29th Congress of the International Council of the Aeronautical Sciences
St. Petersburg, Russia; September 7-12, 2014

A COMPARISON OF TECHNIQUES TO GET SPARSE RATIONAL APPROXIMATIONS FOR LINEAR FRACTIONAL REPRESENTATIONS

C. Roos*, G. Hardier*, C. Döll*
***ONERA The French Aerospace Lab, Systems Control and Flight Dynamics Department**
Toulouse, France, email: georges.hardier@onera.fr

Keywords: *Linear Fractional Representations, Surrogate Models, Evolutionary Algorithms*

Abstract

The objective of this paper is to stress that the size of a Linear Fractional Representation (LFR) significantly depends on the way tabulated or irrational data are approximated during the prior modeling process. It is notably shown that rational approximants can result in much smaller LFR than polynomial ones. Accordingly, 2 new methods are proposed to generate sparse rational models, which avoid data overfitting and lead to simple yet accurate LFR. The 1st one builds a parsimonious modeling based on surrogate models and a new powerful global optimization method, and then translates the result into a fractional form. The 2nd one looks for a rational approximant in a single step thanks to a symbolic regression technique, and relies on Genetic Programming to select sparse monomials. This work takes place in a more general project led by ONERA/DCSD and aimed at developing a Systems Modeling, Analysis and Control Toolbox (SMAC) for Matlab©.

1 General Introduction

A Linear Fractional Representation (LFR) is a model where all known and fixed dynamics of a given system are put together in a linear time-invariant plant M, while the uncertain and varying parameters are stored in a perturbation matrix Δ (Fig. 1). LFR modeling is a widely spread and a very efficient tool in the fields of system analysis and control design. It notably allows the robustness properties of uncertain closed-loop plants to be evaluated (e.g. using μ-analysis or Lyapunov-based methods), and to design robust control laws (especially using H_∞ approaches) or gain-scheduled controllers [40]. But the efficiency of those analysis and synthesis techniques strongly depends on the complexity of the considered LFR, which is measured in terms of both the size of the matrix Δ and the order of the plant M. An increase in complexity is usually source of conservatism, and can even lead to numerical intractability.

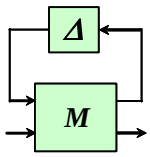

Fig. 1. Linear Fractional Representation

In most industrial applications, physical systems are described using a mix of nonlinear analytical expressions and tabulated data. Therefore, a two-step procedure has to be implemented to obtain a suitable LFR: a linear model with a rational dependence on the system parameters is first generated, and then converted into a linear fractional form. Several techniques such as object-oriented realization exist to perform the latter transformation. Although the minimality of the resulting LFR cannot be guaranteed, symbolic preprocessing techniques, as well as numerical reduction, usually permit to overcome complexity. Efficient software such as the LFR Toolbox for Matlab© is also available (see [21] and references therein for a comprehensive overview of LFR modeling). On the other hand, the preliminary issue of converting the tabulated or irrational data into simple yet accurate rational expressions has been paid much less attention, although it is of significant practical importance. In the aeronautic field for example, most aircraft models include tabulated aerodynamic coefficients determined by CFD, wind tunnel experiments or flight tests, and several controller gains depend on the flight parameters in a tabulated fashion.

The motivations for addressing the issue of

tabulated data approximation in this paper are twofold. The first one is of physical nature. Computing rational expressions with sparse structure, for which the number of terms in the numerator and denominator is as low as possible, is a natural way to prevent data overfitting and to ensure a smooth behavior of the model between the points used for approximation. On the other hand, building an LFR from a polynomial or a rational expression $f(x^1,\ldots,x^n)$ results in a block diagonal matrix $\Delta = \mathrm{diag}[x^1 I_{p_1},\ldots,x^n I_{p_n}]$. The number p_j of repetitions of each parameter x^j in Δ is strongly linked to the number of occurrences of x^j in *f*. Indeed, although this is not an exact rule, the trend is as follows: the fewer the occurrences of x^j in $f(x^1,\ldots,x^n)$, the smaller the size of Δ. In other words, no matter how efficient the LFR generation tools can be, they are of little help if the rational expressions to be converted are unnecessarily complex. Hence, the need to get tractable LFR for analysis and design purposes is another strong motivation for generating sparse rational expressions.

For a given accuracy, an intuitive idea is to determine a rational function for which the numerator *P* and denominator *Q* are two polynomials of the lowest possible degrees. This fairly simple strategy is followed by most existing methods. A classical linear least-squares (LS) technique is notably implemented in the LFR Toolbox [21] in case the rational function is restricted to be polynomial. In the general case, a nonlinear LS technique, implemented for example in the Curve Fitting Toolbox of Matlab©, tries to minimize the approximation error, whereas a Quadratic Programming problem solved by [5] ensures that the resulting rational function intersects a set of intervals containing the data. But all these techniques suffer from the same drawback: all admissible monomials of *P* and *Q* are usually nonzero, regardless of their real ability to model the data. More generally, the question of which terms should be included in the model is often addressed by trial-and-error, or even ignored in practice. A way to deal with this question is to use Orthogonal LS (OLS), which allows to evaluate the ability of each monomial to efficiently model the data and therefore to select only the most relevant ones, leading to sparse expressions. This approach was applied by [24] to model aeronautical data with polynomials, but practical methods leading to rational expressions are still missing. Yet, the additional degrees of freedom offered by such expressions could lead to simpler expressions and thus to smaller LFR.

In this context, the main contribution of this paper is to propose 2 new methods to compute rational expressions with sparse structure and as few monomials in *P* and *Q* as possible. The 1st method relies on an indirect approach that builds a sparse model based on neural networks at first, before translating the result into the final fractional form. A stepwise selection algorithm is used, combining the benefits of forward OLS to estimate the regression parameters with a new powerful global optimization algorithm to determine the best location of the regressors. The 2nd method performs the data approximation by building a sparse modeling in a single step thanks to a symbolic regression technique. A recent evolutionary algorithm, Genetic Programming (GP), is used to select monomials, and is coupled with a nonlinear iterative procedure to estimate the coefficients of the rational function. Besides, the resulting LFR must be well-defined in order to be used for analysis or control [40]. To ensure that its denominator has no roots in the parametric domain, a computationally efficient method based on μ-analysis is presented, which is an additional contribution of this paper.

The paper is organized as follows. The polynomial/rational approximation problem is first stated in §2 and the main existing solutions are briefly recalled. The new methods are then described in §3-4, as well as the algorithm permitting to check the non singularity of the resulting rational functions in §5. A real aeronautical example is finally presented in §6, where the new methods are compared to the existing ones w.r.t. both the accuracy and LFR complexity.

