AUTOMATIC GENERATION OF LFTS FROM OBJECT-ORIENTED NON-LINEAR MODELS WITH UNCERTAIN PARAMETERS

F. Casella[1], F. Donida[1], M. Lovera[1]
[1]Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy

Corresponding author: F. Casella, Politecnico di Milano, Dipartimento di Elettronica e Informazione, Piazza L. da Vinci, 32, 20133 Milano, Italy, casella@elet.polimi.it

Abstract. Linear Fractional Transformations (LFTs) are a widely used model description formalism in modern control and system identification theory, in which uncertain parameters and non-linearities are "pulled out" from the system, resulting in the feedback connection between a linear, time-invariant model and blocks representing the uncertain and/or nonlinear parts. Deriving such representations from physical models is a non-trivial, lengthy and error-prone process, if carried out manually. This paper presents an algorithm to obtain LFT models starting from generic equation-based, object-oriented descriptions of nonlinear plant dynamics, with uncertain parameters. This allows to eliminate the gap between the use of advanced LFT-based control system analysis and design techniques and user-friendly model representations, such as object diagrams with physical connections, block diagrams with signal connections, and generic differential-algebraic models.

1 Introduction

The process of developing control-oriented mathematical models of physical systems is a complex task, which in general implies a careful combination of prior knowledge about the physics of the system under study with information coming from experimental data (leading to the so-called problem of grey-box modelling, see, e.g., [4]), in view of the application of the model to control systems analysis and design. As recently discussed in [14], the critical issue in the development of an effective approach to control-oriented grey-box modelling lies in the integration of existing methods and tools for physical systems modelling and simulation with methods and tools for parameter estimation. Furthermore, throughout the modelling exercise, one should always keep in mind that the eventual application of the model is control system analysis and design, so the mathematical structure of the model has to be compatible with currently available methods and tools for such problems.

In this paper an approach is proposed to bridge the gap between physical and control/estimation-oriented system modelling, by automatically deriving standard model structures used in system identification and control starting from object-oriented (OO) models of nonlinear plants.

As far as OO modelling is concerned, the Modelica modelling language has been considered here, as it allows to describe the plant dynamics in a very general, natural and user-friendly way; note however that the concepts and algorithms introduced here can be applied to any a-causal, equation-based modelling language without any substantial change. On the other hand, the Linear Fractional Transformation (LFT) model structure has been considered as the natural target as a goal for identification and control applications. Indeed, LFTs are a widely used model description formalism both in modern control [21, 15, 7] and in system identification [13, 20, 9, 12].

The aim of this paper is to present the latest results obtained in the development of an algorithm to obtain an LFT representation starting from a system model written with the Modelica language. The paper extends the preliminary results presented in [6], which were limited to linear models, to generic nonlinear models with uncertain parameters.

The paper is organised as follows: in Section 2 the basic concepts of object-oriented and LFT modelling are briefly reviewed. The main results concerning the algorithm are presented in Section 3, with implementation details discussed in the subsequent Section 4. Finally, Section 5 presents some examples of application of the algorithm and Section 6 discusses extensions and open problems.

2 Background: O-O modelling and LFTs

The Modelica language ([16, 11, 18]) was first introduced in 1997, as the product of an international cooperative effort towards the definition of an object-oriented language for the modelling of general physical models, described by algebraic and differential equations. Modelica is an open standard, endorsed by a non-profit organization, it is now quite stable, and is being supported by an increasing number of simulation tools.

Elementary models can be described in Modelica by writing generic differential-algebraic equations (DAEs), following an a-causal, declarative approach: the interface between the model and the outside world is not defined by input and output signals, but by physical ports, with flow and effort variables. The connection between sub-models is defined by *connection equations*, which state that all effort variables (e.g., voltages) are equal, and all flow vari-

ables (e.g., currents) sum to zero. The causality is determined automatically by the tools at the overall system level, by means of symbolic manipulation. It is of course also possible to define causal blocks when this is appropriate, e.g., for signal-processing systems, and to mix causal and a-causal models in the same system when needed.

The object-oriented features of the language (inheritance, replaceable models) promote the re-use of existing and well-tested models at all levels, thus reducing development time and errors. On the other hand, it is very easy to write new components, or to customize existing ones, since one can just write the corresponding differential and algebraic equations as they appear on paper.

