Creating Sparse Rational Approximations for Linear Fractional Representations using Surrogate Modeling

G. Hardier, C. Roos, C. Seren

Creating Sparse Rational Approximations for Linear Fractional Representations using Surrogate Modeling

G. Hardier*, C. Roos*, C. Seren*

** Systems Control and Flight Dynamics Department, ONERA The French Aerospace Lab, Toulouse, France, (e-mail: Georges.Hardier@onera.fr)*

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 modeling process. It is notably shown that rational approximants can result in much smaller LFR than polynomial ones. In this context, a new method is introduced to generate sparse rational models, which avoid data overfitting and lead to simple yet accurate LFR. This method builds a parsimonious model based on neural networks, and then translates the result into a fractional form. A stepwise selection algorithm is used, combining the benefits of forward orthogonal least squares to estimate the regression parameters with a new powerful global optimization to determine the best location of the regressors. The proposed method is evaluated on an aeronautical example and successfully compared to more classical approaches.

Keywords: rational approximation, surrogate modeling, Linear Fractional Representation, stepwise regression, Radial Basis Function networks, Particle Swarm Optimization.

1. 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 to evaluate the robustness properties of uncertain closed-loop plants (e.g. using μ-analysis or Lyapunov-based methods), and to design robust control laws (especially using H_∞ approaches) or gain-scheduled controllers (*Zhou*). But the efficiency of the aforementioned 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.

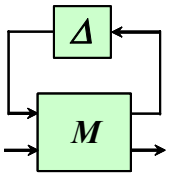

Fig. 1. Linear Fractional Representation

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 *Magni* 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 (Computational Fluid Dynamics), 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 a 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 (*Magni*) 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 solution (*Celis*) 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 *Morelli* 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 a new method to compute rational expressions with sparse structure, i.e. as few monomials in P and Q as possible. This indirect approach first builds a sparse model based on neural networks, before translating the result into a 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. Note that a direct approach computing a rational approximant in a single step thanks to a symbolic regression technique is proposed in *Hardier 2013*, which uses another recent evolutionary algorithm (Genetic Programming) to select sparse monomials.

The paper is organized as follows. The rational approximation problem is first stated in §2, and the main existing solutions are briefly recalled. The new method is then described in §3, and a real aeronautical example is finally presented in §4, where this method is compared to the existing ones in terms of both accuracy and LFR complexity.

2. PROBLEM STATEMENT & BASELINE SOLUTIONS

Let $\{y_k, k \in [1,N]\}$ be a set of samples (measurements, tabulated data...) corresponding to different parametric configurations $\{x_k, k \in [1,N]\}$ of a given system. More precisely, each $x_k = [x_k^1, \dots, x_k^n] \in \mathcal{R}^n$ 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 §4). 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 $\{(x_k^1, x_k^2, \dots, x_k^n), y_k\}$ is not considered here (see e.g. *Petrushev*). 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 *Floater* and references therein).

The most common approach consists in restricting the function 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 $\{r_i, \, i \in [0, n_P]\}$ is a set of 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 (*Magni*), 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 *Morelli*. It was later improved, allowing to compute a sparse polynomial approximant 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 integers (*Roos,Döll*). The standard LS approach and its OLS-based variant will be denoted by **BP1** and **BP2** in the sequel.

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:

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

It is notably implemented in the Curve Fitting Toolbox of Matlab©, where several optimization tools can be used to compute a solution (Levenberg-Marquardt algorithms, trust-region methods...). It will be denoted by **BR1** in the sequel. 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 second method was introduced by *Markov* in the context of polynomial approximation and then generalized by *Celis* to the rational case. Firstly, an uncertainty interval $[\underline{y}_k, \overline{y}_k]$ is defined around each y_k. A rational function is then determined which intersects all these intervals, i.e. $\forall k \in [1,N]$ $\underline{y}_k \le P(x_k)/Q(x_k) \le \overline{y}_k$. 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 **BR2** 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 (*Zhou*). Unfortunately, none of the aforementioned techniques allow to 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 (*Bucharles*, *Seren*)... 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 (see *Bucharles* and *Seren* 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 case corresponds to multilayer perceptrons (*Haykin*), but also to Radial Basis Function Networks (RBFN) when the centers and the radii of the radial units are optimized (*Hardier 2013*). 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 to evaluate the individual regressors regardless of those previously recruited for the modeling (*Chen 2004*). 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) (*Clerc 2006*). The performance of this approach is strongly dependent on the algorithms added to the basic version of PSO (e.g., *Chen 2009* 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.