2 Problem Statement and Baseline Solutions

Let $\{y_k, k\in[1,N]\}$ be a set of samples (measurements, tabulated data...) corresponding to several parametric configurations $\{x_k, k\in[1,N]\}$ of a given system. More precisely, each $x_k = [x_k^1,\ldots,x_k^n] \in \mathcal{R}^n$ contains the values of the *n* explanatory variables for which the sample $y_k \in \mathcal{R}$ is obtained. The objective of this paper is to compute a rational function $f:\mathcal{R}^n \to \mathcal{R}$ of reasonable complexity which approximates these

data, i.e. such that $f(x_k)$ is close to y_k for all $k \in [1, N]$ in the sense of a certain criterion (see below). The main existing approaches are briefly recalled below and will be referred to as the baseline solutions (BS) in the sequel.

Remarks: The case where an analytical expression $f_A : \mathcal{R}^n \to \mathcal{R}$ is available instead of N samples of $(n+1)$-tuples $\{(x_k^1, x_k^2, ..., x_k^n), y_k\}$ is not considered here (see e.g. [27]). Moreover, this paper only deals with approximation (or regression) and not with interpolation, which would aim at finding a rational function f such that the equalities $f(x_k) = y_k$ are strictly satisfied for a large number N of samples (see [13] and references therein).

The most common approach consists in restricting f to be a polynomial one, that is:

$$f(x) = P(x) = \sum_{i=0}^{n_P} a_i \, r_i(x) \tag{1}$$

where $\{r_i, i \in [0, n_P]\}$ are polynomial regressors and $\{a_i, i \in [0, n_P]\}$ are coefficients to be determined. The issue is then to solve a linear LS problem with respect to these coefficients [21], i.e. to minimize the following criterion:

$$C = \sum_{k=1}^{N} [y_k - P(x_k)]^2 \tag{2}$$

A well-known improvement to this approach relies on a preliminary orthogonalization process to decouple the regressors. As a result, the ability of each new regressor to reduce the criterion C can be evaluated regardless of those already selected. Hence, only the most relevant ones can be considered, which amounts to a certain extent to minimize the complexity of the approximation (1) while still guaranteeing a low approximation error. This method was successfully applied by [24]. It was later improved, allowing a sparse polynomial approximant to be computed satisfying the following global and local constraints:

$$\begin{cases} \sqrt{C} \le \varepsilon_1 \\ |y_k - P(x_k)| \le \varepsilon_2 \quad \forall k \in [1, N] \end{cases} \tag{3}$$

where ε_1 and ε_2 are some user-defined positive values [10,29]. The more general case where f is extended to become a rational function is now considered:

$$f(x) = \frac{P(x)}{Q(x)} = \sum_{i=0}^{n_P} a_i \, r_i^P(x) \Big/ \sum_{i=0}^{n_Q} b_i \, r_i^Q(x) \tag{4}$$

A first method consists in solving a nonlinear LS problem with respect to the coefficients $\{a_i, i \in [0, n_P]\}$ and $\{b_i, i \in [0, n_Q]\}$, that is to minimize the following criterion:

$$C = \sum_{k=1}^{N} \left[y_k - \frac{P(x_k)}{Q(x_k)} \right]^2 \tag{5}$$

This is notably implemented in the Curve Fitting Toolbox of Matlab© [37], where several optimization tools can be used to compute a solution (Levenberg-Marquardt algorithms, trust-region methods...). One of its major drawbacks is that several local minima may exist due to the non-convexity. Hence, the results strongly depend on the initialization, which is not a trivial issue. A 2nd method was introduced by [22] in the context of polynomial approximation and then generalized by [5] to the rational case. Firstly, an uncertainty interval $[\underline{y}_k, \bar{y}_k]$ is defined around each y_k. A rational function is then determined that intersects all these intervals: $\forall k \in [1, N] \; \underline{y}_k \le P(x_k)/Q(x_k) \le \bar{y}_k$. This can be achieved by solving a Quadratic Programming problem in the coefficients $\{a_i, i \in [0, n_P]\}$ and $\{b_i, i \in [0, n_Q]\}$ with a strictly convex objective function. It will be denoted by the QP approach in the sequel.

Remark: It is usually desirable that the denominator of f has no roots in the considered parametric domain, so as to ensure that f has a smooth behaviour and that well-posed LFR are obtained [40]. Unfortunately, none of the aforementioned techniques can guarantee this.

3 Indirect Approach for Rational Modeling

Beyond the polynomial/rational expressions, the use of Surrogate Modeling becomes widespread among many scientific domains to replace the system or the reference model when this one is too restrictive for achieving some tasks like optimization, modeling, parameter identification [4,35]... Hence, a wide range of methods has been developed for building surrogate models efficiently, i.e. with both accuracy and parsimony. For example, Neural Networks (NN) are recognized nowadays as an efficient alternative for representing complex nonlinear systems, and tools are available to model static nonlinearities such as the ones encountered when formulating a problem in LFR form. The underlying idea for solving the considered approximation problem via an **indirect approach** consists in using such methods to derive a rational model. The tool

used in the sequel was developed by ONERA for A/C modeling and identification purposes and is named KOALA (*Kernel Optimization Algorithm for Local Approximation*). Due to space constraints, only its main features will be outlined in this paper (cf. [4,35] for more details).

At first, it is noteworthy that a nonlinear model can be either linear, nonlinear, or both in regard to its internal parameters. For NN, the latter corresponds to Multi-Layer Perceptrons [17], but also to Radial Basis Function Networks (RBFN) when the centers and the radii of the radial units are optimized [14]. However, the joint optimization of the whole set of model parameters (linear and nonlinear) practically results in ill-posed problems, and consequently in convergence and regularization issues. That is why Linear-in-their-Parameters models (LP) are often adopted, allowing more simple and robust algorithms. By taking advantage of their features, structural identification, i.e. determining the best set of regressors from the available data only, becomes possible in addition to parametric estimation.

To choose the unknown regressors, KOALA is based on forward selection, starting with an empty subset and adding them one at a time in order to gradually improve the approximation. To speed up that constructive process, a preliminary orthogonalization technique is used, permitting the individual regressors to be evaluated regardless of those previously recruited for the modeling [6]. In the case of local models like RBFN, choosing each regressor amounts to optimizing the kernel functions in the input space. To implement this optimization step, a global method is the best suited, and KOALA uses a new evolutionary metaheuristic known as Particle Swarm Optimization (PSO) [8]. The performance of this approach is strongly dependent on the algorithms added to the basic version of PSO (e.g., [7] uses a standard and very simple version of PSO). After a thorough literature analysis, the most promising techniques have been selected and implemented in the part of the KOALA code used to optimize the regressors' positioning.

Without going into details, a brief survey of the main functionalities is given below: ***fixed and adaptive topologies*** → from static (star, ring, Von Neumann) to dynamic ones (e.g. Delaunay neighboring) [19]; ***particle's displacement*** → standard, with constriction factor, FIPS and weighted FIPS versions of the velocity update laws [23]; ***hybrid local/global method*** → to speed up the convergence with direct search (improved Nelder-Mead, Delaunay tessellation for the initial simplex); ***multiswarm strategies*** → for competing swarms or for partitioning the search domain into several subregions [38]; ***diversity analysis*** → to provide information about the swarm dispersion and to refine the convergence tests [26]; ***swarm initialization*** → from random to low discrepancy sequences (centroidal Voronoï diagram, Hammersley) [9]; ***competitive multirun*** → to benefit from several topologies, algorithm variants and tuning; ***charged vs neutral particles*** → cooperation of particles with different physical properties [1].