Linear Fractional Transformations (LFT) are a widely used model description formalism in modern control and system identification theory [21, 15, 7]. The LFT representation of the plant dynamics, shown in Fig. 1, corresponds to the following equations:

$$\dot{x} = Ax + B_1 w + B_2 \zeta + B_3 u \tag{1}$$
$$z = C_1 x + D_{11} w + D_{12} \zeta + D_{13} u \tag{2}$$
$$\omega = C_2 x + D_{21} w + D_{22} \zeta + D_{23} u \tag{3}$$
$$y = C_3 x + D_{31} w + D_{32} \zeta + D_{33} u \tag{4}$$
$$w = diag\left\{\delta_1 I_{r_1}, ..., \delta_q I_{r_q}\right\} z \tag{5}$$
$$\zeta = \Theta(\omega) \tag{6}$$

where x is the n-dimensional vector of state variables, u is the m_u-dimensional vector of input variables, y is the m_y-dimensional vector of output variables, δ_j are q uncertain (possibly measurable) parameters and r_j are the sizes of the corresponding identity matrices in the Δ block ($j = 1, \ldots, q$). The structure of the vector function $\Theta(\omega)$ is such that its input vector ω and its output vector ζ can each be partitioned into m_f sub-vectors ω_i and ζ_i, such that $\zeta_i = \Theta_i(\omega_i)$.

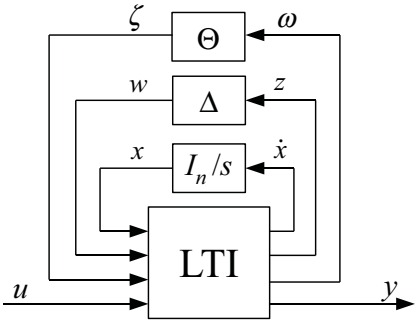

Figure 1: Block diagram of the considered LFT representation.

Deriving such models from first-principles physical models is a non-trivial, error-prone and tedious process, if carried out manually, so there is clearly a strong need of user-friendly tools to perform this task automatically.

A first step in this direction is taken by control-oriented tools such as the LFR Toolbox [15], which is based on MATLAB's Symbolic Toolbox (a Scilab implementation exploiting Maxima's symbolic engine is also available). This tool provides several basic functionalities, which allow to:

- define LFTs corresponding to linear models with uncertain parameters via symbolic objects, starting from parameter-dependent formulations of either the transfer function or the A, B, C, D matrices of the state-space description;
- define LFTs corresponding to static nonlinear functions;
- hierarchically build complex models by series, parallel and feedback interconnection, as well as horizontal or vertical vector concatenation;
- reduce the size of the Δ block by suitable algorithms, obtaining minimal (or close to minimal) realizations.

A fundamental limitation of these tools is that they can only handle causal models, where the interaction between the different sub-models is expressed in terms of input/output interaction. This is not the most natural way to describe the interaction between physical components in a plant. A further step (as first suggested in [6]), is to obtain an LFT representation starting from a system model written with the Modelica language, which allows to describe the plant dynamics in a much more general, natural and user-friendly way. Preliminary results were presented in [5], limited to the case of linear models. This paper presents a more general algorithm to deal with generic nonlinear models.

3 Transformation of nonlinear O-O models into LFT representations

3.1 Applicable plant models

The starting point of the analysis is an object-oriented Modelica model of the plant dynamics, obtained by the aggregation of many sub-models through either causal (input-output), or a-causal (physical ports carrying effort and flow variables) connectors. Pure block diagrams are included as a special case, but in general it is possible to combine block diagrams with object-oriented descriptions of physical systems of diverse nature. It is assumed that the model is time-invariant (i.e., the *time* variable does not appear explicitly within any model equation), and that all variables and equations are continuous-time (i.e., there are no *when* statements and discrete variables in the model). Furthermore, the model is supplemented by a list of the uncertain parameters, with their minimum and maximum values; this information can be either supplied separately from the Modelica model, or embedded into the Modelica model by means of the *min*/*max*/*nominal* attributes, as well as of custom annotations to flag the uncertain parameters.

3.2 Preliminary manipulation within the Modelica compiler

The first steps must be performed within the Modelica compiler, and are similar to the steps routinely performed by all Modelica compilers to produce simulation code.

The object-oriented representation of the system under consideration is first *flattened*, i.e., reduced to a single model without any hierarchical structure left, either in terms of aggregation or of inheritance among sub-models. All variables and parameters are identified through the dot notation (e.g., Circuit.R1.R refers to the parameter R within the model of resistor R1, contained within the model Circuit). The flattened model is equivalent to a system of differential-algebraic equations (DAEs)

$$F_0(x_0, \dot{x}_0, v_0, u, p_0) = 0 \tag{7}$$

$$p_0 = G_0(p_0), \tag{8}$$

where $F_0(\cdot)$ is a vector-valued function, x_0 is the vector of the dynamic variables, which also appear under derivative sign, v_0 the vector of the algebraic variables, which are not differentiated in the original model, u the vector of exogenous inputs, and p_0 the vector of all the system parameters. The values of the parameters are given by the binding equations (8), which specify the value of each parameter either by a numerical value, or as a function of other parameters. We assume equation (8) does not contain any implicit equation or cyclic dependencies among parameters, so that it can be solved by suitably re-ordering the corresponding scalar equations. This hypothesis is enforced by the Modelica language specification (see [19], §4.4.3).

