Exploiting a database to predict the in-flight stability of the F-16

David Amsallem and Julien Cortial

December 12, 2008

1 Introduction

Among the critical phenomena that have to be taken into account when an airplane is designed, flutter is perhaps the hardest to assess. This spurious aeroelastic behavior of the aircraft only appears at certain flying speeds and is highly sensitive with respect to the aerodynamic and structural design of the airplane. While the study of the flutter of an aircraft used to be exclusively done experimentally, aeroelastic computational methods have been developed and successfully applied to full-aircraft configurations in the last 10 years. These computations involve expensive, non-linear, large-scale (millions of degrees of freedom), multiphysics simulations and can only be applied to a few carefully-chosen configurations. In those computations, the output quantity of interest is typically the transient behavior of the lift of the aircraft from which flutter can be assessed. In order to address the cost issue, reduced-order modeling methods such as principal component analysis (PCA) have been successfully applied to the flutter problem. These computational methods are able to capture the main properties of the system such as the linear stability of the aircraft at a much lower cost than the full-order simulations [1].

However, these reduced-order models (ROMs) are not robust with respect to the flight condition and hence have to be rebuilt when any parameter is modified. To alleviate the computational burden, a database approach has been recently proposed: First, a limited number of reference flight conditions are chosen and the corresponding ROMs are built offline. A cheap online interpolation is used to output a ROM adapted to the considered input parameters. The effectiveness and the robustness of the method has been proved when applied to the aeroelastic behavior of two full aircraft configurations (F-16 and F-18/A) [2].

The prediction for a particular input relied so far on an educated guess as well as an ad-hoc trial-and-error process in order to determine the interpolation parameters such as the number of points to use as well as their location. The choice of these parameters is critical as the adapted ROM can lead to accurate or inaccurate time responses as shown in Figure 1. The first interpolated ROM (in red) gives a meaningless approximation of the reference time response shown in blue whereas the second one (in green) is a good approximation. In this work, we will always use the error in L_2-norm between the responses as an accuracy criterion.

The main objective of the project is to bring the framework one step closer to an industrial-strength implementation, which could ideally be treated as a black-box system by the final user instead of requiring an extensive knowledge of the underlying principles. The ultimate goal would be to enable real-time flutter assessment. This implies to be able to exploit the discrete database, used as a training set, to reliably predict the physical behavior at any arbitrary input feature in the domain of interest.

This report is organized as follows: First we present the problem specificities and the anticipated difficulties derived from empirical experience. Then we describe the training methodology that was developed concurrently with a cross-validation process. Finally we show that the resulting framework can be used to make accurate predictions in a realistic test case.

2 Overview

The solution vector for the flow around the aircraft at a given flight condition has $N_{\text{dof}} = 2,019,595$ entries, each corresponding to one degree of freedom (dof) of an accurate numerical model for the considered physical system. However, this full-order model can be reduced using the PCA method by projection onto a subspace of dimension $N_{\text{rom}} = 90$. When the reduced space is properly chosen, experience has shown that this ROM is sufficient to investigate the flutter phenomenon of the aircraft at a specific flight condition that depends on 2 parameters, namely the speed of the aircraft (Mach number) and its attitude (angle of attack). The training

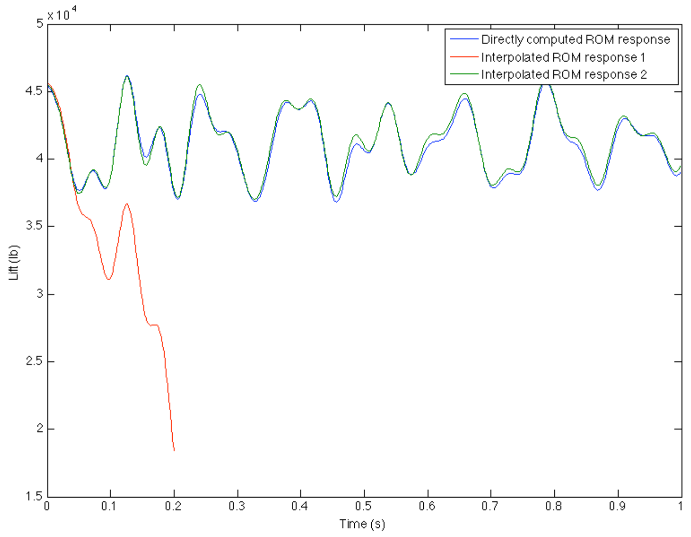


Figure 1: Comparison of the lift time-response computed directly and two time-responses computed using two different interpolated ROMs.

set considered in this project is therefore a database of such ROMs constructed for a F-16 configuration, where each training example corresponds to the mapping of a given flight condition to the corresponding reduced basis stored as a $N_{\mathrm{dof}} \times N_{\mathrm{rom}}$ matrix.