The coupling of that PSO algorithm with the constructive technique detailed in [35] allows structural and parametric optimizations to be jointly performed for different types of regressors with local basis. In the case of KOALA, it is applied to RBFN but also to Local Linear Models (LLM) after some adjustments of the OLS method. LLM networks generalize RBFN [25], but are also related to other local models like Fuzzy Inference Systems. They are derived by replacing the RBF linear weightings (denoted by w in the sequel) by an affine expression depending on the model inputs. It is thus expected that fewer kernels will be required to achieve the same accuracy in most applications. For LLM, the generic formulation used to represent LP models is:

$$\hat{y}_k = f(x_k) = \sum_{j=1}^{m} [\sum_{i=0}^{n} w_{ji} x_k^i] \, r_j(x_k) = \sum_{l=1}^{mn+m} w_l \, r_l^{\#}(x_k) \quad (6)$$

where $r_j(x_k)$ represents the kernel value of the j^{th} regressor function, and with $x_k^0 = 1$ to include the constant terms of the affine modeling into the second sum. This relationship permits to recover a standard LP formulation with an extended set of regressors $r_l^{\#}$. To adapt the constructive algorithms to the kernel functions $r_l^{\#}$, the group of regressors sharing the same kernel r_j needs to be considered as a whole when adding or subtracting terms, and no more separately as it was the case for RBFN or polynomials [6,24].

KOALA aims at gradually selecting a series of regressors by optimizing their kernel parameters, i.e. the ellipsoid centers c and radii σ related to the radial unit (see (7) hereafter).

At each step of the process, the PSO particles are associated with vectors of $\mathcal{R}^{2n}$ gathering these parameters for the *n* explanatory variables. To sum up, the global performance of KOALA algorithm results from two complementary aspects: applying efficient OLS-based forward selection and Separable Nonlinear Least Squares optimization to powerful modeling (LLM) and, on the other hand, implementing a new PSO algorithm which outperforms the standard ones [35]. To give a rough idea, the use of KOALA results in a model comprising only 5 radial units in the benchmark case of §6 (for a global quadratic error $C \approx 2.5\,10^{-3}$), whereas a more standard algorithm, e.g. the one of [7], requires not less than 15 RBF units to achieve the same level of approximation. According to what is explained below, the (maximum) degree of the rational approximant can be reduced from 30 to 10 and the LFR size from 60 to 20.

Back to rational modeling, a first idea for an indirect approach capitalizing on KOALA results is to convert equation (6) *a posteriori* into a rational form. By choosing Gaussian radial functions, this regression expresses as the sum of *m* terms, the j^{th} one being for any *x*:

$$f_j(x) = [\sum_{i=0}^{n} w_{ji} x^i]\, r_j(x) = [\sum_{i=0}^{n} w_{ji} x^i]\, e^{-\sum_{i=1}^{n} \frac{(x^i - c_{ji})^2}{\sigma_{ji}^2}} \quad (7)$$

Therefore, it is possible to use Pade approximants of the exponential function, so as to replace it by a rational function in reduced form $\exp_{[p,q]}$. The latter expresses as the quotient of two polynomials of the p^{th} and q^{th} degrees, and the corresponding approximant to $f_j(x)$ becomes a rational function of the $(2p+1)^{\text{th}}$ and $2q^{\text{th}}$ degrees for every explanatory variable x^i. However, getting high quality approximants (e.g. decreasing rapidly to 0 as x^i increases) requires large values for *q* (with $q - p > 2$ or 3). Hence, the degree of the resulting rational function is penalized, with no guarantee about the accuracy of the global regression $f(x)$.

Consequently, a more relevant approach consists in replacing the exponential function straight away by such an approximant, and then to proceed to the optimization of the regression with this new kernel function. The simplest transform corresponds to the reduced form $\exp_{[0,1]}$, i.e. to the sum of *m* components like:

$$f_j(x) = \left(\sum_{i=0}^{n} w_{ji}\, x^i\right) \Big/ \left(1 + \sum_{i=1}^{n} (x^i - c_{ji})^2 / \sigma_{ji}^2\right) \quad (8)$$

This solution is preferred here and is used for the comparisons shown in §6. Hence, another class of models is added to the RBF/LLM kernels proposed by KOALA, based on the Pade approximant $\exp_{[0,1]}$. It must also be mentioned to conclude that the post-processing of the resulting regression, prior to the derivation of the LFR, makes use of the Matlab© Symbolic Toolbox. Again, several options can be considered for gathering the *m* components $f_j(x)$ into a single rational function: global expansions of numerator and denominator, factorization of the denominator, sum of elementary rational terms. The latter appears to be the most relevant since it favors some simplifications when building the final LFR. A factor of 2 can usually be gained in the final LFR size.

4 Direct Approach for Rational Modeling

When the model structure has to be determined as a whole (numerator-denominator degrees, number and type of monomials), the approximation problem cannot generally be solved by means of classical techniques. Over a few explanatory variables, a sequential and systematic exploration of the terms cannot reasonably be expected owing to the curse of dimensionnality. For example, with 2 variables and a maximum degree of 10, there are not less than 10^{15} rational candidates available! Moreover, this dual-purpose optimization (model structure & parameters) is complicated by the fact that a rational model is no more a Linear-in-its-Parameters one (LP). Fortunately, several promising techniques have recently appeared for global optimization, with the purpose of solving symbolic regression problems close to this one. This is the case of Genetic Programming, and after a short description of its main principles, the way it can be adapted to the approximation of a rational function will be examined.

GP is part of the evolutionary family, as Genetic Algorithms are (GA). It uses the same principles inspired by those of natural evolution to evolve a population of individuals randomly created, until a satisfactory solution is found. Opposite to GA, it is not based on a binary coding of information but uses a structured representation instead, as syntax trees. These parse trees appear more suited to solve structural or symbolic optimization problems, since they can have different sizes and shapes.

The alphabet used to create these models is also flexible enough to cope with different types of problems. So, it can be used to encode mathematical equations, behavior models, or computer programs. First works date back to the early 60s, but GP was really implemented and brought up to date only in the early 90s by [18], thanks also to an increase of computing power. He was able to prove the interest of GP in many application fields, and laid the foundations of a standard paradigm which did not evolve much since then. The iterative process breeds a population and transforms the individuals generation after generation by applying Darwinian mechanisms: reproduction, mutation, crossover, but also gene duplication or deletion. They are applied to the hierarchically structured trees of the individuals, comprising a set of nodes which fall into 2 categories: the set **F** of internal nodes called *functions* or *operators*, and the set **T** of tree's leaves called *terminals*.