The output vector y of the system is declared as a subset of the set of state and algebraic variables x_0, v_0.

The system (7) can have structural index greater than one: this is typically the case when some of the algebraic equations within (7) are algebraic constraints between dynamic variables. For example, parallel connections of capacitors and series connections of inductors lead to index-2 problems, while rigid connections among mechanical objects with inertia lead to index-3 problems.

In these cases, it is possible to transform the system into an equivalent index-1 formulation by using Pantelides' algorithm and the dummy derivative algorithm by Mattsson and Söderlind [17]. The result is a system of DAEs with (structural) index 1, characterised by a reduced number of dynamic variables x which can now be called state variables, as they can be given arbitrary values at the initial time, and with a different choice of algebraic variables v, including the dummy derivatives. The dummy derivatives algorithm can also be used to change the selection of state variables from the default choice corresponding to the variables that appear differentiated in the original model (x_0) to a different set of variables of the model.

Following these transformations, the system equations are formulated as the index-1 DAE

$$F(x, \dot{x}, v, u, p_0) = 0. \tag{9}$$

At each time instant, the values of states and inputs are known; the numerical values of the parameters p_0 are not known explicitly, but they can be considered as known, given the binding equations (8). The goal is now to compute the state derivatives $\dot{x}$ and the algebraic variables v. To this end, the equations and the variables of the problem (9) can be re-ordered so that the incidence matrix (equations on the rows, unknowns on the columns) is brought in Block-Lower-Triangular (BLT) form. This task is accomplished by using the well-known Tarjan algorithm [10], applied to the equations-variables bipartite graph, which is equivalent to the incidence matrix of the system. The strongly connected components of the graph correspond to the blocks on the diagonal, and a partial ordering among equations can be deduced from the graph after the algorithm has terminated.

After re-ordering, the system (9) can be formulated as

$$\Phi(x, u, \Xi, p_0) = 0, \tag{10}$$

where $\Phi(\cdot)$ is the set of re-ordered equation residuals and Ξ is the re-ordered set of the system unknowns (i.e., all the elements of vectors $\dot{x}$ and v). By defining $\Phi_j(\cdot)$ as the j-th sub-set of equations corresponding to one block of the BLT form, Ξ_j as the corresponding sub-set of unknown variables, and q as the number of blocks on the diagonal of the BLT incidence matrix, the re-ordered system equations (10) can be formulated as

$$\Phi_1(x,u,\Xi_1,p_0) = 0 \tag{11}$$
$$\Phi_2(x,u,\Xi_1,\Xi_2,p_0) = 0 \tag{12}$$
$$... \tag{13}$$
$$\Phi_q(x,u,\Xi_1,...,\Xi_{q-1},\Xi_q,p_0) = 0. \tag{14}$$

The system expressed in the form (11)-(14) can now be exported from the Modelica compiler and imported into a suitable environment, which supports the symbolic manipulation of LFTs.

3.3 Recursive formulation of the system equations

The first step of the transformation to LFT form is to bring the system of equations in a form which only depends explicitly on the uncertain parameter vector p, whose elements are a sub-set of the elements of p_0; this operation can be performed by using a symbolic manipulation tool. For this purpose, the binding equations corresponding to the uncertain parameters p are removed from the system (8), the elements of p are defined as symbolic objects, and the remaining equations are solved for the remaining parameters, which will then be given by either straight numerical values or symbolic expressions depending on elements of p. These expressions can then be symbolically substituted in the system equations (11)-(14), thus re-defining the functions Φ_j so that they only depend on the uncertain parameters p

$$\Phi_1(x,u,\Xi_1,p) = 0 \tag{15}$$
$$\Phi_2(x,u,\Xi_1,\Xi_2,p) = 0 \tag{16}$$
$$... \tag{17}$$
$$\Phi_q(x,u,\Xi_1,...,\Xi_{q-1},\Xi_q,p) = 0. \tag{18}$$