While the considered problem has only 2 scalar input features, it exhibits the following specific properties:

- The output variables may change arbitrarily fast when the input features change, which may lead to inaccurate predictions or even to ill-posed interpolation schemes if the training examples are too far apart;
- Because of the high computational costs, it is desirable 1) to perform as much preprocessing as possible to ensure an efficient online interpolation and 2) to limit the amount of offline computations as well.

Both of these observations lead us to consider only local interpolation schemes. Therefore, we postulate that the exploitation of a database should begin with the definition a series of *clusters* in which we can reasonably expect to derive meaningful local interpolation schemes.

Furthermore, since our experience so far has clearly shown that extrapolation was defective, we want to make sure that a proper interpolation is used on any point of the domain of interest. Thus it appears more relevant to define clusters in terms of subdomains covering the domain of interest in the feature space.

To sum up, our goal is to determine a unique and accurate interpolation scheme for each subdomain of the parameter space.

3 Training phase

3.1 Database construction

Using 32 CPUs on a computational cluster, it takes about 30 minutes to compute each one of the 46 entries in the database, which are reported as blue dots in Figure (2a): The convex hull (shown as a blue line) in the parameter space (Mach number and angle of attack) corresponds to the domain of interest.

3.2 Triangulation and dual cells construction

The main goal is to partition the parameter domain in *cells* such that each cell contains one and only one training point and such that the union of all these cells covers all the convex hull, except a very limited region along its boundaries. This proceeds in two steps:

- Step 1: Triangulate the set of points in the database using Delaunay's triangulation. The resulting triangles are shown in blue in Figure (2b).

- Step 2: Consider the set of the centers of gravity of the triangles from step 1. For each interior point in the database, define its dual cell as the polygon whose vertices are the centers of gravity of the adjacent triangles. The resulting dual cells are reported in red in Figure (2b). Notice that the dual cells cover the interior of the database and that each cell contains one and only one point (shown in green in the figure).

3.3 Clustering

The idea is to define several clusters of training examples via the cells defined previously. These cells are first partitioned into groups using a variant of the k-means algorithm. Then, for each of these groups, a cluster is defined by the set of points inside every cell of the group as well as every neighboring point.

The k-means algorithm is here applied to the set of dual cells in order to partition this set into clusters. To balance accuracy, exhaustivity and computational cost, we have determined that it is optimal to have about 10 points in each database subset, which means that each cluster of dual cells must have between 2 and 4 dual cells. This gives therefore an upper and a lower bounds for the number of cells in any given cluster. The algorithm proceeds as follows.

- Step 1: Specify an a-priori number of clusters k, as well as lower and upper bounds on the number of cells in each cluster.
- Step 2: Randomly determine the initial k cluster centers.
- Step 3: Run the k-means clustering algorithm on the set of cells until convergence. The cell centers are used to compute the distances.
- Step 4: Check that every cluster has a valid number of points. If any cluster has too many points, a new cluster is added, its initial center being randomly chosen in the original cluster. If any cluster does not have enough points, it is simply removed.
- Step 5: If at least one cluster does not have the required size, go back to Step 3. Otherwise terminate the algorithm.

The resulting grouping of the dual cells is reported in Figure (2c). 11 clusters have been created, each one containing between 2 and 4 cells and between 9 and 13 points of the database, after inclusion of the neighboring points.

3.4 In-cluster cross-validation

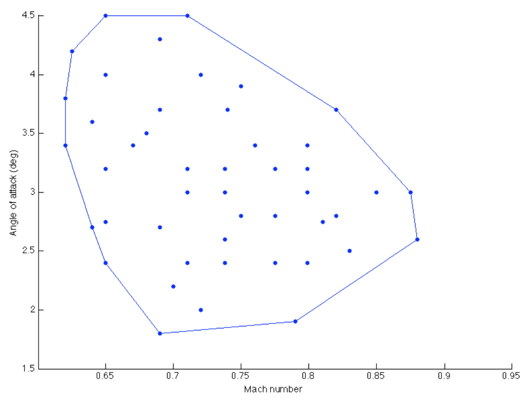
Inside each cluster, a cross-validation process is used to select the best local model inside each cell. For the sake of simplicity and to limit the computational time, two different interpolation models are defined:

- M_1 : Interpolation using $N_1 = 4$ pre-computed points.
- M_2 : Interpolation using $N_2 = 5$ pre-computed points.

