. This approach has been recently coded and publicly released in ONERA's linear fractional representation (LFR) toolbox Magni, J.F. (2004).

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, Wolfram, S. (1991). 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).

Note that the structured-tree decomposition approach does not directly incorporate Horner factorizations for polynomial expressions and thus might involve more operations (sums and products) than required. Also, the use of the maximal factor order as a decision logic might not be the 'optimal' (i.e. best at that step) reduction of the total presence degree. On the other hand, current Horner factorization algorithms follow either a pure lexicographical order or an exhaustive approach (i.e. try all combinations of the parameters) for the ordering of the symbolic parameters. For large number of parameters these approaches are too computationally expensive and do not typically result in a minimal realization.

Furthermore, the Horner factorization algorithm considers only nested factorizations for each symbolic polynomial in an independent manner (i.e. unrelated to those for the other polynomials or matrix coefficients). This might not result in a minimal realization in the matrix case, since the best nesting of a polynomial in terms of its total σ reduction might not result in the largest σ reduction for the matrix, as shown in the example:

Example 1 *Given* $M = \begin{bmatrix} \delta_1^3 + \delta_1^3\delta_2^2\delta_3^2 + \delta_2^2\delta_3^2 & \delta_2^2\delta_3^2 \end{bmatrix}$, *and assuming it is desired to apply Horner factorization in the* $\{1,1\}$ *coefficient rather than sum decompositions (no affine factorization exits 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* $\sigma = 10$. $\square$

2.4 Other Symbolic Algebraic Techniques

Other techniques are available which facilitate matrix manipulation and improve algorithmic performance when applied to symbolic expressions, see Armita, P. and De Micheli, G. (2001). This has long been recognized, and some algorithms already implement several or all of these techniques as they have also become available and optimized on all computer algebra systems. The two techniques incorporated in the proposed algorithm are expansion and substitution.

The procedure called "expand" distributes all products and positive powers over sums. Like-terms are also collected and simplified, which can often result in immediate gains, e.g. $expand\{\delta_1(1+\delta_2) + \delta_1(1+\delta_1)\} = \delta_1(2 + \delta_1 + \delta_2)$. In the proposed algorithm, expansion is used after an iteration is completed, i.e. all possible factorizations and decompositions have been performed for the present matrix form. Its purpose is to reset the polynomial factorizations to check whether a new reduced-order matrix can be found.

Another important symbolic tool is "substitution", which replaces an expression by a new symbolic parameter. In its most developed form, it can be used to reduce the complexity of an expression and improve speed of decomposition. For example, if a symbolic row matrix is given by $M = [\ \delta_1\delta_3 + \delta_2\delta_3 + \delta_1\delta_4 + \delta_2\delta_4 \quad \delta_1 + \delta_2]$ then standard symbolic factorizations can be used to obtain $M = [(\delta_1+\delta_2)(\delta_3+\delta_4) \quad (\delta_1+\delta_2)]$; since it is more difficult for an algorithm to recognize the entire expression $(\delta_1 + \delta_2)$, it can be substituted by δ_a to obtain $M = [\delta_a(\delta_3+\delta_4) \quad \delta_a]$ which is now easily recognized and affine factorized.

3 Logic Horner Tree Decomposition Algorithm

The proposed decomposition algorithm is called *Logic Horner Tree (LHT)* to highlight its connection to the previous approaches and to emphasize the inclusion of decision logic. It 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 decision logic uses the metrics defined before to choose which symbolic parameters to select and which matrix operation to perform (i.e. sum decomposition, Horner or affine factorization).

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:

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

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 $\bar{M}_{A_{ii-1}}$. A pseudo-code of the structure of the algorithm is given in page 4. Each step is detailed in the next sub-sections (in general terms in 3.1 and more detailed in 3.2 and 3.3), but note that the first step is to expand M and perform a sum decomposition to shift the constant monomials/coefficients to MB_1.

3.1 Information Management

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. Its main objectives are: *i*) to gather the proper information at each step, *ii*) to take a decision regarding operation and parameter ordering, and *iii*) to prepare the decomposition matrix for the chosen operation. These objectives are performed at different stages of the algorithm as shown in the pseudo-code.