If the original model is well-posed, the Jacobian $\partial\Phi/\partial\Xi$ of the system equations after index reduction is non-singular, and it is possible to locally solve (10) for Ξ, thanks to the implicit function theorem. Given the block-triangular structure of the system, it is then possible (at least in principle) to locally solve equations (15)-(18) in sequence, thus conceptually bringing the system into the form

$$\Xi_1 = \Psi_1(x,u,p) \tag{19}$$
$$\Xi_2 = \Psi_2(x,u,\Xi_1,p) \tag{20}$$
$$\Xi_3 = \Psi_3(x,u,\Xi_1,\Xi_2,p) \tag{21}$$
$$... \tag{22}$$
$$\Xi_q = \Psi_q(x,u,\Xi_1,...,\Xi_{q-1},p) = 0. \tag{23}$$

It is then possible to collect all the input arguments of each $\Psi_j(\cdot)$ function (except the uncertain parameters p) in a vector X_j for convenience

$$X_j = \begin{bmatrix} x \\ u \\ \Xi_1 \\ ... \\ \Xi_{j-1} \end{bmatrix}. \tag{24}$$

Equations stemming from object-oriented models are usually characterized by a high sparsity degree, so that each function $\Psi_j(X_j)$ will only depend on a few elements of the vector X_j. In order to obtain a more compact formulation, it is then possible to re-define the functions $\Psi_j(\cdot)$ so that they are a function of the vectors Σ_j, which only contain those elements of X_j that appear explicitly as input arguments of the function

$$\Xi_j = \Psi_j(\Sigma_j,p), \quad j = 1,...,q. \tag{25}$$

The relationship between Σ_j and X_j is given by suitable selection matrices S_j (i.e., matrices containing just zeros and ones)

$$\Sigma_j = S_j X_j. \tag{26}$$

It now possible to cast the DAE problem (10) in the recursive form

$$X_1 = \begin{bmatrix} x \\ u \end{bmatrix}, \quad X_{j+1} = \begin{bmatrix} X_j \\ \Psi_j(S_j X_j, p) \end{bmatrix}, \quad j = 1,...,q, \tag{27}$$

which can be interpreted in terms of a block diagram, as shown in Fig. 2.

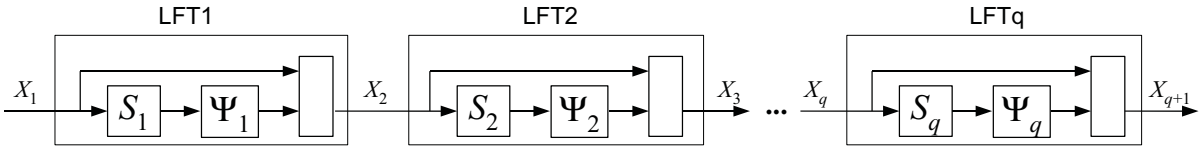

Figure 2: Block diagram representation of the recursive LFT definition.

3.4 Construction of the LFT representation

The functional relationships expressed by the functions $\Psi_j(\Sigma_j, p)$ can be either linear or nonlinear with respect to the input argument Σ_j.

In the case of a linear relationship, whose coefficients are rational functions of the uncertain parameters p, the Ψ_j block can be interpreted as an instance of an LFT system such as (1)-(6) with a suitable Δ block, with neither dynamics nor nonlinear function feedback blocks.

In the case of a nonlinear relationship, assuming that it does not involve the uncertain parameters p, this can be represented as an instance of the LFT (1)-(6) with the $\Psi_j(\Sigma_j)$ function as the nonlinear function block $\Theta(\cdot)$, with neither dynamics or Δ blocks, and with the trivial equations

$$\omega = u \tag{28}$$

$$y = \zeta. \tag{29}$$

Remark 3.1 *It is important to point out at this stage that there are at least two situations in which the outlined algorithm for LFT contruction will abort: the first one occurs if the dependency on the parameters is not rational; the second one corresponds to those situations in which a nonlinear relationship includes some of the uncertain parameters p. A more detailed discussion of such critical cases and of possible remedies is provided in Section 6.*

It is then possible to build the LFT from the inputs $X_1 = \begin{bmatrix} x \\ u \end{bmatrix}$ to the outputs $X_q = \begin{bmatrix} x \\ u \\ \Xi \end{bmatrix}$ by recursive series connection and vertical vector concatenation, as suggested by equation (27) and by Fig. 2. This operation is supported by tools such as the LFR toolbox (see [15], Chapter 7 for details). The LFT obtained so far will contain a Δ block and a Θ block, but no dynamics block yet.

