International Forum on Aeroelasticity and Structural Dynamics
IFASD 2019
9-13 June 2019, Savannah, Georgia, USA

A NEW FLUTTER PREDICTION ALGORITHM TO AVOID P-K METHOD SHORTCOMINGS

Ludovic Colo[1], Gabriel Broux[2], and Eric Garrigues[3]

Dassault Aviation
78 quai Marcel Dassault, 92552 Saint-Cloud France
[1] ludovic.colo@dassault-aviation.com
[2] gabriel.broux@dassault-aviation.com
[3] eric.garrigues@dassault-aviation.com

Keywords: flutter, invariant pairs theory, block Newton, NLEVP.

Abstract: Flutter is a dynamic phenomenon occurring on aircraft structures for particular flight points when structural and aerodynamic modes couple in an unstable way. This phenomenon is computed during design phases to predict the aeroelastic behaviour of the aircraft and prevent any instability in the flight domain. To solve this problem, aircraft manufacturers currently use algorithms pertaining to a family of methods labelled as the "p-k method". These algorithms have been optimized for the flutter problem and provide quite accurate results but in the same time exhibit some flaws which can be prejudicial when one wants to fully automate the flutter analysis. New techniques to solve the non-linear eigenvalue problems such as the flutter problem have been developed recently. Based on the invariant pairs theory, they lead to more robust solutions in terms of unicity. A bloc Newton algorithm applying these techniques has been implemented, validated on standard flutter cases (both civilian and military aircraft) and tested on cases displaying convergence and unicity issues with the p-k method. The outputs present the expected behaviour despite a longer computational time. In order to keep computational times within industrial timeframe, a hybrid method, switching from p-k to block Newton "when required" has been developed. Thanks to this hybrid method implemented in Dassault Aviation in-house solver Elfini, standard flutter computations at Dassault Aviation benefit from the advantages of the invariant pairs theory.

1 INTRODUCTION

Flutter predictions are part of the design and certification process of a new aircraft. They need to be as accurate as possible in order to ensure structural safety and to limit the structural mass added to meet the certification criterion. Even though flutter has been studied since the early developments of aviation [1], it has become an important part of civilian aircraft design since the 1950s. This is due to the use of jet engines which led to higher aircraft speeds, the trend to design lighter hence more flexible aircraft and the development of computers, allowing a more efficient way to numerically solve the flutter problem. Several computational methods exist to predict the aeroelastic behaviour and the flutter speed. The most widely used is known as the p-k method. It has been developed in the 1960s specifically for the aircraft industry and is nowadays used by almost all aircraft manufacturers.

The p-k method is known to be pretty accurate in terms of flutter predictions (it has been validated against numerous numerical data on many occasions) and has been optimised in order to produce results as fast as possible. It is now extensively used at Dassault Aviation

from pre-design to certification phases for the development of civilian and military aircraft [2]. In the past decades the number of flutter computations performed during the various phases of an aircraft development has been significantly increasing. This is due to the fact that civilian aircraft are getting bigger and more flexible which lead to a higher modal density at low frequencies. Besides for military aircraft, the diversity of under-wing external store configurations (cf. Figure 1), coupled to the number of fuel cases to consider give rise to a large number of combinations to study. The trend nowadays is to automate as much as possible flutter computations and their analyses to limit their cost.

Figure 1: Diversity of the under-wing store configurations on a Rafale aircraft.

However, the p-k method presents some flaws: a certain lack of accuracy in the predicted behaviour for modes with strong damping and/or low frequencies, merging of modes due to the inappropriate treatment of close eigenvalues, etc. The latter is the source of discrepancies far from the flutter point while the former may lead to the loss of flutter mechanisms during the more and more automated analyses, hence the need for “manual” intervention to treat the cases showing such behaviour. These interventions ensure that at the end of the analyses all the flutter mechanisms are accounted for but they reduce the benefit of the automation of the flutter computation post-treatment. This is one of the reasons why a computational method as robust as possible is needed.