The information gathering pertains mainly to calculating the metrics 'METRICS' presented in Section 2.2 for all the specified symbolic parameters and also 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 factor order $fac(\delta_k)$ is identified along any row or column, if a higher possible reduction order $red_{pos}(\delta_k)$ is also calculated, the corresponding symbolic parameter can be scheduled for a sum decomposition, see the following example:

Example 2 *Given* $M = \begin{bmatrix} \delta_3^2 & \delta_1 \\ \delta_3^2 & \delta_3^2 \\ \delta_1\delta_3 + \delta_2\delta_3^2 & \delta_3^3 \end{bmatrix}$,
a direct affine factorization of δ_3 *can be performed for the 2nd and 3rd rows:* $M_{dec} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \delta_3^2 & 0 \\ 0 & 0 & \delta_3 \end{bmatrix} \begin{bmatrix} \delta_3^2 & \delta_1 \\ 1 & 1 \\ \delta_1 + \delta_2\delta_3 & \delta_3^2 \end{bmatrix}$

Further manipulations on M_{dec} 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 we can perform an affine factorization of δ_3 along the 1st column followed by a sum decomposition to get:

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

which can be further manipulated (columns and row affine fact. 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. □

```
LHT($M(\delta)$)
  $i \leftarrow 1$;  $ok \leftarrow 1$
  while $ok = 1$
    do $M \leftarrow$ EXPAND($M$)
    if $i = 1$
      then $MA[1] \leftarrow$ Symbolic-Coeff($M$)
           $MB[1] \leftarrow$ Constant-Coeff($M$)
    $metrics \leftarrow$ Metrics($MA[i]$)
    $ordvect \leftarrow$ Sigma-Ordering($MA[i]$)
    if Exist-PolyCoeff($MA[i]$) = $yes$
      then $Hlist \leftarrow$ Horner-Ordering($MA[i], metrics$)
           $MHA \leftarrow$ Horner-Factorization($MA[i], Hlist$)
           $MA[i], polylist \leftarrow$ Poly-Substitution($MHA$)
           $metrics \leftarrow$ Metrics($MA[i]$)
           $ordvect \leftarrow$ Sigma-Ordering($MA[i]$)
    for $j \leftarrow 1$ to LENGHT[$ordvect$]
      $affdir \leftarrow$ Affine-Direction($MA[i], metrics, ordvect[j]$)
        ▷ if $affdir = 0 \Rightarrow$ skip parameter affine factorization
      $L[i], MA[i], R[i] \leftarrow$ Aff-Factor($MA[i], affdir, ordvect[j]$)
        ▷ short-hand: $LMAR[i] \equiv L[i], MA[i], R[i]$
    if NOEMPTY($polylist$)
      then $LMAR[i] \leftarrow$ Poly-BackSubs($LMAR[i], polylist$)
           $LMAR[i] \leftarrow$ EXPAND($LMAR[i]$)
           $polylist \leftarrow empty$
    $declist \leftarrow$ Decomposition-list($MA[i]$)
    $MA[i+1], MB[i+1] \leftarrow$ Sum-Decomp($MA[i]$, [], $declist[1]$)
      ▷ short-hand: $MAMB[i+1] \equiv MA[i+1], MB[i+1]$
    $j \leftarrow 1$
    while NOEMPTY($declist$)
      do $j \leftarrow j + 1$
      $declist \leftarrow$ Conflict-Analysis($MAMB[i+1], declist[j]$)
      $MAMB[i+1] \leftarrow$ Sum-Decomp($MAMB[i+1], declist[j]$)
    $M \leftarrow MA[i+1]$
    $i \leftarrow i + 1$
    if Fully-Dec($M$) and Nonfully-Dec($MB[k]$)
      then $M \leftarrow MB[k]$
           ▷ $k$ is special index to get correct $MB$
    elseif Fully-Dec($M$) and Fully-Dec($MB[k]$)
      then $ok \leftarrow 0$
```

The parameter ordering 'SIGMA-ORDERING' and matrix preparation depends on the symbolic operation that follows. 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'. For the sum decomposition, the IM forms a 'DECOMPOSITION-LIST'.

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, see example 1.

In the 'POLY-SUBSTITUTION' step (performed before the affine factorizations), the IM performs a substitution of the sub-polynomials arising from the Horner factorization 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. This 'POLY-BACKSUBS' step followed by expansion allows the IM to adequately update the metrics for all the symbolic parameters.

The 'DECOMPOSITION-LIST' contains information for the sum decomposition routine to attempt 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, sub-classified 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 in Section 3.3.

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.

3.2 Factorization

The factorization stage 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 matrix extension of the polynomial Horner factorization is based on the ordering of the symbolic vector given by the 'HORNER-ORDERING' step. In this manner, as shown in example 1, the emphasis is placed on decreasing the reduction order *red* of the matrix and not for each individual polynomial coefficient.