It is now possible to obtain the LFT from the inputs $\tilde{u} = \begin{bmatrix} x \\ u \end{bmatrix}$ to the outputs $\tilde{y} = \begin{bmatrix} \dot{x} \\ y \end{bmatrix}$ by selecting the the rows of matrices D_{31}, D_{32}, and D_{33} that correspond to the elements of $\dot{x}$ and y within X_q, and re-ordering them accordingly. Finally, it is possible to apply order-reduction algorithms, such as those described in [8, 2, 3], and available in the LFR Toolbox [15], in order to obtain equivalent LFT representations having the Δ block as small as possible.

After these steps have been performed, the resulting LFT will have the structure:

$$z = D_{11}w + D_{12}\zeta + D_{13}\tilde{u} \tag{30}$$

$$\omega = D_{21}w + D_{22}\zeta + D_{23}\tilde{u} \tag{31}$$

$$\tilde{y} = D_{31}w + D_{32}\zeta + D_{33}\tilde{u} \tag{32}$$

$$w = diag\left\{\delta_1 I_{r_1}, ..., \delta_q I_{r_q}\right\} z \tag{33}$$

$$\zeta = \Theta(\omega). \tag{34}$$

The system (30)-(34) can now be brought into standard form (1)-(6) by splitting D_{13} and D_{23} column-wise according to

$$D_{13} = \begin{bmatrix} D_{131} & D_{132} \end{bmatrix}, \quad D_{23} = \begin{bmatrix} D_{231} & D_{232} \end{bmatrix}, \tag{35}$$

where D_{131} and D_{231} contain, respectively, the first n columns (corresponding to x) of D_{13} and D_{23}, and D_{132} and D_{232} are formed with the remaining m_u columns (corresponding to u).

Similarly, the D_{31} and D_{32} matrices are each split as

$$D_{31} = \begin{bmatrix} D_{311} \\ D_{312} \end{bmatrix}, \quad D_{32} = \begin{bmatrix} D_{321} \\ D_{322} \end{bmatrix}, \tag{36}$$

where D_{311} and D_{321} contain the first n rows (corresponding to $\dot{x}$) of D_{31} and D_{32}, respectively, while D_{312} and D_{322} are given by the remaining m_y rows (corresponding to y). Finally, matrix D_{33} is split into a 2×2 block matrix, with a partitioning consistent with the previous ones

$$D_{33} = \begin{bmatrix} D_{3311} & D_{3312} \\ D_{3321} & D_{3322} \end{bmatrix}. \tag{37}$$

The LFT system can thus be written in standard form, including the dynamic feedback block, by re-arranging the matrices according to

$$\dot{x} = D_{3311}x + D_{311}w + D_{321}\zeta + D_{3312}u \tag{38}$$
$$z = D_{131}x + D_{11}w + D_{12}\zeta + D_{132}u \tag{39}$$
$$\omega = D_{231}x + D_{21}w + D_{22}\zeta + D_{232}u \tag{40}$$
$$y = D_{3321}x + D_{312}w + D_{322}\zeta + D_{3322}u \tag{41}$$
$$w = diag\left\{\delta_1 I_{r_1}, ..., \delta_q I_{r_q}\right\} z \tag{42}$$
$$\zeta = \Theta(\omega). \tag{43}$$

4 Implementation of the algorithm

The proposed algorithm has been implemented by using a combination of different software tools. The transformation of an object-oriented Modelica model into a flattened form is performed by the OpenModelica tool [1], which outputs the model equations after flattening, index reduction, and BLT transformation in the form of an XML file, for further processing by other tools.

The XML file is then read into MATLAB, where it is possible to perform symbolic computations by means of the Symbolic Toolbox, and to manipulate LFT objects by exploiting the features of the LFR Toolbox [15]. The LFR Toolbox provides basic functionalities for LFT manipulation, such as:

- building the LFT representation of linear systems starting from symbolic formulations of their parameter-dependent matrices (via the `lfr()` function);
- building the LFT representation of a static nonlinear function (via the `lfr()` function);
- building a normalized LFT representation, with $\delta_j = -1$ corresponding to the lower bound of the uncertain parameter, and $\delta_j = +1$ corresponding to the upper bound (via the `normalizelfr()` function);
- building the LFT corresponding to the cascaded connection of two LFTs (via the `*` operator);
- building the LFT corresponding to the vector concatenation of two LFTs (via the standard MATLAB syntax for block matrix definitions);
- building the LFT between a subset of the inputs and a subset of the outputs of a given LFT (via the slice operator);
- reducing the size of the Δ block through minimal representations (via the `minlfr()` function).

An alternative implementation is possible using Scilab/Maxima and the Scilab version of the LFT toolbox.