After deriving the flutter equation (Section 2), describing the p-k method (Section 3) and detailing the reasons for the merging of modes (Section 4), this article presents a new approach based on block computations of invariant pairs to solve the flutter problem without the rise of merging modes (Section 5). The theory of invariant pairs has been established and developed at the end of the 2000s by mathematicians and has been used so far to solve several non-linear eigenvalue problems (crack propagation, fluid-solid vibrations or damped structural vibrations [3], PDE resolution, radio-frequency gun cavity [4]), but to the best of the authors knowledge, the work presented in this article constitutes the first application of this theory to an industrial case such as the flutter problem.

This new method has been implemented in ELFINI, the in-house aero-structural solver platform at Dassault Aviation, after that some work to fit the flutter problem into the invariant pairs formalism was performed (Section 6). The algorithms have been validated against standard cases and also tested on cases displaying the default of "merging of modes" when solved by the p-k method. It is now currently used in aircrafts projects.

2 THE NONLINEAR EIGENVALUE FLUTTER PROBLEM

A flutter analysis is a dynamic stability analysis of a system taking into account the coupling between the structural forces (elastic and inertial) and the aerodynamic forces applied on this structure due to its deformation. The criterion to assess whether or not a system is stable is based on the damping of the modes computed during the flutter analysis. The modal parameters solution of a flutter problem (frequency, mass, shape and damping) vary with the modifications of the Mach number, dynamic pressure and speed due to a variation of flight point. By solving the flutter problem, one wants to get the evolution of the modal parameters along variations of the aerodynamic forces in order to detect the flight points for which the system becomes unstable.

Starting from the Equation of Motion in the Laplace-domain applied to an aircraft model expressed on a modal basis, one can write:

$$[p^2M + pC + K]x = F_{aero}(p, Mach, V, x) \tag{1}$$

With p the complex Laplace variable, M, C and K the real mass, damping and stiffness matrices, x the generalised coordinates of the modal basis and F_{aero} the complex aerodynamic forces (depending on the Mach number $Mach$ and the aircraft speed V) due to the deformation of the structure. These forces can be expressed under a linearity assumption as:

$$F_{aero}(p, Mach, V, x) = \frac{\partial F_{aero}}{\partial x} x = \bar{q} GAF(p, Mach, V)x \tag{2}$$

Where $\bar{q}$ is the dynamic pressure and GAF is the matric of the Generalised Aerodynamic Forces for given p and Mach number. The combination of (1) and (2), yields the flutter equation, where p and x are the unknown to be computed:

$$[p^2M + pC + K - \bar{q}GAF(p, Mach, V)]x = 0 \tag{3}$$

The dependency of the GAF with the Laplace variable is expressed throughout the reduced frequency $k = \frac{\omega c}{V}$ (with ω being the pulsation, c a characteristic length, and V the aircraft

speed). This comes from the relation between p, ω and the damping ξ and the assumption of low damping for the aircraft structure near flutter point:

$$p = \omega(j - \xi) \tag{4}$$

From (4) it comes that $k = \frac{Im(p)c}{V}$. Finally, for given aerodynamic conditions, the flutter equation can be written as:

$$T(p)x = 0 \tag{5}$$

Equation (5) has the canonical form of an eigenvalue problem as soon as the constraint $x \neq 0$ is added. Furthermore, the dependency of the matrix operator T with the Lapalce variable p is non-linear allowing one to characterise the flutter problem as a NonLinear EigenValue Problem (NLEVP). This problem is nonlinear with respect to p but it is linear with respect to x which is of paramount importance for the methods presented in this article.

Without further assumptions on the GAF formulation, one can only state that they depend nonlinearly of p, hence the nonlinear characterisation. With a quasi-steady assumption or more generally a state-space expression for the GAF [5], the flutter problem would be a polynomial eigenvalue problem for which numerous solving methods already exist and for which the eigenvectors solution of the problem form an orthogonal basis. This is not the case for NLEVP, nothing ensures that two eigenvectors are orthogonal (see §1.1 of [6]). This implies for instance that all the solving techniques based on deflation are not relevant to solve the flutter problem.

