A LOCAL LEARNING APPROACH TO REAL-TIME PARAMETER ESTIMATION
Application to an Aircraft

Lilian Ronceray, Matthieu Jeanneau
Stability and Control Department, Airbus France, Toulouse, France
Lilian.L.Ronceray@airbus.com

Daniel Alazard
Supaéro, Toulouse, France

Philippe Mouyon
Département Commande des Systèmes et Dynamique du Vol, ONERA, Toulouse, France

Sihem Tebbani
Département Automatique, École Supérieure d'Électricité, Gif-sur-Yvette, France

Keywords: Local learning, radial-basis neural networks, real-time parameter estimation.

Abstract: This paper proposes an approach based upon local learning techniques and real-time parameter estimation, to tune an aircraft sideslip estimator using radial-basis neural networks, during a flight test. After a presentation of the context, we recall the local model approach to radial-basis networks. The application to the estimation of the sideslip angle of an aircraft, is then described and the various results and analyses are detailled at the end before suggesting some improvement directions.

1 INTRODUCTION

In the aeronautical field, although most of the useful parameters (like inertial data, airspeed) are calculated directly using probes, it is often relevant to use estimators to consolidate the information, thus increasing the redundancy of the aircraft systems or to replace the probes in order to save weight. It then becomes critical to have these estimators tuned in the early days of the flight tests of a new aircraft, when it is only known by an inaccurate numerical model.

The problem that is dealt with in this paper, is that of retuning a specific estimator in real-time (or near real-time) using flight test data. We must then take into account that during flight tests, the aircraft flies around in small regions of the flight domain, yielding a strong locality constraint on the retuning.

The general idea will be to make a combined use of local learning and real-time parameter estimation techniques to tune the estimator, in a small neighbouring of its input space, without degrading its performance in other regions.

2 ABOUT NEURAL NETWORKS

In this section, some generalities about RBF networks are recalled.

2.1 General Description

Such a network is composed by a set of N local estimators $\{\widehat{f}_i(\mathbf{x})\}_{i=1}^N$, defined in the neighbouring of some points $\{\mathbf{c}_i\}_{i=1}^N$ in an input space $\mathcal{I}$ (Murray-Smith, 1994).

The resulting global estimation $\widehat{\xi}$ is the weighted sum of the outputs of all local estimators, for a query point $\mathbf{x} \in \mathcal{I}$ (see equation 1).

The weighting φ_i of the local estimation $\widehat{f}_i(\mathbf{x})$ is a function of $\frac{\|\mathbf{x}-\mathbf{c}_i\|}{\sigma_i}$ for all $i = 1 \dots N$. φ_i is then considered as a radial-basis function. The resulting output is then:

$$\widehat{\xi} = \sum_{i=1}^{N} \varphi_i \widehat{f}_i(\mathbf{x}) \tag{1}$$

2.2 Local Parameters Computation

The local models $\widehat{f}_i(\mathbf{x})$ can either be constants in the neighbouring of the centers, or affine models in the inputs, or any nonlinear model.

From now on, we will show how the parameters of the local models are computed in the particular case where these models are linear in the inputs (Haykin, 1999):

$$\widehat{f}_i(\mathbf{x}) = \mathbf{x}^T \theta_i \quad \forall \mathbf{x} \in I$$

The problem to solve is set as follows: given a set of K different points $\{\mathbf{x}_k \in \mathbb{R}^{m_0}\}_{k=1}^K$ and a corresponding set of K reference values to estimate $\{\xi_k \in \mathbb{R}\}_{k=1}^K$, find a function $\widehat{\xi}$ such that :

$$\widehat{\xi}(\mathbf{x}_k) = \xi_k, \quad \forall k = 1 \ldots K \tag{2}$$

The RBF technique consists in choosing a function $\widehat{\xi}$ that has the following form :

$$\widehat{\xi}(\mathbf{x}) = \sum_{i=1}^{N} \varphi_i \mathbf{x}^T \widehat{\theta}_i = \sum_{i=1}^{N} \psi_i(\mathbf{x})^T \widehat{\theta}_i \tag{3}$$

$$= \psi(\mathbf{x})^T \widehat{\theta} \tag{4}$$

where N is the chosen number of local estimators, and with $\psi_i = \varphi_i \mathbf{x}$.