All types of functions are acceptable: from mathematical operators ($+, -, \times, \div, \sqrt{}$,exp...), to logical, conditional (tests) or user-defined. The terminals correspond to the function arguments but can also include some internal parameters or predefined constants. The content of **T** is a central issue for the problem of a joint structural/parameter optimization. That requires the best functional structure to be discovered by choosing and arranging relevant operators from **F**, but also to rule the coefficients involved in this functional structure by adapting the numerical values of the parameters included in **T**. Such a symbolic regression extends the notion of numerical regression. To be able to discover the *right* parameter values, an extra mechanism must be added to the GP algorithm [18]. It relies on introducing *ephemeral* random constants in **T** and applying evolution mechanisms to those new kinds of terminals. Though consistent with the GP formalism, this constant creation is not efficient since tuning a single parameter mobilizes many subtrees and raises the nodes number and the tree depth considerably. With LP models, it is wiser to simplify this formulation by taking the regression parameters away and including only the explanatory variables x^i and possibly some predefined constants in **T**. Hence, the individuals are just mobilized to represent the functional relationships between the x^i. At every GP iteration, the regressor functions r_j (and their number $m \le n_P + n_Q$) are derived from the trees corresponding to any individual by analyzing the tree structure from its root. The numerical value of the regression parameters (a_j, b_j) can then be adapted afterwards, by applying any minimization technique to the squared error. Moreover, coupling GP with an OLS algorithm allows to solve the optimization of the (a_j, b_j) very efficiently [20].

It is also noteworthy that GP permits to produce polynomials by setting $\mathbf{F}=\{+,\times\}$ and $\mathbf{T}=\{x^0=1, x^1, x^2, \ldots, x^n\}$, hence restricting the regressors to simple monomials. The modeling complexity can also be controlled by penalizing some internal GP parameters like the tree depth, the number of branches/leaves, or by favoring the selection of the simplest operators. Practically, this can be achieved thanks to the fitness function which is used to handle the GP mechanisms of evolution. Similarly to what is done in ridge regression, a penalty component can be added to the fitness function to favor the simplest models and to prevent overfitting. In the framework of LP models and OLS algorithms, more elaborated regularization strategies can be implemented, e.g. using Leave-One-Out validation errors [35].

The interest in combining orthogonalization methods with symbolic optimization of the kernel functions has already been studied some years ago and has resulted in the Matlab© code GP-OLS [20]. More recently, a Toolbox has been developed (GPTIPS), permitting to encode and to adapt a LP model in a multigene symbolic regression form [33]. However, none of these tools is suitable for synthetizing a rational modeling directly, especially because the parameter optimization becomes a nonlinear process then. Consequently, the **direct approach** proposed here is somewhat different and is dedicated to the rational case. It is issued from several considerations: ❶ the rational case extends the polynomial one (structured model expressed as the quotient of two polynomials) ❷ GP is fully justified since there is no other classical option available for jointly optimizing the structure of numerator and denominator (e.g. the brute-force search of the BS does not minimize the number of monomials) ❸ numerator and denominator remain LP when considered separately, and it would be a pity not to take advantage of that. Hence, GP is clearly a promising alternative for rational modeling, but

a prior adaptation of the method is required to use it with maximum efficiency. A tool named TRACKER (*Toolbox for Rational Approximation Computed by Knowing Evolutionary Research*) was thus developed from scratch by ONERA for rational modeling, and is described below.

Each component of the rational function (numerator and denominator) is represented by a single separate chromosome which comes in a syntax tree form as usual, and a priori includes several genes. The sets **T** and **F** are chosen as for the polynomial case, and a peculiar syntax rule is defined to ensure that all the non terminal nodes located below a '×'-type node are also '×'-type nodes. This trick avoids creating useless branching which could result in splitting and multiplying some monomials. Thanks to this architecture, a gene appears as a subtree linked to the root node of its chromosome through one or several '+'-type nodes. A parse analysis of the different genes composing a chromosome also permits to avoid the creation of spurious genes by identifying and grouping them if any. Fig. 2 depicts a tree's architecture corresponding to a simple $f(x)$ example. Five genes related to the different monomials are highlighted by colors (except from constants a_0, b_0 which are integral part of the structure).

To solve the parametric optimization, i.e. to estimate the regression coefficients (a_j, b_j) of any created tree, it is suitable to implement a well-known technique, in use for identifying transfer functions in the frequency domain [30]. It consists in iteratively linearizing the expression of the quadratic cost function as:

$$\varepsilon_k \approx \frac{\left[\sum_{j=0}^{n_P} a_j\, r_j^P(x_k) - \sum_{j=1}^{n_Q} b_j\, y_k\, r_j^Q(x_k) - y_k\right]}{1+\sum_{j=1}^{n_Q} \hat{b}_j\, r_j^Q(x_k)} \qquad (9)$$

where the approximation of ε_k comes from the replacement of the parameters b_j by their most recent estimates $\hat{b}_j$ at the running iteration, and the denominator is arbitrarily normalized by choosing $b_0 = 1$. By denoting the corresponding estimate of the denominator, for any sample x_k:

$$\hat{D}(x_k) = 1 + \sum_{j=1}^{n_Q} \hat{b}_j\, r_j^Q(x_k) \qquad (10)$$

the method relies on the fact that the approximation $D(x_k) \approx \hat{D}(x_k)$ becomes fully justified when the iterative process has converged. The vector of normalized outputs can then be written as $y^* = [y_1 / \hat{D}(x_1) \cdots y_N / \hat{D}(x_N)]^T$, and the k^{th} row R_k of the regression matrix follows $\hat{D}(x_k) R_k = [1 \;\cdots\; r_1^P(x_k) \;\; r_{n_P}^P(x_k) \;\; -y_k r_1^Q(x_k) \;\cdots\; -y_k r_{n_Q}^Q(x_k)]$. Finally, the parameters to be determined $w = [a_0 \;\; a_1 \cdots a_{n_P} \;\; b_1 \cdots b_{n_Q}]^T$ are a solution of the linearized LS problem $w = (R^T R)^{-1} R^T y^*$.