A NLEVP is solved iteratively by either successive methods (determination of the different eigenvalues-eigenvectors pairs one by one) or simultaneous ones (determination of all the eigenpairs solution at once). The iterative character of these methods means that at the end, only an approximation of the solution is obtained, by opposition to direct methods providing the exact solution. In order to get the evolution of the modal parameters with the aerodynamic loads, several flutter problems are solved at different flight conditions. For close flight conditions, the problems to solve are quite similar, therefore it is common to use the solution of one problem to initialise another one close enough. This set of problems is known as parametric nonlinear eigenvalue problem.

3 THE P-K METHOD

There are many flavours of the p-k method to solve the flutter problem. In this article, the one presented uses the inverse iteration method on a Taylor expansion. Assuming an approximation p_0 of the eigenvalue solution p is known, the idea is to determine Δp in order that $p = p_0 + \Delta p$. If p is an eigenvalue of the flutter problem (5), then there is a right eigenvector x verifying (5) and a left eigenvector y verifying $y^T T(p) = 0$. Left-multiplying (5) by y^T yields:

$$y^T T(p)x = 0 \tag{6}$$

This formulation is known at Dassault Aviation as the "Brévan formulation" of the p-k method [7] and its advantage lies in the fact that one is working with scalar values instead of vectors.

By applying a first order Taylor expansion to (6), and by making the assumptions that the variations in x and y are negligible compared to the ones in p and that a method to compute x and y for a given eigenvalue is at hand, one can write:

$$y^T T(p_0 + \Delta p)x = y^T \left[T(p_0) + \frac{\partial T}{\partial p}(p_0)\Delta p \right] x = 0 \tag{7}$$

Hence:

$$\Delta p = -\left[y^T \frac{\partial T}{\partial p}(p_0)x \right]^{-1} y^T T(p_0)x \tag{8}$$

By denoting the residual $y^T T(p_0)x = R$, it is direct to see that if $p = p_0$, then $R = 0$. The convergence criterion of this method can be placed on the residual as well as on the increment in the eigenvalue Δp.

This allows to get the increment needed for the estimated eigenvalue to get closer to the solution. The remaining steps are to compute the eigenvectors associated to a given eigenvalue, this can be done thanks to the inverse iteration method described in [8]. The algorithm for a right eigenvector can be summed up as:

Algorithm 1: Inverse Iteration Method

1. Pick an initial vector $v^{(0)}$ such that $\left\| v^{(0)} \right\| = 1$

For $k = 1,2, \ldots$

2. Determine w such that $T(p)w = v^{(k-1)}$
3. Define $v^{(k)} = w/\|w\|$

Stop when $\left\| v^{(k)} - v^{(k-1)} \right\| < \varepsilon$

The same kind of procedure is used to determine the left eigenvector. With Algorithm 1 in mind, it is possible to describe the algorithm allowing to compute the solution for given flight conditions and a given initial value of p_0.

Algorithm 2: Brévan's p-k method (for one eigenvalue)

1. Pick an estimate of the eigenvalue solution p_0

While convergence criterion is not met

2. Compute the eigenvectors associated to p_k [Algorithm 1]
3. Set the residual to $R = (y_k)^T T(p_k)x_k$
4. Compute the increment Δp according to (8)
5. Update $p_{k+1} = p_k + \Delta p$
6. Convergence test

A close estimate of the solution at the first step for Algorithm 1 and Algorithm 2 reduces the number of iteration required to reach convergence. The solution found at the end of Algorithm 2 is only dependant on the initial guess p_0, the matrices $T(p_k)$ and their derivatives. According to the properties of the inverse iteration method, Algorithm 2 converges toward the eigenvalue p_j such that $\left\| p_j - p_0 \right\| \leq \|p_i - p_0\|$ for any eigenvalue p_i of T.