A detailed description of the software is out of the scope of the paper, but 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) (*Lane*); ***particle's displacement*** → standard, with constriction factor, FIPS and weighted FIPS versions of the velocity update laws (*Mendes*); ***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 (*Trojanowski*); ***diversity analysis*** → to provide information about the swarm dispersion and to refine the convergence tests (*Olorunda*); ***swarm initialization*** → from random to low-discrepancy sequences (Hammersley, centroidal Voronoï diagram) (*Clerc 2008*); ***competitive multirun*** → to benefit from several topologies, algorithm variants and tuning; ***charged vs neutral particles*** → cooperation of particles with different physical properties (*Blackwell*).

The coupling of that PSO algorithm with the constructive technique detailed in *Seren* allows to proceed to structural and parametric optimizations jointly 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 (*Nelles*), 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:

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

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 substracting terms, and no more separately as it was the case for RBFN or polynomials (*Morelli*, *Chen 2004*).

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 the 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 (*Seren*). To give a rough idea, the use of KOALA results in a model comprising only 5 radial units in the benchmark case of §4 (for a global quadratic error $C \approx 2.5\,10^{-3}$), whereas a more standard algorithm, e.g. the one of *Chen 2009*, 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:

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

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 $(2p+1)^{th}$ and $2q^{th}$ 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 $f(x)$.

On the other hand, 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 §4. 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 two can usually be gained in the final LFR size.

4. COMPARISON OF THE METHODS AND RESULTS

Equations (9) describe the longitudinal motion of a rigid A/C (*Boiffier*), in body axis (x forwards and z downwards):

$$\begin{cases} mV\dot{\alpha} = mVq - \dfrac{\rho S V^2}{2} C_L - F_{eng}\sin\alpha + mg\cos\gamma \\ I_{yy}\dot{q} = \dfrac{\rho S V^2}{2}\left[L\, C_M + \delta x (C_L\cos\alpha + C_D\sin\alpha)\right] + \delta z\, F_{eng} \end{cases} \quad (9)$$

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. In order to translate equations (9) 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 next subsections for the drag coefficient C_D of a generic fighter aircraft model (*Döll*). The reference data depend on both Mach number Ma and AoA α (in radians). They are given on a fine 50x90 grid and are represented in Fig. 2.

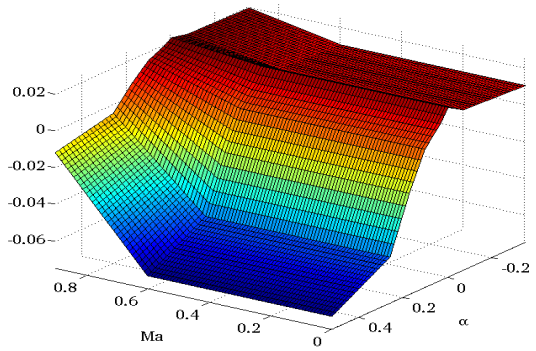

Fig. 2. Drag coefficient represented on a fine 50x90 grid

4.1 Comparison of the baseline solutions (BS)

At first, the 4 BS described in §2 are compared. The goal is to achieve the simplest possible approximation, while ensuring that the Root Mean Square Error (RMSE) between appro-ximant and reference data remains close to a given value ε on the grid. All results are gathered in Tables 1 and 2, which correspond to $\varepsilon \approx 2.10^{-3}$ and $\varepsilon \approx 9.10^{-4}$ respectively. The second and third columns specify the degree of the polynomial or rational function, as well as the total number of monomials when both numerator and denominator are expanded in the form of equation (4). The RMSE and the maximum local error $\varepsilon_{\max} = \max\left|y_k - f(x_k)\right|$ $(\forall k \in [1, N])$ are then given in the fourth and fifth columns. The complexity of the resulting LFR obtained using the LFR Toolbox (*Magni*) is finally measured via the size of its Δ matrix.

Table 1. Comparison of the BS for $\varepsilon \approx 2.10^{-3}$

Method	Degree	Monomials	RMSE	Max error	LFR size
BP1	6	28	$1.94\ 10^{-3}$	$8.94\ 10^{-3}$	12
BP2	6	21	$2.17\ 10^{-3}$	$1.01\ 10^{-2}$	9
BR1	3	20	$1.90\ 10^{-3}$	$5.71\ 10^{-3}$	8
BR2	3	20	$1.90\ 10^{-3}$	$5.48\ 10^{-3}$	8

Table 2. Comparison of the BS for $\varepsilon \approx 9.10^{-4}$

Method	Degree	Monomials	RMSE	Max error	LFR size
BP1	12	91	$9.20\ 10^{-4}$	$4.84\ 10^{-3}$	24
BP2	12	44	$9.95\ 10^{-4}$	$5.68\ 10^{-3}$	20
BR1	6	56	$9.15\ 10^{-4}$	$3.51\ 10^{-3}$	17
BR2	6	56	$9.28\ 10^{-4}$	$3.81\ 10^{-3}$	17

As expected, rational approximation is more efficient than polynomial approximation, since both the number of monomials and the size of the resulting LFR are lower. Moreover, numerical problems are often encountered with **BP2,** and no solution can usually be found for polynomials of the 15th degree or higher. **BR1** and **BR2** thus appear to be the

most efficient BS and provide very similar results.