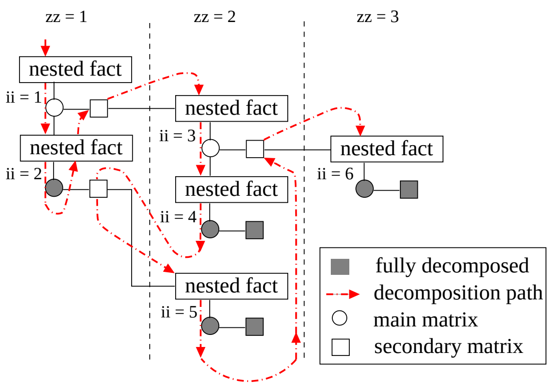

Fig. 1. Graph Logic Horner Tree (LHT) algorithm.

In the 'AFF-FACTOR' step, 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.

3.3 Sum Decomposition

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' given by the information manager and using a conflict logic 'CONFLICT-ANALYSIS' to form the primary and secondary matrices.

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. □

4 LFT modelling Application

In the context of robust control theory, a linear fractional transformation (LFT) is generally used to represent an uncertain system using two matrix 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.

In the case of parametric uncertainty (the case of interest in this paper), 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 Magni, J.F. (2004).

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 this property and, in particular, the symbolic 'object-oriented' realization of LFTs as proposed in Magni, J.F. (2004). 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 Lambrechts, P., Terlouw, J., Bennani, S., and Steinbuch, M. (1993); Magni, J.F. (2004)). 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 can be transformed into an LFT, with a diagonal uncertainty matrix of order equal to the respective matrix total presence degree. The resulting 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 an 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 Balas, G.J. et al. (1998). This is valid as well for exact nonlinear modelling Marcos, A., Bates, D.G., and Postlethwaite, I. (2005) when the non-linearities are considered as parameters to be 'pulled out' into the Δ matrix.

5 Monte Carlo Tests

Monte-Carlo tests of the LHT algorithm software implementation are performed to evaluate its capability and efficiency. A flight dynamics example using an earlier implementation (more conservative in the results) is given in Marcos, A., Bates, D.G., and Postlethwaite, I. (2004). In order to provide a baseline comparison, the command 'symtreed' provided in ONERA's LFR toolbox version 1.1.1 Magni, J.F. (2004), which implements the structured-tree decomposition is also used. The results from 'symtreed' depends on the ordering of the symbolic parameters - thus, to simplify the comparison a lexicographic arrangement is used.

The Monte-Carlo test is performed using 500 random polynomial matrices (200 3×3 matrices, 200 5×7 and 100 9×5). For each matrix dimension, half of the matrices are populated with 3 different symbolic parameters and the other half with 9. The random polynomial matrices are obtained by adding three independent random monomial matrices (thus, only polynomials of up to three monomials can be found in the matrix coefficients). Each independent monomial matrix is obtained by creating a sparse random matrix with the chosen matrix dimensions $n \times m$ and a specified density: one monomial matrix with density 60 and the other two with density 0.3 (so that not all the final non-zero coefficients will be polynomials). These specifications yield approximately $n{\cdot}m{\cdot}density$ uniformly distributed non-zero entries for each random monomial matrix. In order to fill each non-zero coefficient of the monomial matrices, k passes (where k is equal to 3 times the number of parameters, hence 9 or 27 passes) are made multiplying each time the coefficients by δ_i^{pow} where δ_i is a randomly selected symbolic parameter from the defined list and pow is a random binary selection for the parameter's power (either 0 or 1, which means that the non-zero coefficients for each of the independent monomial matrices can have a total degree of up to 9 or 27 depending on the number of parameters).

Table 1 shows for each matrix dimension the percentage of runs for which one algorithm resulted in a smaller total presence degree σ than the other (i.e. smaller LFT order). The number in parentheses indicates the average LFT order of, respectively for each column: the random polynomial matrix, the logic Horner tree decomposed matrix, and the structured-tree matrix.

Table 1
Algorithm efficiency (% total runs $\sigma_A > \sigma_B$).

dimension	**LHT**	**Symtree**
3×3 (116)	90.5 % (61)	4.0 % (68)
5×7 (434)	99.0 % (209)	1.0 % (243)
9×5 (551)	99.0 % (259)	1.0 % (302)

The first important conclusion obtained from looking at the average total presence degree is that these order-reduction techniques should always be incorporated in the LFT modelling process, since significant reductions in LFT order are achieved by both algorithms. Note that this reduction in the total σ is not associated with any loss of modelling fidelity, since the reduction is accomplished solely through algebraic manipulations. Also, the current availability of software for LFT modelling and order-reduction (e.g. Magni, J.F. (2004) and that stemming from this research) which simplifies its use to a simple MATLAB command further supports this conclusion.