Finally, the main algorithm to solve the entire parametric flutter problem and to get the modal behaviour along the various flight conditions considered is given below:

Algorithm 3: p-k method for the entire flutter problem

1. Compute the modal characteristics "on ground" (with $\bar{q} = 0$)

For each mode of interest

For each flight point considered

2. Estimate $\Delta\omega$ and $\Delta\xi$ from previous flight point
3. Determine the pair (p, x) solution at the flight point [Algorithm 2]
4. Extract modal frequency and damping from p
5. Detect flutter for this mode at this flight point

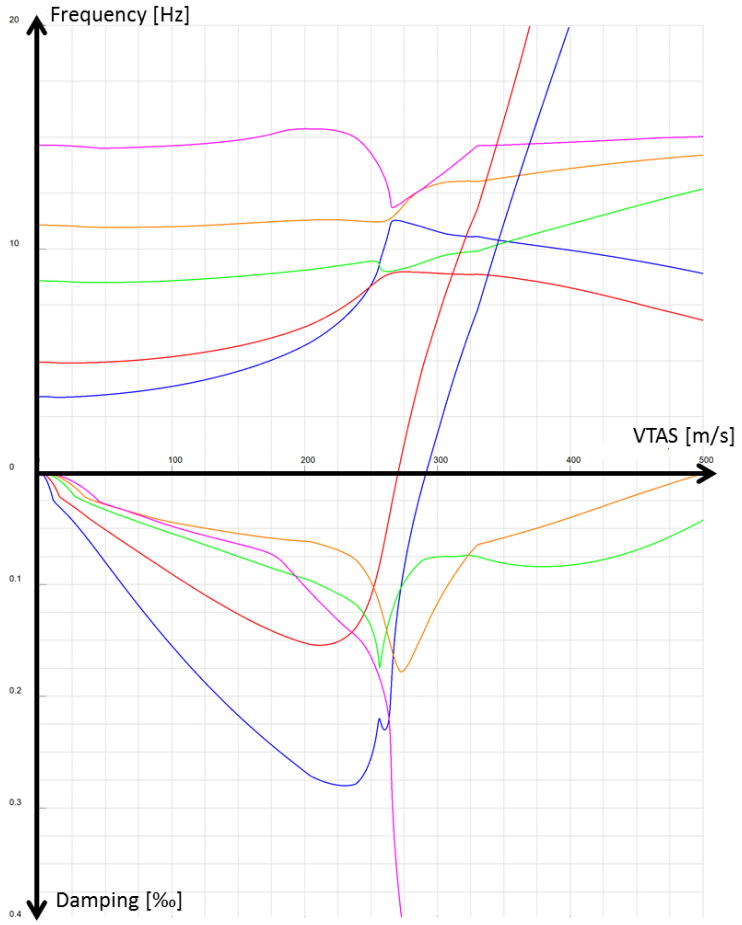

Figure 2: Flutter diagram for 5 modes.

The p-k method has been developed specifically to solve the flutter problem. It has been optimised for industrial purposes both in accuracy and computational time. This is why it is widely used among aircraft manufacturers. One way to visualise the output obtained by Algorithm 3 is to plot the frequency and the damping of the modes on a graph function of one aerodynamic parameter (such as dynamic pressure, true airspeed, etc.). An example of such visualisation is provided Figure 2 for a "simple" case with 5 modes. For industrial cases of civilian aircraft, hundreds of modes are usually computed. An example of such computation is given Figure 3.