The implementation closely follows the procedure outlined in Section 3.4. Linear LFTs are built by defining symbolic objects for the variables and parameters of the corresponding equations in the $\Phi_j(\cdot)$ block, via the `lfrs` command; the equations are then symbolically solved for the variables within Ξ_j, and the coefficient matrix of the solution, which contains symbolic expressions in the uncertain parameters, is used to build the LFT object via the `lfr()` function.

LFTs containing the nonlinear functions Ψ_j are also built using the `lfr()` function; for example, a 2×3 nonlinear function corresponding to block Ψ_3 can be defined as `lfr('Psi_3','nlmfr',[2,3])`. Note that the LFR Toolbox *per se* cannot do anything with the nonlinear function block, which is just a black box identified by a tag, `Psi_3` in this case. The tag is associated with the corresponding block of nonlinear equations Φ_j. Depending on the particular application of the LFT (e.g., for control system analysis and design or for system identification), by means of further symbolic processing, it is possible to try to explicitly solve the equations of the block in the form $\Xi_j = \Psi_j(\Sigma_j)$, and/or to linearize them, and so on. Alternatively, the nonlinear blocks Φ_j might be scanned by a human reader, who will recognize their content and tag each block with the relevant properties (e.g., min/max slope, min/max bounds, etc.).

The core of implementation is the recursive construction of the LFT representation, corresponding to equations (27). The input-output relationship between X_1 and X_{q+1} is easily built step by step by using the above-mentioned functionalities of the LFR toolbox. For numerical reasons, the LFT is also normalized, so that a value of -1 in the Δ block corresponds to the lower bound of the parameter, and a value of $+1$ corresponds to the upper bound.The corresponding MATLAB code reads:

```
LFT = lfr(eye(n+m_u));
for j =1:q
  LFT = [lfr(eye(n+m_u+sum(n_j[1:j-1]))); normalizelfr(Psi{j}*S{j})]*LFT;
end
```

5 Examples

5.1 Example 1: a block diagram

The first example is a simple block diagram (see Fig. 3), built by the cascade connection of a first order linear system with transfer function $G(s) = 4/(1+p_1 s)$, a block with gain p_2, a sine block, and a block with gain $1/p_2$, so that p_2 appears twice in the model. The model has one state variable, one input, one output and two uncertain parameters p_1 and p_2. The bounds for the uncertain parameters are: $1 < p_1 < 3$ and $3 < p_2 < 5$.

After applying the algorithm, the model is brought in standard form (38)-(43), with a 3×3 normalized block $\Delta = diag(p_1, p_2, p_2)$ and with a 1×1 Θ block containing the sine function. The values of the matrices are:

$$\begin{array}{cccc} D_{3311} = -0.5, & D_{311} = \begin{bmatrix} -0.71 & 0 & 0 \end{bmatrix}, & D_{321} = 0, & D_{3312} = 0.5 \\ D_{131} = \begin{bmatrix} -0.35 \\ 0 \\ -4 \end{bmatrix}, & D_{11} = \begin{bmatrix} -0.5 & 0 & 0 \\ 0 & -0.25 & 0 \\ 0 & 0 & 0 \end{bmatrix}, & D_{12} = \begin{bmatrix} 0 \\ 0.25 \\ 0 \end{bmatrix}, & D_{132} = \begin{bmatrix} 0.35 \\ 0 \\ 0 \end{bmatrix} \\ D_{231} = 16, & D_{21} = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}, & D_{22} = 0, & D_{232} = 0 \\ D_{3321} = 0, & D_{312} = \begin{bmatrix} 0 & -0.25 & 0 \end{bmatrix}, & D_{322} = 0.25, & D_{3322} = 0 \end{array} \tag{44}$$

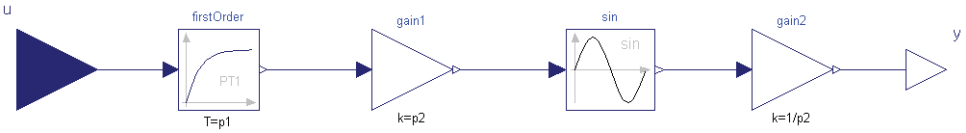

Figure 3: Block diagram of Example 1.