Comparing the efficiency of the algorithms, it is observed that the LHT algorithm results in better order reduction in almost 99 percent of the trials (except for the smaller matrices where it is only better around 90 percent of the time). Those cases where the structured-tree decomposition algorithm is superior probably stem from a need to implement extra logic in the LHT algorithm or to expand its "pre-viewing" capability in the affine factorization step of the algorithm.

Table 2 shows the relative quality of the improvement between the algorithms. The first number in each column is the average percentage improvement in the total presence degree between both algorithms, e.g. $pct_{symtree} = 100 \sum [\frac{abs(\sigma(LHT) - \sigma(symtree))}{\sigma(symtree)}] / \#_{runs}$. The numbers in parentheses indicate average order by which one algorithm improved over the other. Note that the percentage improvements due to the LHT algorithm are at least triple those from the structured-tree algorithm, and that this improvement increases as the complexity increases, i.e. as larger σ and matrix dimensions are used.

Table 2
Algorithm performance (% average σ improvement).

dimension	**LHT**	**Symtree**
3x3	12.00 % (9)	4.20 % (2)
5x7	13.35 % (34)	1.10 % (1)
9x5	14.20 % (43)	1.68 % (2)

Table 3, as in Table 1, presents the average percentage of times (and the average σ) one algorithm was better than the other but including the results when the n-D numerical minimization technique (MIN) by D'andrea, R. and Khatri, S. (1997), as implemented by 'minlfr' in Magni, J.F. (2004), is applied after the symbolic order-reduction algorithms. The case shown is for the 250 random matrices with 9 symbolic parameters.

Table 3
Algorithm efficiency using n-D numerical minimization.

dim	**LHT**	**MIN**	**Symtree**	**MIN**
3×3(172)	95 (93)	87 (92)	3 (105)	8 (101)
5×7(645)	100 (330)	100 (326)	0 (383)	0 (364)
9×5(828)	100 (418)	100 (413)	0 (485)	0 (458)

Note that even after the application of the numerical minimization technique, the results from the LHT algorithm are better (although the difference in the total σ degrees for a given matrix dimension is now smaller). Also, notice that the improvement in order reduction accomplished by the n-D technique in comparison to the σ degree obtained from the direct application of the symbolic algorithms is smaller for the LHT algorithm (typically, 1.5 percent improvement for the LHT versus 5.5 percent improvement for the structured-tree decomposition). For the case of three or fewer parameters (not shown) both algorithms gave similar performance after the n-D reduction technique (a more conservative result that the one from Table 1). This indicates that the LHT algorithm is superior for symbolic matrices with large number of parameters (more than four) regardless of the application of numerical reduction techniques, but that for the other cases both algorithms should be employed (together with the numerical techniques).

In order to provide further insight into the operation of the algorithms, a statistical analysis using the 200 matrices of dimension 5×7 was carried out – similar results were obtained for the other dimensions. Figure 2 shows the scatter plot of the relative σ degrees standard deviation (STD) for each matrix. The STD is calculated using the presence degree for each of the symbolic parameters $\sigma(\delta_i)$ (recall, the first 100 matrices have 3 symbolic parameters and the last 100 have 9). The solid line indicates the mean value of the 200 standard deviations.

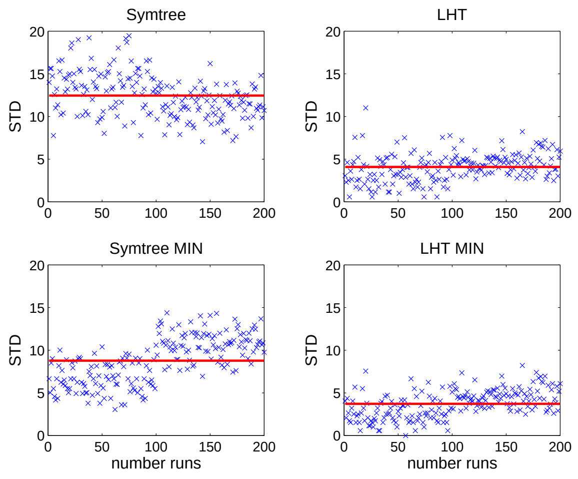

Fig. 2. Standard deviation scatter plot for 5×7 matrices.