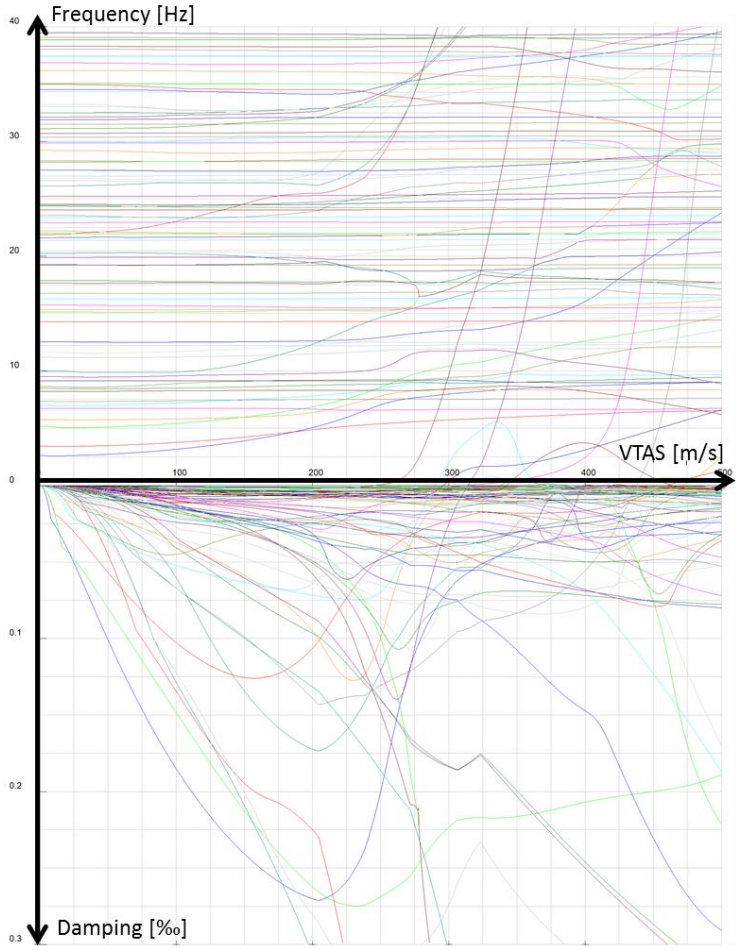

Figure 3: Typical flutter diagram for a civilian aircraft.

4 MERGING OF MODES IN THE P-K METHOD: ORIGIN, CONSEQUENCES AND DETECTION

Due to the properties of the p-k method, it can happen that for given aerodynamic conditions, the algorithm finds twice the same eigenvalue: this is called merging of modes. For the next flight points considered after a merging of modes, the p-k method finds exactly the same solution for both merged modes, hence the loss of one mode after a certain point in the output of the computations. This might lead to some problems during the analysis of the results:

- the lost mode might become unstable and the analysis is short of one flutter mechanism,
- the lost mode merges onto a mode which becomes unstable at some point after the merge, the results then show two unstable mechanisms instead of one,
- when comparing computational results to experimental data, the behaviour of one mode is missing.

For all these reasons, it seems important to understand why modes can merge with the p-k method, how one can automatically detect such merge and if there are other options to make sure these merges do not happen.

An example of a merging of modes is given Figure 4. After approximately 250 m/s, the red and the blue modes merge and one modal behaviour is missing on the diagram.