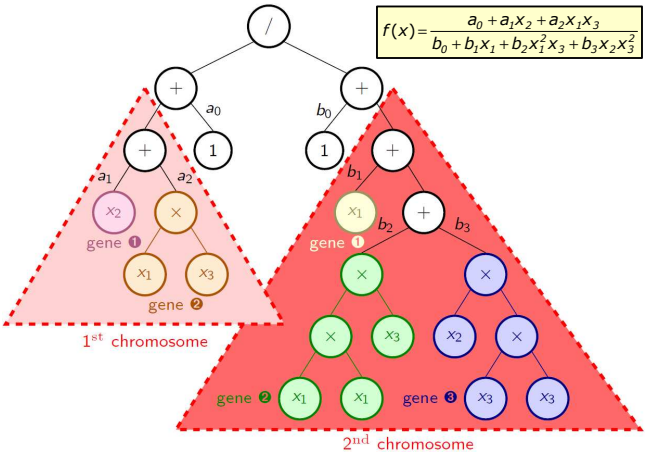

Fig. 2. Ex. of TRACKER parse tree for rational modeling

In practice, 2 or 3 iterations are usually sufficient to ensure the convergence of this process, conveniently initialized by choosing $\hat{D}(x_k) = 1$ at the first iteration. In case of ill-conditioning, a few iterations of Levenberg-Marquardt optimization are used to recover a satisfactory result. Introduced into the selection process, that technique enables to evaluate the performance of every individual very easily, by coming down to a short series of ordinary LS. The overcost remains limited because the major part of the computations required by matrix R can be stored and reused through the loop.

5 Algorithmic Aspects

An important issue is now to check that the denominator Q has no roots in the parametric domain, i.e. $Q(x) \neq 0$ for all $x \in \mathcal{D}$. Such a requirement is actually a prerequisite to build a well-defined LFR. Unfortunately, checking whether a multivariate polynomial of degree 4 or higher with real coefficients has a real zero is NP-hard in the general case [2]. A classical strategy is therefore to focus on sufficient conditions, and some methods based on parameter-dependent slack variables [32] or sum-of-squares [36] have been recently introduced. They give quite accurate results, but can become computationally demanding if some multivariate polynomials of high degrees are

considered. An efficient alternative based on μ-analysis is proposed here.

$Q(x)$ is firstly converted into an LFR according to Fig. 1, where $M \in \mathcal{R}^{(p+1)\times(p+1)}$ is a fixed matrix and $\Delta = \mathrm{diag}[x^1 I_{p_1}, \ldots, x^n I_{p_n}] \in \mathcal{R}^{p\times p}$. By breaking M up into a block form $(M_{ij})_{i,j\in[1,2]}$ according to this interconnection, the relation $Q(x) = M_{22} + M_{21}\Delta(I_p - M_{11}\Delta)^{-1}M_{12}$ holds. It is also considered without loss of generality that $\mathcal{D}$ corresponds to the unit hypercube, i.e. $x^i \in [-1,+1]$ for all $i \in [1,n]$, an assumption which can always be achieved using a suitable affine transformation. The following technical lemma is then introduced, noting that $M_{22} = Q(0)$ is nonzero (otherwise, it could be directly concluded that Q has a real zero in $\mathcal{D}$).

***Lemma*:** Let $X = M_{11} - M_{12}M_{22}^{-1}M_{21}$. Then, for all $x \in \mathcal{D}$: $Q(x) \neq 0 \Leftrightarrow \det(I_p - X\Delta) \neq 0$.

This result shows that the considered non singularity check is strongly linked to the notion of structured singular value (see [12] and references therein), the definition of which is recalled below.

***Definition*:** Let $\mathbf{\Delta}$ denote the set of all real $p \times p$ matrices with the same block-diagonal structure as Δ. If no matrix $D \in \mathbf{\Delta}$ renders the matrix $I_p - XD$ singular, then the structured singular value $\mu_{\mathbf{\Delta}}(X)$ is equal to 0. Otherwise, it is defined as the inverse of the size of the smallest $D \in \mathbf{\Delta}$ satisfying $\det(I_p - XD) = 0$:

$$\mu_{\mathbf{\Delta}}(X) = \left[\min_{D\in\mathbf{\Delta}} \left\{ \bar{\sigma}(D),\ \det(I_p - XD) = 0 \right\} \right]^{-1} \qquad (11)$$

where $\bar{\sigma}(.)$ denotes the largest singular value.

Computing the exact value of $\mu_{\mathbf{\Delta}}(X)$ is NP-hard, but polynomial-time algorithms allow both an upper bound $\bar{\mu}_{\mathbf{\Delta}}(X)$ [39] and a lower bound $\underline{\mu}_{\mathbf{\Delta}}(X)$ [34] to be computed. The following algorithm can then be derived:

- If $\underline{\mu}_{\mathbf{\Delta}}(X) > 1$, a value $\tilde{x} \in \mathcal{D}$ has been computed for which $Q(\tilde{x}) = 0$.
- If $\bar{\mu}_{\mathbf{\Delta}}(X) < 1$, it can be concluded that $Q(x) \neq 0, \forall x \in \mathcal{D}$.
- Otherwise, nothing can be assessed. A strategy is then to recursively divide $\mathcal{D}$, and to apply the procedure again on each subdomain until either a μ lower bound becomes larger than 1, or the maximal μ upper bound on all subdomains becomes smaller than 1.

Such a procedure is only heuristic, but its convergence properties appear quite good in practice. Actually, it can usually be observed that the smaller a sub-domain, the smaller the gap between the bounds (see [28] for a detailed study of such a behavior). A conclusion is thus usually obtained after only a few iterations, sometimes even after a single one.

If a worst-case value $\tilde{x} \in \mathcal{D}$ is identified, for which $Q(\tilde{x}) = 0$, the algorithms presented in §2 and §4 are applied again with the additional constraint $Q(\tilde{x}) \geq \varepsilon > 0$, where ε is a user-defined tolerance. The trick is to repeat the whole procedure until a nonsingular rational function is computed (or a maximum number of iterations is reached). Once a suitable rational function is obtained, with no poles in the considered domain, it is finally converted into an LFR. Finding a minimal order representation (i.e. for which the size of Δ is minimum) remains an open problem. Yet, efficient techniques exist to compute a reasonably low-order one, and an extra numerical reduction can be applied without altering the accuracy. All these computations can be performed using the LFR toolbox for Matlab© [21].

6 Comparison of the methods and results

Equations (12) describe the longitudinal motion of a rigid A/C [3], in body axis (x forwards and z downwards):

$$\begin{cases} mV\dot{\alpha} = mVq - \dfrac{\rho SV^2}{2} C_L - F_{eng}\sin\alpha + mg\cos\gamma \\ I_{yy}\dot{q} = \dfrac{\rho SV^2}{2}[LC_M + \delta x(C_L\cos\alpha + C_D\sin\alpha)] + \delta z\, F_{eng} \end{cases} \qquad (12)$$

The flight parameters are the Angle of Attack α (AoA), the pitch rate q, the airspeed V, and the flight path angle γ. The *constants* are denoted by g (gravity), ρ (air density), m (A/C mass), S (reference surface), I_{yy} (lateral y-axis inertia), and L (mean aerodynamic chord). F_{eng} is the thrust, whereas $\delta x = x_{ref} - x_{cg}$ and $\delta z = z_{eng} - z_{ref}$ represent the distances between the aerodynamic reference point and the centre of gravity x-location or the engine z-location.