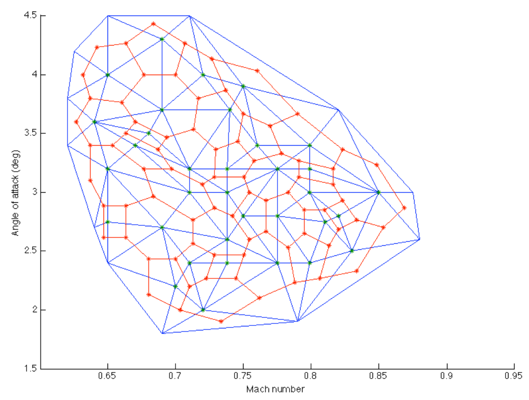
Each cluster C includes interior points $x_i,\ i = 1, ..., n_{iC}$ and boundary points $\bar{x}_i,\ i = 1, ..., n_{bC}$. Let us then define the singletons $S_i = \{x_i\}$, for $i = 1, ..., n_{iC}$ and the set of boundary points $\bar{S} = \{\bar{x}_i,\ i = 1, ..., n_{bC}\}$. Leave-one-out cross-validation is then applied as follows:

- Evaluate each model $M_i,\ i = 1, 2$:
  For $j = 1, ..., n_{iC}$:
  Train the model M_i on $T_j = S_1 \cup \cdots \cup S_{j-1} \cup S_{j+1} \cup \cdots \cup S_{n_{iC}} \cup \bar{S} = C \backslash S_j$, that is perform 10 interpolations using N_i points randomly chosen in T_j. Compare the 10 responses obtained by interpolation to the response obtained directly at the training point and retain the most accurate one in the L_2-norm error. The corresponding interpolation scheme (set of points that were used in the interpolation) becomes the hypothesis h_{ij}.
  Denote by $\hat{\varepsilon}_{s_j}(h_{ij})$ the L_2-norm error between the direct response at S_j and the one obtained by interpolation using the hypothesis h_{ij}.
  Compute the generalized estimation error for the model M_i.
- Pick the model M_i with the lowest estimated generalization error and for each cell S_j assign the corresponding interpolation method as the interpolation method to be used whenever interpolation is required inside the cell.

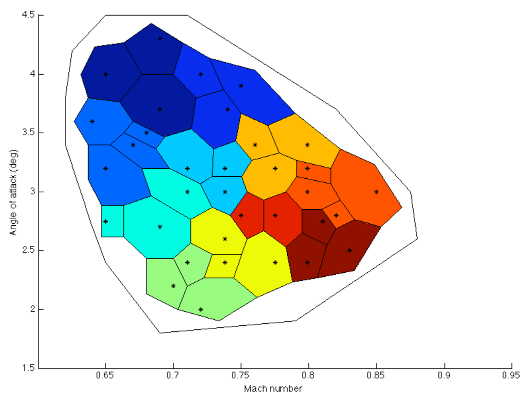
The interpolation parameters determined using this algorithm are shown in Figure (2d) and their corresponding retained models in Figure (2e). In Figure (2d), for each interior point represented by a dot in the figure, the corresponding set of interpolation point is shown as a polygon of the same color enclosing it. In Figure (2e), the cells where the model M_1 has been retained are reported in blue and the one for model M_2 are in red.
In conclusion, this training step has assigned to each cell a unique interpolation scheme to be used whenever an arbitrary ROM whose input parameters belong to the cell is desired.



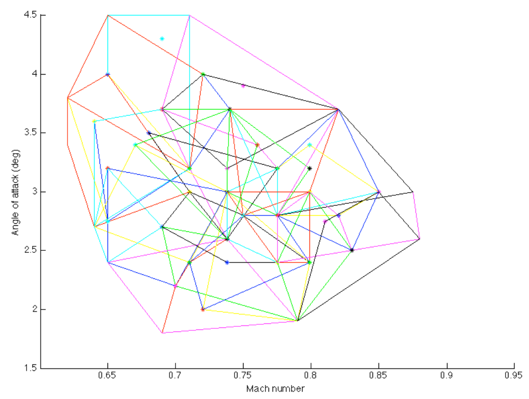
(a) Training database and domain of interest