With $\Psi = \{\Psi_{ki}\} = \{\psi_i(\mathbf{x}_k)\}$, the interpolation condition (2) can be written as a linear system :

$$\Psi^T \widehat{\theta} = \xi \tag{5}$$

In order to find a solution to equation (5) and as Ψ is not a square matrix, we must verify that $\Psi\Psi^T$ is nonsingular, which can be done using Michelli's theorem (Michelli, 1986). A solution $\widehat{\theta}$ satisfying the interpolation condition (2), can then be found using least square optimization :

$$\widehat{\theta}^* = \left(\Psi\Psi^T\right)^{-1} \Psi\xi \tag{6}$$

2.3 Selection of the Centers

A simple solution is to make a regular gridding on the normalized input space. As minimum and maximum variations of the input parameters are known, the input space can be normalized. The gridding is then made on a unitary hypercube.

The issue is that we face the curse of dimensionality though the dimension of each network's input space is smaller than 5. However, some physical considerations, depending on the considered application, may help reducing the number of neurons by making a "truncated" hypercube.

For instance, in our application, some parameters have a dependency in Mach number and angle of attack α. As the aircraft is not designed to fly at both high Mach and high α, we may remove the corresponding part of the domain.

3 APPLICATION

The application we considered here is the estimation of the sideslip angle β of a civilian aircraft, which is the angle between the aircraft longitudinal axis and the direction of flight (Russell, 1996).

To do so, we have a formula for the estimation of β, based on equation (7) that describes the aircraft lateral force equation where we neglect the longitudinal coupling terms and equation (8) which is a classical decomposition of the lateral force coefficient (Boiffier, 1998):

$$mg \cdot Ny_{cg} = P_d S Cy - F_{eng,y} \tag{7}$$

$$Cy = Cy_\beta \beta + \Delta Cy_\beta^{NL} + \frac{l}{V_{tas}}\left(Cy_r r + Cy_p p\right) + \Delta Cy_{\delta r} + \Delta Cy_{\delta p} \tag{8}$$

where Ny_{cg} denotes the lateral load factor, P_d the dynamic pressure, $F_{eng,y}$ the projection of thrust on the lateral axis, β the sideslip angle, p the rolle rate, r the yaw rate, δp the ailerons deflection, δr the rudder deflection, $Cy_\star$ the Cy gradient w.r.t. $\star$ (β, p or r), $\Delta Cy_\star$ the Cy effect due to $\star$ (δp, δr or β), l the mean aerodynamic chord and V_{tas} the true airspeed velocity.

An approximation of the aircraft sideslip can then be deduced :

$$\widehat{\beta} = -\left[\frac{1}{Cy_\beta}\right]\frac{Mg}{P_d S} Ny_{cg} - \left[\frac{\Delta Cy_{\delta p}}{Cy_\beta}\right] - \left[\frac{\Delta Cy_{\delta r}}{Cy_\beta}\right] - \delta_{HL}\left[\frac{\Delta Cy_\beta^{NL}}{Cy_\beta}\right] - \frac{l}{V_{tas}}\left(\left[\frac{Cy_p}{Cy_\beta}\right] p + \left[\frac{Cy_r}{Cy_\beta}\right] r\right)$$

The key points treated in the sequel are the definition of the architecture and the initialisation of the neural networks from a given set of simulated data, and the in-flight tuning of these networks.

3.1 Rbf Networks Applied to Sideslip Estimation

We will start by noticing that the expression (9) is linear in ratios of aerodynamic coefficients.

In order to ease the reader's effort, the following notations are introduced. Let M denote the number of unknown ratios of aerodynamic coefficients, $\widehat{\xi}^m$ the

m-th unknown ratio with $m = 1 \dots M$ and y^m its attached auxiliary measurement. (9) will then be written as:

$$\widehat{\beta} = \sum_{m=1}^{M} y^m \cdot \widehat{\xi}^m \tag{9}$$

The m-th unknown ratio is modelled by a RBF network with N^m linear local estimators $\widehat{f}_{i,m}(\mathbf{x})$. Its input space I^m is a subset of the flight enveloppe variables.

The $\widehat{\xi}^m$ depends on the following variables : α, Mach number, P_d, δr and δp, the last two being only used respectively for $\Delta Cy_{\delta r}$ and $\Delta Cy_{\delta p}$.

According to equation (3), estimated sideslip can then be rewritten as :

$$\begin{aligned} \widehat{\beta} &= \sum_{m=1}^{M} y^m \psi^{mT} \widehat{\theta}^m = \sum_{m=1}^{M} \zeta^{mT} \widehat{\theta}^m \\ &= \zeta^T \Theta \end{aligned} \tag{10}$$

A linear expression of the estimated sideslip is thus obtained, allowing the recursive least algorithm (RLS) algorithm to be directly applied. For a complete formulation of the RLS algorithm and the related criterion, one may refer to (Labarrère et al., 1993).

3.2 The Process in Details

The process will be divided into three main parts.

Initialization: the optimal network structure must be found : RBF, distance function, feature scaling (transformation on the input space), centers location, and smoothing parameter σ_i (Atkeson et al., 1997).