C_L, C_D, C_M represent the aerodynamic coefficients relative to the lift, drag and pitching moments. They are usually obtained as nonlinear look-up tables during wind tunnel tests [4]. In order to translate equations (12) in fractional form, these tabulated data have to be replaced by polynomial or rational expressions, which can theoretically be achieved using any of the previous approximation methods. This is illustrated in the sequel for the drag coefficient

C_D of a generic fighter aircraft model [10]. The reference data depend on both Mach number Ma and AoA α (in radians), and are depicted on a fine 50x90 grid in Fig. 3 which is used as a validation set to evaluate the approximation results achieved with a rougher 40x60 grid of learning data.

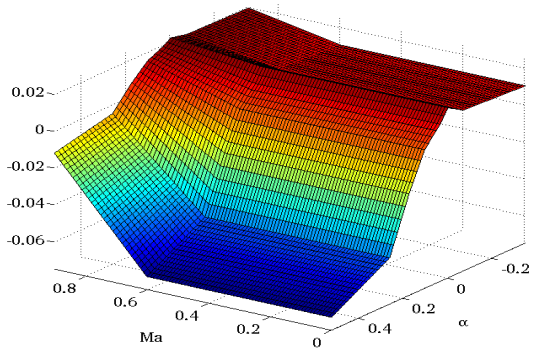

Fig. 3. Drag coefficient displayed on the validation grid

The two approaches proposed for rational approximation are compared with the Quadratic Programming based approach of §2 (QP), that gave the best results so far amongst the so-called baseline solutions. All results are gathered in Table 1, and some are also displayed in Fig. 4-5. For the Surrogate Modeling based algorithm of §3 (SM), the number of monomials is given both for the factorized expression directly obtained with KOALA (without brackets) and for the underlying expanded form (between brackets).

GP has the advantage that rational approximants with sparse structures are obtained. Only a few monomials are actually nonzero, which results in low-order LFR. Moreover, good numerical properties are observed, and significantly higher degrees can be considered than with QP. For a given degree, SM and QP give quite similar results regarding the number of monomials. Indeed, both methods generate rational functions for which the numerator/denominator are composed of almost all admissible monomials when written in expanded form. But SM offers two major pros. First, the size of the resulting LFR is smaller, since the symbolic expression does not appear as a single expanded rational function but is already factorized as a sum of elementary components. Besides, SM is numerically much more efficient and allows higher degree approximations to be computed very quickly (some seconds) and easily. This is not possible with QP, since numerical troubles appear for degrees larger than 10, thus leading to poor results.

GP and SM thus appear to be the most promising methods on this example. Moreover, these methods prove quite complementary, as far as the trade-off between parsimony and accuracy is considered. SM provides very accurate approximations, which do not have a sparse structure but are directly factorized in a compact form resulting in low order LFR. On the other hand, GP directly selects the most relevant monomials to generate very sparse symbolic expressions. It also appears that GP is more accurate for low degree approximations, while SM gives better results for degrees larger than 10. Actually, at lower degrees, the number of radial units used by SM (half the required degree) is not sufficient to represent the shape of the reference data accurately enough. Hence, a minimum number of radial basis functions is required to get the most out of this method.

Method	Degree	Monomials	RMSE	Max error	LFR size
QP	6	56	$9.73\ 10^{-4}$	$1.91\ 10^{-3}$	17
	8	90	$5.80\ 10^{-4}$	$1.27\ 10^{-3}$	23
	10	132	$4.57\ 10^{-4}$	$1.22\ 10^{-3}$	29
	12	182	$4.56\ 10^{-4}$	$1.21\ 10^{-3}$	35
GP	6	34	$9.46\ 10^{-4}$	$4.86\ 10^{-3}$	15
	8	31	$7.85\ 10^{-4}$	$4.47\ 10^{-3}$	17
	10	29	$6.92\ 10^{-4}$	$3.47\ 10^{-3}$	21
	12	36	$6.61\ 10^{-4}$	$3.23\ 10^{-3}$	26
	14	43	$6.53\ 10^{-4}$	$3.05\ 10^{-3}$	28
SM	6	18 (47)	$1.27\ 10^{-3}$	$6.24\ 10^{-3}$	12
	8	24 (79)	$7.87\ 10^{-4}$	$5.12\ 10^{-3}$	16
	10	30 (119)	$6.74\ 10^{-4}$	$3.46\ 10^{-3}$	20
	12	36 (167)	$5.44\ 10^{-4}$	$2.56\ 10^{-3}$	24
	14	42 (223)	$4.10\ 10^{-4}$	$2.12\ 10^{-3}$	28
	16	48 (287)	$2.81\ 10^{-4}$	$1.38\ 10^{-3}$	32
	24	72 (623)	$1.83\ 10^{-4}$	$1.07\ 10^{-3}$	48

Table 1. Comparison of results with different approaches

Finally, it is worth noting that the computational cost is strongly in favour of SM and QP. As the degree of the rational function increases, the Darwinian mechanisms of evolution involved by Genetic Programming require more generations to produce very accurate solutions, and the CPU time is seriously impaired. Nevertheless, a parallel implementation will be provided with the next release of the SMAC/APRICOT library to address this issue when using multicore computers.

Remark: It is worth emphasizing that the non singularity of the rational functions is guaranteed in all cases thanks to the μ-analysis based test proposed in §5.

7 Conclusion and prospects

This work takes place in the framework of a more general project aimed at developing a *Systems Modeling, Analysis and Control* (SMAC) Toolbox. This Matlab/Simulink©-based library is being developed by ONERA to provide both researchers and control engineers with a complete set of tools for making the design, tuning and validation of control laws easier. More precisely, the purposes of the SMAC project are to control aeronautical vehicles throughout their whole flight domain in the presence of nonlinearities, uncertainties, external disturbances and imperfectly measured or estimated data, while obtaining strong guarantees w.r.t. the stability margins and the performance levels. A free (limited) version of SMAC can be downloaded at w3.onera.fr/smac, that includes 3 kinds of tools:

- **modeling tools** allowing the considered physical systems (usually represented in industrial context using a mix of nonlinear analytical expressions and tabulated data) to be described as a single parameterized model (typically a LFR),
- **design tools** combining robustified nonlinear dynamic inversion techniques, structured H∞ synthesis and anti-windup compensation, so as to produce simple yet powerful controllers, which can be easily implemented but do not require any interpolation as this is the case with classical gain-scheduled techniques,
- **analysis and validation tools** permitting the robustness properties of the resulting closed-loop systems to be evaluated in the presence of model uncertainties (μ-analysis), but also of time-varying parameters and hard non-linearities such as magnitude and rate saturations (IQC-based analysis).

Accordingly, the methods described in this paper belong to the 1st group of modeling tools, and are aimed at creating accurate LFR with sizes as reduced as possible in order to facilitate the subsequent use of the design and analysis tools belonging to the 2nd or 3rd group. They are implemented in the APRICOT library (*Approximation of Polynomial and Rational-type for Indeterminate Coefficients via Optimization Tools*) of the SMAC toolbox that includes a set of optimization tools to convert numerical data into simple yet accurate polynomial or rational expressions. A limited version is available at w3.onera.fr/smac/apricot. It can be applied to any sampled data with $n \leq 2$ explanatory variables and $N \leq 100$ samples. The full version has been tested on much more complicated benchmarks and has proved successful in several real-world applications. It can be obtained in the context of a close cooperation with ONERA/DSCD.