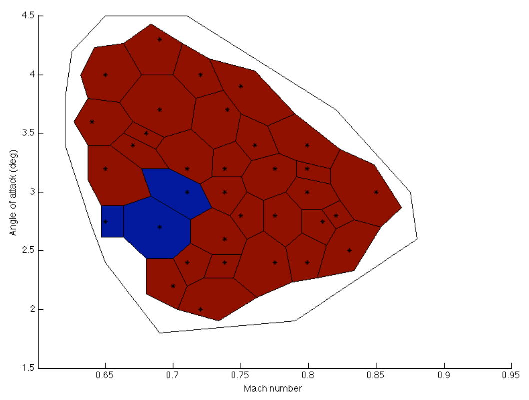
(b) Triangulation and corresponding dual cells



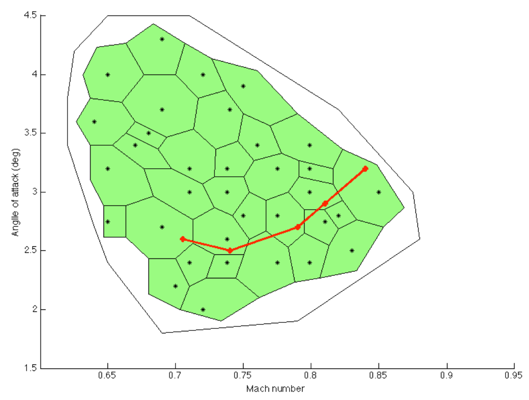
(c) A possible set of groups of cells



(d) Retained local interpolations schemes



(e) Local interpolation models



(f) Test points for the prediction phase

Figure 2: Training and prediction phases for the F-16 model

4 Prediction

We take 5 points in the input feature domain representing a “realistic” flight test and compare the predicted ROM given by our methodology to the (supposedly) accurate ROM obtained by a direct method. These 5 points are represented as the 5 red dots in the parameter space in Figure (2f).

Test point #	1	2	3	4	5
Relative error (%)	1.49	1.26	0.55	1.52	0.81

Table 1: Accuracy of the interpolated ROM versus the ROM obtained by a direct method

The relative error of the time-dependent stability analysis is given in Table 1. It can be shown that the results obtained by interpolation are very accurate. The corresponding time-histories of the lift at the test point 2 using the direct and the interpolation methods are shown in Figure (3). Very good agreement can be seen as the error is of the same magnitude as the intrinsic error of a ROM versus a full-order model.

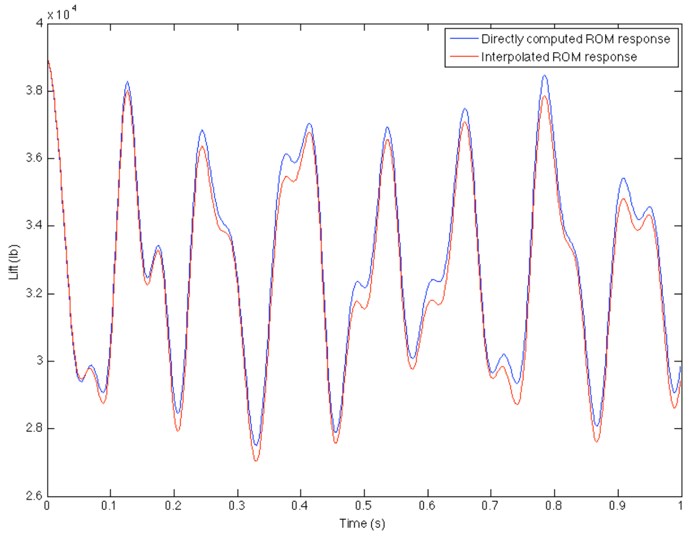


Figure 3: Lift time-histories comparison for test point 2.

5 Conclusion

The presented method enables a full-scale and accurate database exploitation for real aircraft stability investigation, which is of great importance for in-flight flutter prevention. An interesting and important extension of this work would be the definition of an improved database generation process. For instance, when the produced prediction is not considered adequate, the database should be automatically locally refined. The feasibility of a completely automated data gathering process could also be investigated.

References

[1] Lieu, T., Farhat, C. and Lesoinne, M., “Reduced-order fluid/structure modeling of a complete aircraft configuration”, Comput. Methods Appl. Mech Engrg., Vol. 195, 2006, pp. 5730-5742.

[2] Amsallem, D. and Farhat, C., “Interpolation method for adapting reduced-order models and application to aeroelasticity”, AAIA Journal, Vol. 46, No. 7, 2008, pp. 1803-1813.