A direct offline learning: where each network is trained individually on a database of aerodynamic coefficient values, computed by the numerical model. A particular attention will be paid to the norm of the local parameter vector, which is an indicator of the generalization performance of the network.

An indirect online learning: where the learning criterion is no longer the error of the networks outputs but the error between the estimated sideslip and the true sideslip. All the networks are trained at the same time and must achieve both local performance, i.e. on the considered flight point and global performance, i.e. on the whole flight domain.

3.3 Analysis

The structure of the neural networks is a key element in the outcome of the process and requires an analysis of the networks' offline performance, with respect to the various degrees of freedom available for the networks' structure.

For clarity reasons, we will only show the performance of the ($\Delta Cy_{\delta r}$) network in the sequel.

Prior to the analysis, the input space of the network is normalized.The effect of the smoothing parameter σ on the learning performance and the Euclidean norm of the parameter vector, which gives an idea of the network's capacity to generalize, will be studied. Both the Euclidean and Infinity norm will be used as distance functions and inverse multiquadrics as RBF ($f : x \mapsto \frac{1}{\sqrt{x^2+1}}$). We perform an offline learning for each value of σ and compute the relative error on the whole training data :

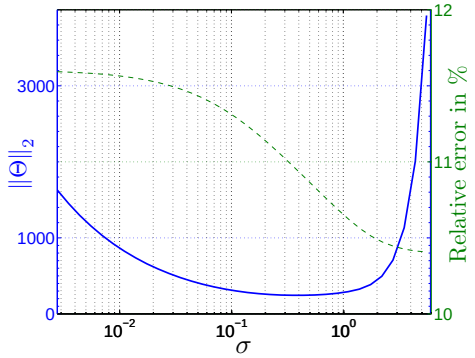

Figure 1: Influence of σ using $\|.\|_2$.

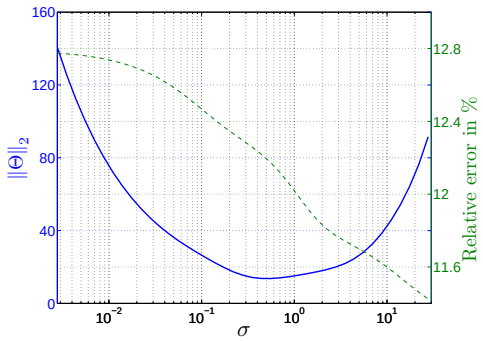

Figure 2: Influence of σ using $\|.\|_\infty$.

The plain line represents the norm of the parameter vector and the dashed line the relative error.

For both norms, we can see a behaviour that can be compared with the overfitting phenomenon with multilayer perceptrons (Haykin, 1999; Dreyfus et al., 2004). We could name this "over-covering", as it seems that local models are strongly interfering with each other, hence over-compensating their interaction. σ must then be chosen that reaches a compro-

mise between a decent performance and a relatively small norm for the parameter vector.

In terms of compared performance, the Infinity norm allows better generalization, as the norm reaches lower values for a quite similar performance. This can be justified by figure 3: the covering of 2D normalized space is presented using gaussian kernels with Euclidean (solid line circles) and Infinity norm (dotted lines squares).

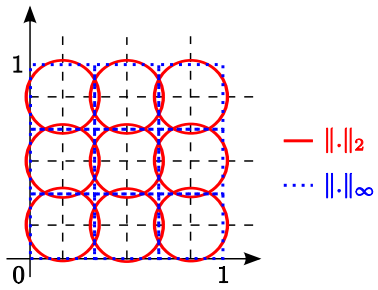

Figure 3: Input space covering using $\|.\|_2$ and $\|.\|_\infty$.

3.4 Implementation

We then chose the structure of our networks : linear local models, inverse multiquadrics as RBF[1], optimal smoothing parameter according to the previous analysis, normalisation of the input space as feature scaling and a regular gridding to locate the centers.

To test our method, we used flight tests recordings to be as close as possible to real conditions.

3.5 Results

The obtained results are presented in this section, from learning on the pre-flight test identification data to the online tuning during simulated flight tests (in-flight recorded data is fed through the estimator).

About the offline part, as it is basic least squares optimization, we will only say that the points were generated randomly using an inaccurate numerical model of the aircraft.

For the online adaptation, we will present results in clean configuration for two distinct flight points (FP1 and FP2) on a steady sideslip maneuver.

First, estimations without and then with the RLS algorithm are presented. The dot-dashed line represents the real sideslip at the center of gravity, the dashed line the estimated sideslip computed by the current method and the solid line the estimated sideslip computed by our estimator (figures 4 and 5).