5.2 Example 2: a mechanical system

The second example is a simple mechanical system with two rotational degrees of freedom, represented in Fig. 4. The system is built by the connection of an ideal torque generator, a first rotating body with coefficient of inertia $J = p_1$, a nonlinear spring, a second rotating body with coefficient of inertia $J = p_2$, and an angle sensor. The nonlinear spring is described by the following equations:

$$\tau = \tau_l(1+\alpha) \tag{45}$$

$$\tau_l = -K\varphi \tag{46}$$

$$\alpha = \frac{\varphi^2}{\varphi_0^2}, \tag{47}$$

where τ is the torque applied by the end flanges, τ_l is the linear torque component, φ is the relative angle between the two flanges, K is the elastic coefficient, and φ_0 is the characteristic coefficient of the nonlinear effect. The model has four state variables, one input, one output, and two uncertain parameters p_1 and p_2 with bounds $10 < p_1 < 15$ and $2 < p_2 < 4$, while $K = 100$ and $x_0 = 0.5$.

After applying the algorithm, the model is brought in standard form (38)-(43), with a 2×2 normalized block $\Delta = diag(p_1, p_2)$ and with a 2×4 block Θ, which is partitioned into the two following nonlinear functions:

$$\zeta_1 = (\omega_1 - \omega_2)^2/0.5 \tag{48}$$

$$\zeta_2 = \omega_3(1+\omega_4) \tag{49}$$

The values of the matrices are:

$$\begin{array}{cccc} D_{3311} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, & D_{311} = \begin{bmatrix} 0 & 0 \\ 0.11 & 0 \\ 0 & 0 \\ 0 & 0.33 \end{bmatrix}, & D_{321} = \begin{bmatrix} 0 & 0 \\ 0 & 0.08 \\ 0 & 0 \\ 0 & -0.33 \end{bmatrix}, & D_{3312} = \begin{bmatrix} 0 \\ 0.08 \\ 0 \\ 0 \end{bmatrix} \\ D_{131} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, & D_{11} = \begin{bmatrix} -0.2 & 0 \\ 0 & -0.33 \end{bmatrix}, & D_{12} = \begin{bmatrix} 0 & -0.14 \\ 0 & 0.33 \end{bmatrix}, & D_{132} = \begin{bmatrix} -0.14 \\ 0 \end{bmatrix} \\ D_{231} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -100 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, & D_{21} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, & D_{22} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}, & D_{232} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ D_{3321} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}, & D_{312} = \begin{bmatrix} 0 & 0 \end{bmatrix}, & D_{322} = \begin{bmatrix} 0 & 0 \end{bmatrix}, & D_{3322} = 0 \end{array} \tag{50}$$

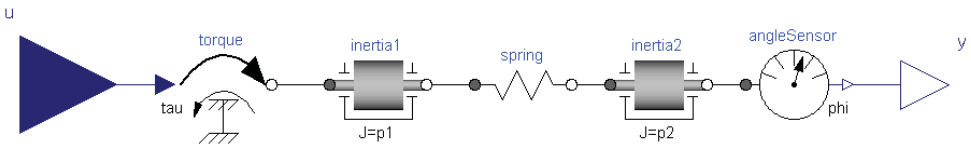

Figure 4: Object diagram of Example 2.

6 Extensions

6.1 Non-rational dependency from uncertain parameters

If the coefficients of some linear equation block are not rational functions of the uncertain parameters, it is impossible to represent the corresponding linear relationships as LFTs, and the algoritm presented in Section 3.4 will fail. It is possible to think of extensions of this algorithm, that could approximate non-rational expressions involving uncertain parameters by means of rational approximations, e.g., polynomial Taylor expansions around a nominal value. The nominal value could be supplied by the *nominal* attribute of the parameter, as defined in the Modelica language, and the series expansion might be performed by suitable symbolic manipulation.

6.2 Nonlinear parameter-dependent functions

The object-oriented model might contain nonlinear equations which also involve uncertain parameters. This would cause the basic algorithm presented in Section 3.4 to fail. In most cases, though, it would be possible to deal with this case just by defininining new auxiliary variables. For example, consider the following case, where the block Φ_3 has the form

$$\Phi_3 = v_3 - p_1 \sin(p_2 v_1 + v_2). \tag{51}$$

It is apparent that if two new variables v_4 and v_5 are defined by means of the two extra blocks

$$\Phi_4 = v_4 - p_2 v_1 + v_2 \tag{52}$$
$$\Phi_5 = v_5 - \sin(v_4), \tag{53}$$

then it is it would be possible to rewrite equation (51) as

$$\Phi_3 = v_3 - p_1 v_5. \tag{54}$$

Applying now the BLT algorithm to the new system of equations, it will be split into the following sequence of two scalar linear equations with uncertain parameters and one scalar nonlinear equation with no uncertain parameters

$$\Phi_4 = v_4 - p_2 v_1 - v_2 \tag{55}$$
$$\Phi_5 = v_5 - \sin(v_4) \tag{56}$$
$$\Phi_3 = v_3 - p_1 v_5. \tag{57}$$

Such kind of analysis requires to formulate the nonlinear expressions containing uncertain parameters in binary tree form (as routinely done by compilers) and then to define new auxiliary variables and equations for each node in the binary tree, thus re-formulating the nonlinear expression in terms of the auxiliary variables.