More generally, the SMAC toolbox contains a new release of the LFR toolbox [21], that implements a more powerful *lfr* object as well as additional modeling tools. It also provides the SMART library (*Skew-Mu Analysis based Robustness Tools*) [30], which collects most of the μ-analysis based algorithms developed at ONERA/DCSD during the last decade. Finally, some routines dedicated to robustness analysis in the presence of time-varying parameters and nonlinearities, as well as control-oriented tools, will also be included soon.

References

[1] Blackwell T.M. and Bentley P.J. Dynamic search with charged swarms. *GECCO*, pp. 19-26, New York, USA, 2002.

[2] Blum L., Cucker F., Shub M., and Smale S. *Complexity and real computation*. Springer, New York, 1998.

[3] Boiffier J.L. *The dynamics of flight : the equations*. John Wiley & sons, Chichester, 1998.

[4] Bucharles A. et al. An overview of relevant issues for aircraft model identification. *AerospaceLab*, Issue 4, http://www.aerospacelab-journal.org/al4, 2012.

[5] Celis O.S., Cuyt A., and Verdonk B. Rational approximation of vertical segments. *Numerical Algorithms*, 45(1-4), pp. 375-388, 2007.

[6] Chen S., Hong X., Harris C.J., and Sharkey P.M. Sparse modelling using orthogonal forward regression with PRESS statistic and regularization. *IEEE Trans. on Systems, Man and Cybernetics (B)*, 34 (2), pp. 898-911, 2004.

[7] Chen S., Hong X., Luk B.L., and Harris C.J. Non-linear system identification using particle swarm optimization tuned radial basis function models. *Int^{al} J^{al} of Bio-inspired Computation*, 1 (4), pp. 246-258, 2009.

[8] Clerc M. *Particle swarm optimization*. ISTE, London, 2006.

[9] Clerc M. Initialisations for particle swarm optimization. http://clerc.maurice.free.fr/pso/, 2008.

[10] Döll C., Bérard C., Knauf A., and Biannic J.M. LFT modelling of the 2-DOF longitudinal nonlinear aircraft behaviour. *IEEE Symposium on Computer-Aided Control System Design*, pp. 864-869, San Antonio, USA, 2008.

[11] Ferreira C. Gene expression programming: a new adaptive algorithm for solving problems. *Complex Systems*, 13(2), pp. 87-129, 2001.

[12] Ferreres G. *A practical approach to robustness analysis with aeronautical applications*. Springer, Berlin, 1999.

[13] Floater M.S. and Hormann K. Barycentric rational interpolation with no poles and high rates of approximation. *Numerische Mathematik*, 107(2), pp. 315-331, 2007.

[14] Hardier G. Recurrent RBF networks for suspension system modeling and wear diagnosis of a damper. *IEEE World Congress on Computational Intelligence*, 3, pp. 2441-2446, Anchorage, USA, 1998.

[15] Hardier G., Roos C. Creating sparse rational approximations for LFR modeling using genetic programming. *3rd IFAC Intal Conf on Intelligent Control and Automation Science*, Chengdu, China, 2013.

[16] Hardier G., Roos C. Creating sparse rational approximations for LFRs using surrogate modeling. *3rd IFAC Intal Conf. on Intelligent Control and Automation Science*, Chengdu, China, 2013.

[17] Haykin S. *Neural networks: a comprehensive foundation*. IEEE Press, MacMillan, New York, 1994.

[18] Koza J.R. and Poli R. A Genetic Programming Tutorial. In Burke ed., *Intoductory tutorials in optimization, search and decision support*, 2003.

[19] Lane J., Engelbrecht A.P., and Gain J. Particle swarm optimization with spatially meaningful neighbours. *IEEE Swarm Intelligence Symposium*, pp. 1-8, St Louis, USA, 2008.

[20] Madar J., Abonyi J., and Szeifert F. Genetic programming for the identification of nonlinear input-output models. *Industrial and Engineering Chemistry Research*, 44 (9), pp. 3178-3186, 2005.

[21] Magni J.F. *User manual of the LFR Toolbox (V 2.0)*. http://w3.onera.fr/smac/lfrt, 2006.

[22] Markov S., Popova E., Schneider U., and Schulze J. On linear interpolation under interval data. *Mathematics and Computers in Simulation*, 42, pp. 35-45, 1996.

[23] Mendes R. and Kennedy J. The fully informed particle swarm: simpler, maybe better. *IEEE Trans. on Evol. Computation*, 8 (3), pp. 204-210, 2004.

[24] Morelli E.A. and DeLoach R. Wind tunnel database development using modern experiment design and multivariate orthogonal functions. *41st AIAA Aerospace Sciences Meeting & Exhibit*, Reno, USA, 2003.

[25] Nelles O. and Isermann R. Basis function networks for interpolation of local linear models. *35th IEEE Conf. on Decision and Control*, pp. 470-475, Kobe, Japan, 1996.

[26] Olorunda O. and Engelbrecht A.P. Measuring exploration/exploitation in particle swarms using swarm diversity. *IEEE Congress on Evol. Computation*, pp. 1128-1134, Hong Kong, China, 2008.

[27] Petrushev P.P. and Popov V.A. Rational approximation of real functions. *Encyclopedia of mathematics and its applications*, Vol. 28, University Press, Cambridge, 1987.

[28] Roos C. and Biannic J-M. Efficient computation of a guaranteed stability domain for a high-order parameter dependent plant. *ACC*, pp. 3895-3900, Baltimore, USA, 2010.

[29] Roos C. Optimization based clearance of flight control laws. In Varga-Hansson-Puyou, *Generation of LFRs for a flexible aircraft model*, §4. Lecture Notes in Control and Information Sciences, Springer-Verlag, 2011.

[30] Roos C. Systems Modeling, Analysis and Control (SMAC) Toolbox: an insight into the robustness analysis library. *IEEE Multiconference on Systems and Control*, pp. 176–181, Hyderabad, India, http://w3.onera.fr/smac/smart, 2013.

[31] Sanathanan C. and Koerner J. Transfer function synthesis as a ratio of 2 complex polynomials. *IEEE Trans. on Automatic Control*, 8 (1), pp. 56-58, 1963.

[32] Sato M. Parameter-dependent slack variable approach for positivity check of polynomials over hyper-rectangle. *ACC*, pp. 5357-5362, St Louis, USA, 2009.

[33] Searson D.P., Lealy D.E., and Willis M.J. GPTIPS: an open source GP toolbox for multigene symbolic regression. *Intal Multiconference of Engineers and Computer Scientists*, Hong Kong, China, 2010.