4.2 Evaluation of the new algorithm (IS)

The objective is now to compare **BR1** with the indirect solution **IS** proposed in §3. All results are summarized in Table 3, and some are also displayed in Fig. 3-4. For a given degree, **IS** and **BR1** give quite similar results regarding the number of monomials and the approximation accuracy. Indeed, both methods generate rational functions for which the numerator and the denominator are composed of almost all admissible monomials when written in expanded form. But **IS** offers three major pros. First, the size of the resulting LFR is significantly smaller, since the symbolic expression does not appear as a single expanded rational function, but is already factorized as a sum of elementary components described by (8). Besides, **IS** is numerically much more efficient and allows to compute higher degree approximations very quickly and easily. This is not possible with **BR1**, since numerical troubles appear for degrees larger than 8, thus leading to poor results. Finally, it is worth emphasizing that the non singularity of the rational function is guaranteed with **IS** since the obtained solutions are always strictly positive.

For a given degree, the direct solution (**DS**) described in *Hardier 2013* and **BR1** give quite similar results in terms of accuracy. However, **DS** has a great advantage, since it creates rational approximants with sparse structures. 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 too.

IS and **DS** thus appear to be much more efficient than the BS. Besides, these two methods prove quite complementary. **IS** provides very accurate approximations which do not have a sparse structure, but can be directly factorized in a compact form resulting in low order LFR. On the other hand, **DS** directly selects the most relevant monomials to generate very sparse symbolic expressions. It also appears that **DS** is more accurate for low degree approximations, while **IS** gives better results for degrees larger than 8. Indeed, at lowest degrees, the number of radial units used by **IS** (half the required degree) is not sufficient to represent accurately enough the shape of the reference data. Hence, a minimum number of RBF is required to get the most out of this method. Finally, it is worth noting that the computational cost is strongly in favor of the **IS** algorithm. As the degree of the rational function increases, the Darwinian mechanisms of evolution involved by the **DS** require more generations to produce very accurate solutions, and the CPU time is seriously impaired.

Table 3. Comparison of the different algorithms

Method	Degree	Monomials	RMSE	Max error	LFR size
Baseline solution (**BR1**)	4	30	$1.65\ 10^{-3}$	$4.76\ 10^{-3}$	11
	6	56	$9.15\ 10^{-4}$	$3.51\ 10^{-3}$	17
	8	90	$8.10\ 10^{-4}$	$4.67\ 10^{-3}$	23
Indirect approach (**IS**)	6	46	$1.28\ 10^{-3}$	$6.23\ 10^{-3}$	12
	8	77	$7.92\ 10^{-4}$	$5.12\ 10^{-3}$	16
	14	218	$3.55\ 10^{-4}$	$1.96\ 10^{-3}$	28
	26	716	$1.58\ 10^{-4}$	$1.03\ 10^{-3}$	52
Direct approach (**DS**)	6	30	$9.74\ 10^{-4}$	$4.60\ 10^{-3}$	14
	8	25	$8.73\ 10^{-4}$	$4.71\ 10^{-3}$	15
	14	43	$6.53\ 10^{-4}$	$3.05\ 10^{-3}$	28

5. CONCLUSIONS

This paper considers the sparse rational approximation of tabulated or irrational data, in order to compute accurate and yet simple LFR, convenient for system analysis and control design. A novel method based on evolutionary techniques is proposed to solve the underlying optimization problem. It constructs ad hoc modeling by using RBF-type NN, and then converts the result into a sum of elementary rational functions. Together with a constructive OLS procedure, the resulting KOALA tool relies on PSO to select the best local kernels, hence contributing to the parsimony of the result.

The proposed method compares very favorably to classical ones when applied to a realistic aeronautical example. For a given precision, the symbolic expressions are more compact, and hence the size of the resulting LFR is significantly smaller. Good numerical properties are also observed and as accurate as desired results can be quickly obtained provided the degree used for approximation is high enough.

Regarding prospects, the method could be improved by using an adaptive Design of Experiments. Instead of optimizing on a fine fixed-size grid, a coarse sampling could be computed initially and then refined by adding new samples in subdomains whenever the generalization errors prove unsatisfactory. This strategy would result in a reduction of the computational load, but could also be beneficial to the modeling process within strongly nonlinear regions.

REFERENCES

Blackwell T.M. & Bentley P.J. (2002). Dynamic search with charged swarms. *GECCO*, pp. 19-26, New York, USA.