7 Conclusions

The problem of obtaining reduced-order LFT models starting from generic equation-based, object-oriented descriptions of nonlinear plant dynamics, with uncertain parameters has been considered in this paper. Under suitable assumptions, an algorithm achieving this goal has been presented and illustrated by means of examples. Open problems and future extensions have been outlined and discussed.

8 References

[1] OpenModelica home page. URL: `http://www.ida.liu.se/labs/pelab/modelica/OpenModelica.html`.

[2] C. Beck and R. D'Andrea. Minimality, controllability and observability for uncertain systems. In *Proceedings of the 1997 American Control Conference*, pages 3130–3135, Albuquerque, USA, 1997.

[3] C. Beck and J. Doyle. A necessary and sufficient minimality condition for uncertain systems. *IEEE Transactions on Automatic Control*, 44(10):1802–1813, 1999.

[4] T. Bohlin. *Practical Grey-box Process Identification: Theory and Applications*. Springer-Verlag, 2006.

[5] F. Casella, F. Donida, and M. Lovera. Automatic generation of LFTs from object-oriented Modelica models. In *Proceedings of the 2nd IEEE Multi-conference on Systems and Control (Late News Paper)*, San Antonio, USA, 2008.

[6] F. Casella, F. Donida, and M. Lovera. Beyond simulation: Computer-aided control systems design using equation-based object oriented modelling for the next decade. In *Proceedings of the 2nd International Workshop on Equation-Based Object-Oriented Languages and Tools*, Paphos, Cyprus, 2008.

[7] F. Casella and M. Lovera. LPV/LFT modelling and identification: overview, synergies and a case study. In *Proceedings of the 2nd IEEE Multi-conference on Systems and Control*, San Antonio, USA, 2008.

[8] R. D'Andrea and S. Khatri. Kalman decomposition of linear fractional transformation representations and minimality. In *Proceedings of the 1997 American Control Conference*, pages 3557–3561, Albuquerque, USA, 1997.

[9] F. Demourant and G. Ferreres. Closed loop identification of a LFT model. *Journal Européen des Systémes Automatisés*, 36(3):449–464, 2002.

[10] I. S. Duff and J. K. Reid. An implementation of Tarjan's algorithm for the block triangularization of a matrix. *ACM Transactions on Mathematical Software*, 4(2):137–147, 1978.

[11] P. Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley, 2003.

[12] K. Hsu, T. Vincent, G. Wolodkin, S. Rangan, and K. Poolla. An LFT approach to parameter estimation. *Automatica*, 44(12):3087–3092, 2008.

[13] L. Lee and K. Poolla. Identification of linear parameter-varying systems using nonlinear programming. *Journal of Dynamic Systems, Measurement and Control - Transactions of the ASME*, 121(1):71–78, 1999.

[14] L. Ljung. Perspectives on system identification. In *2008 IFAC World Congress, Soul, South Korea*, 2008.

[15] J.-F. Magni. User manual of the Linear Fractional Representation Toolbox, version 2.0. Technical Report TR 5/10403.01F DCSD, ONERA, Feb 2006. URL: `http://www.cert.fr/dcsd/idco/perso/Magni/download/lfrt_manual_v20.pdf`.

[16] S. E. Mattsson, H. Elmqvist, and M. Otter. Physical system modeling with Modelica. *Control Engineering Practice*, 6(4):501–510, 1998.

[17] S. E. Mattsson and G. Söderlind. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM Journal on Scientific Computing*, 14(3):677–692, 1993.

[18] The Modelica Association. Modelica home page. Online. URL: `http://www.modelica.org/`.

[19] The Modelica Association. Modelica - A unified object-oriented language for physical systems modeling - Language specification version 3.0. Online, Sep. 5 2007. URL: `http://www.modelica.org/news_items/documents/ModelicaSpec30.pdf`.

[20] G. Wolodkin, S. Rangan, and K. Poolla. An LFT approach to parameter estimation. In *Proceedings of the 1997 American Control Conference*, Albuquerque, USA, 1997.

[21] K. Zhou, J. U. Doyle, and K. Glover. *Robust and optimal control*. Prentice-Hall, New Jersey, 1996.