[34] Seiler P., Packard A., and Balas G. A gain-based lower bound algorithm for real and mixed μ problems. *Automatica*, 46(3), pp. 493-500, 2010.

[35] Seren C., Hardier G., and Ezerzere P. On-line Estimation of Longitudinal Flight Parameters, *SAE AeroTech Congress and Exhibition*, Toulouse, France, 2011.

[36] Shor N.Z. Class of global minimum bounds of polynomial functions. *Cybernetics*, 23(6), pp. 731-734, 1987.

[37] The Mathworks Curve fitting toolbox user's guide, 2010.

[38] Trojanowski K. Multi-swarm that learns. *Intelligent Information Systems*, Vol. XVI, pp. 121-130, 2008.

[39] Young P.M., Newlin M.P., and Doyle J.C. Computing bounds for the mixed μ problem. *Intal Jal of Robust and Nonlinear Control*, 5(6), pp. 573-590, 1995.

[40] Zhou K., Doyle J.C., and Glover K. *Robust and optimal control*. Prentice-Hall, Upper Saddle River, 1996.

Copyright Statement

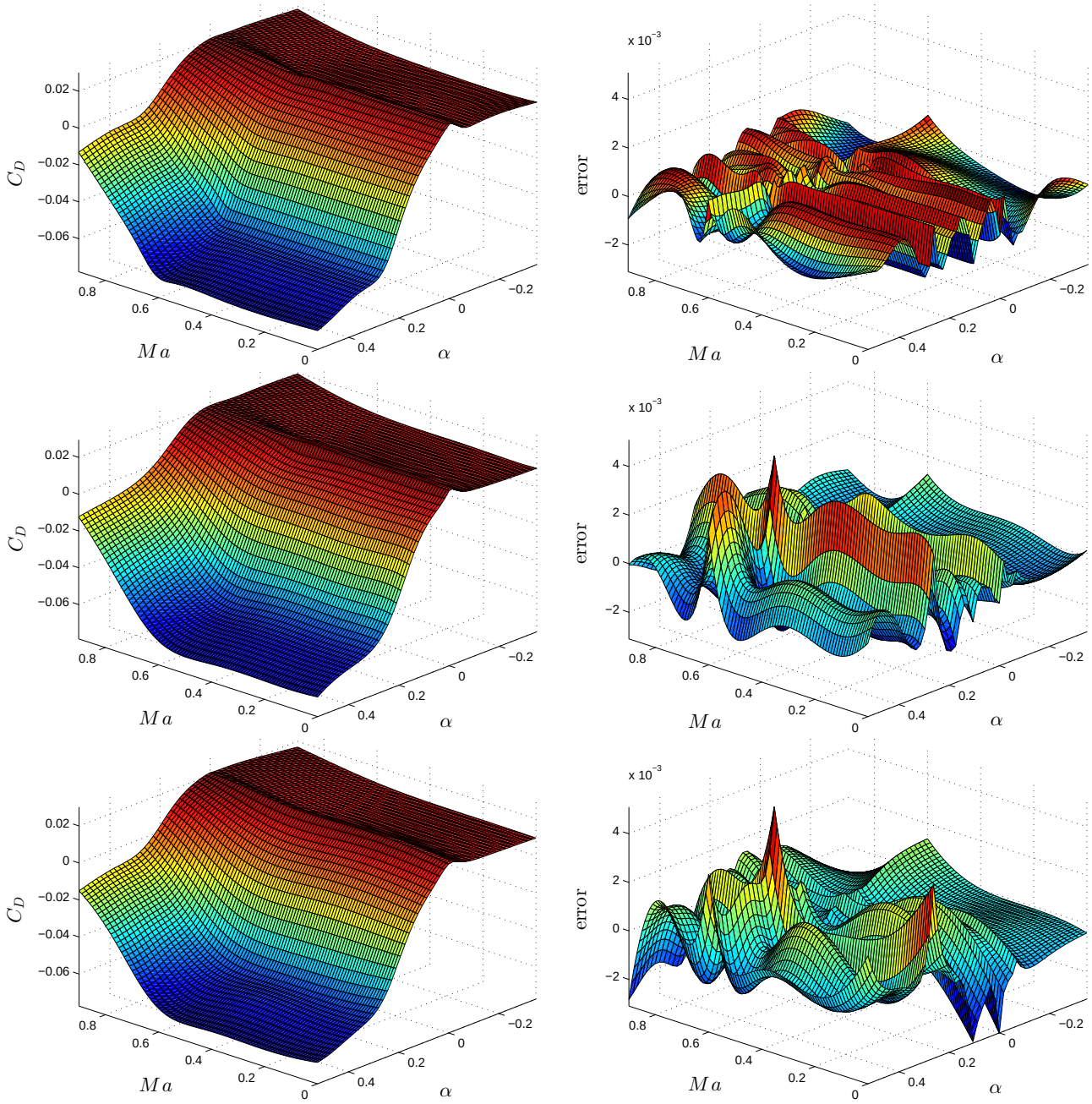

Fig. 4. Approximated coefficients of degree 8 and approximation errors (top=QP, middle=GP, bottom=SM)

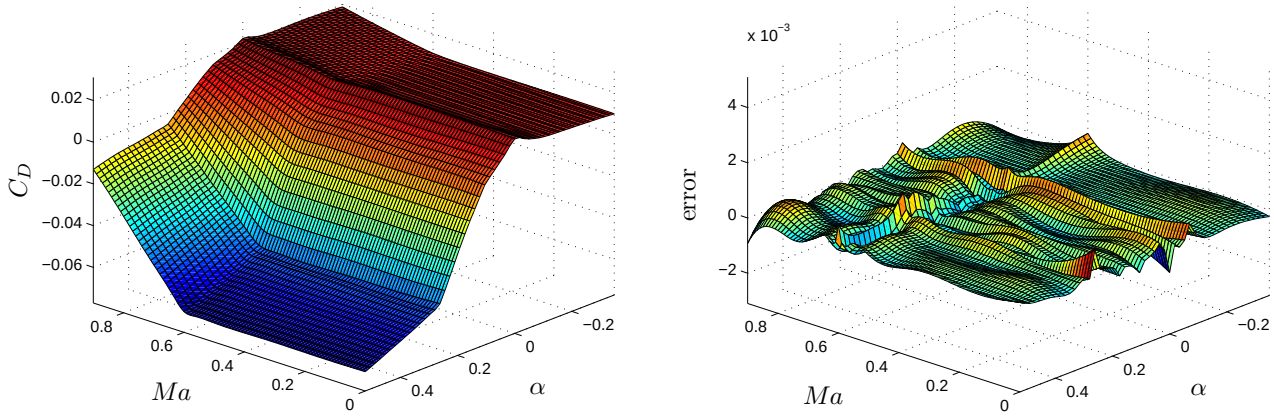

Fig. 5. Approximated coefficient of degree 24 and approximation error (SM)