Boiffier J.L. (1998). *The dynamics of flight : the equations*. John Wiley & sons, Chichester.

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

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

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

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

Clerc M. (2006). *Particle swarm optimization*. ISTE, London.

Clerc M. (2008). Initialisations for particle swarm optimization. http://clerc.maurice.free.fr/pso/.

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

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

Hardier G. (1998). 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.

Hardier G., Roos C., Seren C. (2013). Creating sparse rational approximations for LFR modeling using genetic programming. Submitted to the *3rd IFAC Intal Conf on Intelligent Control and Automation Science*, Chengdu, China.

Haykin S. (1994). *Neural Networks: A Comprehensive Foundation*. IEEE Press, MacMillan, New York.

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

Magni J.F. (2006). *User manual of the LFR Toolbox (V 2.0)*. http://www.onera.fr/staff-en/jean-marc-biannic.

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

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

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

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

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

Petrushev P.P. and Popov V.A. (1987). Rational approximation of real functions. *Encyclopedia of Mathematics and its Applications*, Vol. 28, University Press, Cambridge.

Roos C. (2011). 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.

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

Trojanowski K. (2008). Multi-swarm that learns. *Intelligent Information Systems*, Vol. XVI, pp. 121-130.

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

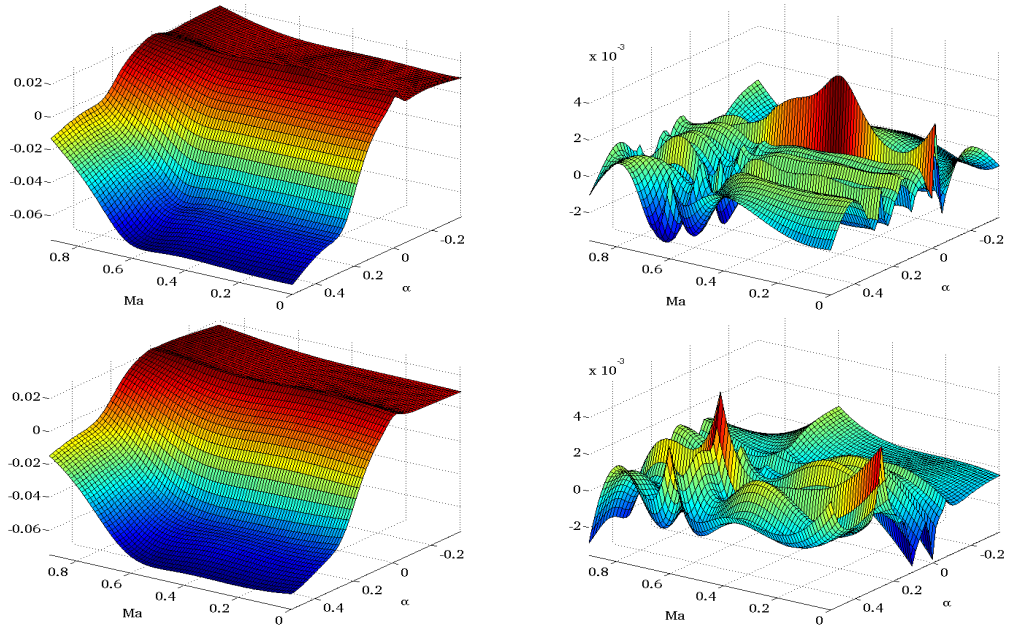

Fig. 3. Approximants of the 8th degree and local approximation errors (top = BS, bottom = IS)

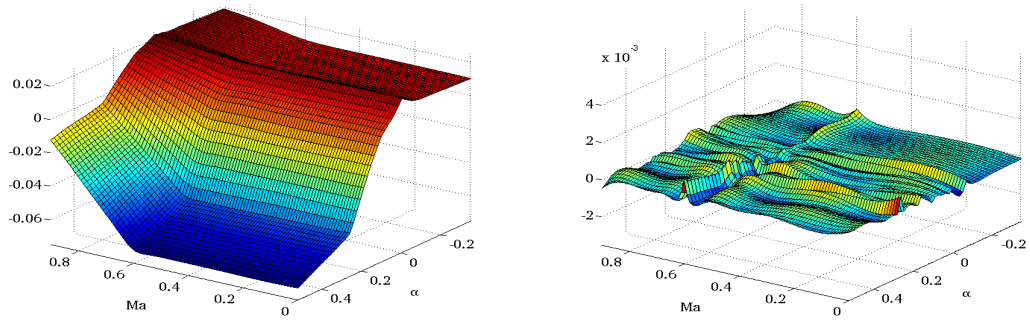

Fig. 4. Approximant of the 26th degree and local approximation error (IS)