Note that the LHT algorithm results in a smaller mean STD even after the application of the n-D minimization technique (MIN). Also, the previous comments regarding the smaller effect of MIN for this algorithm are also easily appreciated (the mean value and the populations are very similar for LHT before and after the numerical minimization). Note as well that the population of the STDs are closer to their corresponding mean value for the LHT algorithm than for the structured-tree algorithm which indicates that the LHT algorithm is more consistent in achieving its order reduction objective (i.e. less likely to produce statistical outliers). Although no general conclusions can be inferred from these statistical analyses, it is interesting to note that there seems to be a standard deviation minimization objective in the way the LHT algorithm and the n-D technique work.

Finally, we note that the implementation used for the structured-tree decomposition (i.e. 'symtreed') is faster than the present implementation of the LHT algorithm (for 9×5 random matrices with σ of 400 and 9 symbolic parameters, typically one minute versus five minutes although for larger dimensions it can be around 10 times slower). Nevertheless, the results from the 'symtreed' implementation depend on the ordering of the symbolic parameters vector and, in theory, it should be run $n!$ times where n is the number of parameters (which for the case of 9 parameters implies 362,880 times), see example 1. This is not the case for the LHT algorithm since the ordering of the symbolic vector is automatically performed based on the metrics from Section 2.2.

6 Conclusions

In this paper an algorithm for the decomposition of multivariate polynomial matrices has been proposed. The resulting representation is equivalent but of reduced order, if order reduction is possible, to the original matrix. The algorithm is applicable to a wide range of fields, but particularly to LFT/LFR modelling, one of the cornerstones in modern robust control theory. Monte-Carlo tests of the proposed algorithm applied to LFT modelling showed significant improvement in performance when compared with other algorithmic approaches.

References

Armita, P., De Micheli, G., June 2001. Symbolic Algebra and Timing Driven Data-Flow Synthesis. In: Design Automation Conference. Las Vegas, NV, pp. 277–282.

Balas, G.J., Doyle, J.C., Glover, K., Packard, A., Smith, R., June 1998. μ-Analysis and Synthesis Toolbox.

Belcastro, C.M., November 1998. On the numerical formulation of parametric linear fractional trasnformations (lft) uncertainty models for multivariate matrix polynomial problems. Tech. Rep. NASA/ TM-1998-206939, NASA Langley Research Center, Hampton, VA.

Belcastro, C.M., Chang, B.C., June 1998. LFT Formulation for Multivariable Polynomial Problems. In: American Control Conference. Philadelphia, PA, pp. 1002–1007.

Cockburn, J.C., June 2000. Multidimensional realizations of systems with parametric uncertainty. In: Mathematical Theory of Networks and Systems. Perpignan, France.

Cockburn, J.C., Morton, B.G., 1997. Linear Fractional Representations of Uncertain Systems. Automatica 33 (7), 1263–1271.

D'andrea, R., Khatri, S., June 1997. Kalman decomposition of linear fractional transformation representations and minimality. In: American Control Conference. Alburquerque, NM, pp. 3557–3561.

Heck, A., 1993. Introduction to Maple. Springer-Verlag.

Hecker, S., Varga, A., 2004. Generalized lft-based representation of parametric uncertain models. European Journal of Control 10 (4).

Lambrechts, P., Terlouw, J., Bennani, S., Steinbuch, M., June 1993. Parametric Uncertainty Modeling using LFTs. In: American Control Conference. San Franciso, CA, pp. 267–272.

Magni, J.F., January 2004. Linear Fractional Representation Toolbox Modelling, Order Reduction, Gain Scheduling. Tech. Rep. TR 6/08162 DCSD, ONERA, Systems Control and Flight Dynamics Department, Toulouse, France.

Marcos, A., Bates, D.G., Postlethwaite, I., November 2004. Flight Dynamics Application of a New Symbolic Matrix Order-Reduction Algorithm. In: International Conference on Polynomial Symbolic Systems. Paris, FR.

Marcos, A., Bates, D.G., Postlethwaite, I., June 2005. Exact Nonlinear Modeling using Symbolic Linear Fractional Transformations. In: IFAC World Congress. Praga, CH.

Skogestad, S., Postlethwaite, I., May 1996. Multivariable Feedback Control Analysis and Design. Wiley.

Varga, A., Looye, G., August 1999. Symbolic and Numerical Software tools for LFT-based Low Order Uncertainty Modeling. In: IEEE Symposium on Computed Aided Control System Design. Hawai'i, USA.

Wolfram, S., 1991. Mathematica: *a System for Doing Mathematics by Computer*. Addison-Wesley.