The aim is to compare the performance of the existing sideslip estimator which uses interpolated approximate values for the aerodynamic coefficients.

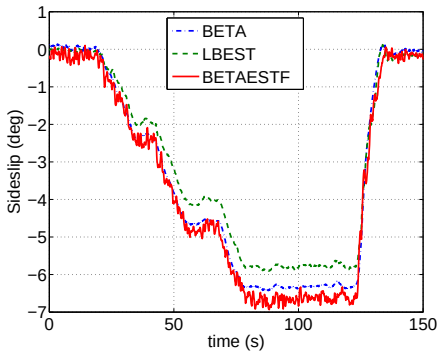

Figure 4: Flight point 1 - Fixed estimator.

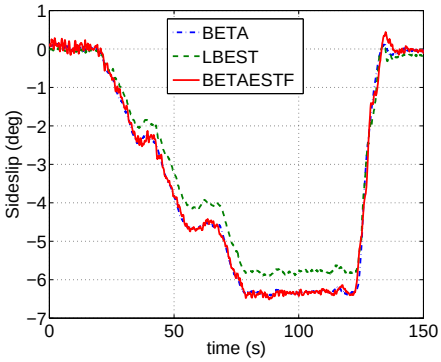

Figure 5: Flight point 1 - RLS estimator.

The results are quite satisfactory because better estimation than the existing estimator is achieved. The noise we can see on both figures comes from the *Ny* sensor and the derivation *p* and *r*.

The issue is the generalization and is emphasized by the following procedure. The estimator is tuned on FP1 then on FP2. Local performance is achieved for both flight points as shown in figure 5 for FP1. The estimator is then verified on FP1. We can see on figure 6 that the original performance has been degraded. The learning algorithm does not tune locally enough and impacts the whole flight domain.

[1]They are better suited for an implementation on an embedded computer

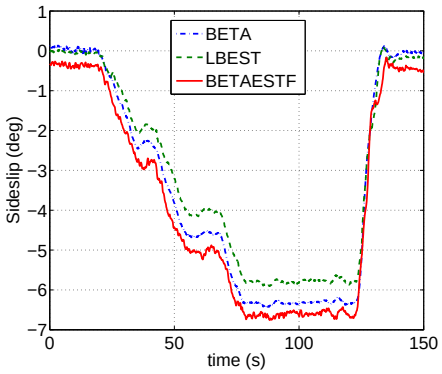

Figure 6: Generalization from FP2 to FP1.

4 CONCLUSION

Throughout this paper, we studied the application of RBF-based neural networks on the estimation of an aircraft sideslip and tried to find a method to tune it in real-time during a simulated flight test.

Though such networks have interesting local properties, some improvements are required on the different steps of the process, mainly on the generalization performance. Work is currently on-going about using total least squares algorithm instead of the classical least squares (Huffel and Vandewalle, 1991; Björck, 1996) and their recursive form (Boley and Sutherland, 1993). Other work directions will be investigated :

- allowing directional forgetting in the RLS algorithm (Kulhavy and Kárny, 1984)
- reducing numerical complexity
- adaptive filtering on the estimator output to soften input noise effects

REFERENCES

Atkeson, C., Moore, A., and Schaal, S. (1997). Locally weighted learning. *AI Review*, 11:11–73.

Björck, A. (1996). *Numerical Methods for Least Squares Problems*. S.I.A.M., first edition.

Boiffier, J.-L. (1998). *The Dynamics of Flight, The Equations*. Wiley.

Boley, D. L. and Sutherland, K. T. (1993). Recursive total least squares: An alternative to the discrete kalman filter. Technical Report TR 93-32, Computer Science Dpt, University of Minnesota.

Dreyfus, G., Martinez, J.-M., Samuelides, M., Gordon, M., Badran, F., Thiria, S., and Hérault, L. (2004). *Réseaux de neurones, méthodologies et applications*. Eyrolles, second edition.

Haykin, S. (1999). *Neural Networks, a comprehensive foundation*. Prentice-Hall, second edition.

Huffel, S. V. and Vandewalle, J. (1991). *The Total Least Squares Problem : Computational Aspects and Analysis*. S.I.A.M., first edition.

Kulhavy, R. and Kárny, M. (1984). Tracking of slowly varying parameters by directional forgetting. *Preprints of the 9th IFAC World Congress*, X:78–83.

Labarrère, M., Krief, J.-P., and Gimonet, B. (1993). *Le Filtrage et ses Applications*. Cépaduès.

Michelli, C. A. (1986). Interpolation of scattered data : Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22.

Murray-Smith, R. (1994). Local model networks and local learning. In *Fuzzy-Duisburg*, Duisburg.

Russell, J. B. (1996). *Performance and Stability of Aircraft*. Arnold.