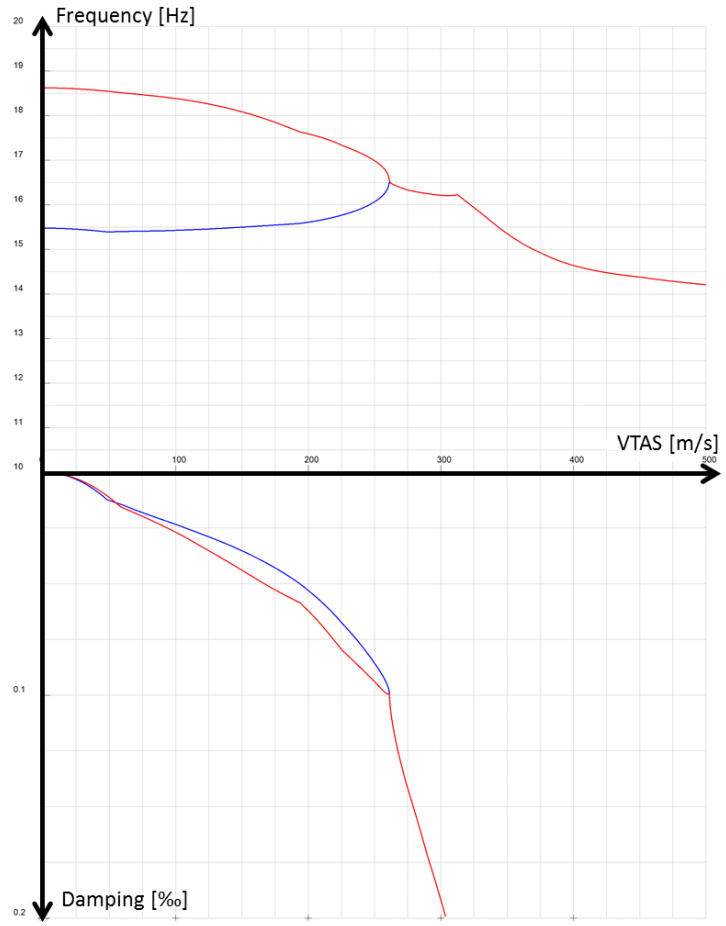

Figure 4: Flutter diagram of two merging modes.

The origin of the merging of modes is linked to the fact that the p-k method is a successive method acting on each modal solution independently of the others. This means that no account of the already found modal solutions is taken while looking for a new modal solution. This flaw can be found in several successive methods. Figure 5 helps understand what happens in the p-k method. As mentioned in §3, p-k method finds the eigenvalue closest to the initial estimate provided. From the situation showed on the left of Figure 5, for both modes, p-k method finds the same eigenvalue (middle of Figure 5), whereas the expected outcome is shown on the right. Due to its successive resolution, nothing in the p-k method ensures that an eigenvalue is found only once for a given flutter problem.

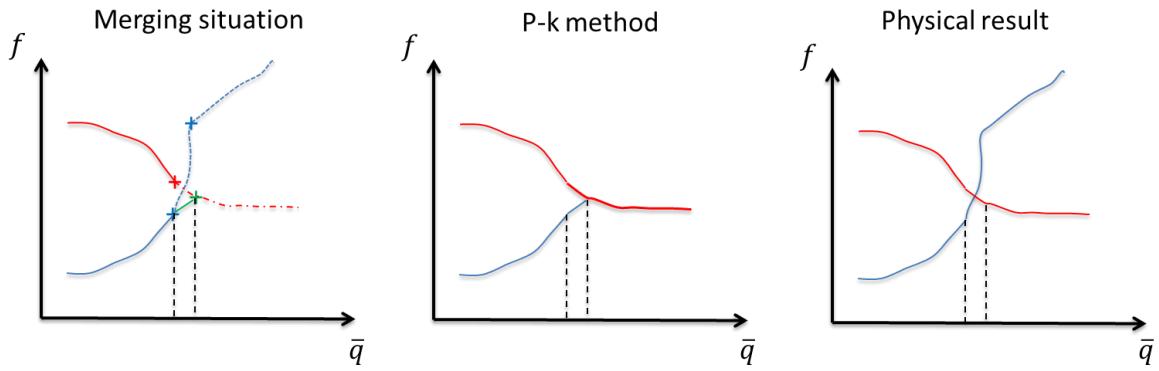

Figure 5: Schemes illustrating the merging of modes in the p-k method.

From a mathematical point of view, this situation can be characterised by looking at the algebraic multiplicity of the eigenvalues, denoted in this article by $alg_T(p)$. By definition, this is the smallest integer j such that $\frac{d^j}{dx^j}\det\big(T(p)\big) \neq 0$. The eigenvalues solutions of the flutter problem are in the vast majority of cases simple solution of the problem, meaning their multiplicity is 1. In the case of merged modes, the multiplicity then equals at least 2 for the eigenvalue which is incorrect. At most, a multiplicity of 2 can occur when two modes cross each other in frequency and damping simultaneously as shown Figure 6. This is one point on the flutter diagram and the probability for such an occurrence to match the discretisation of the aerodynamic parameters is very small.

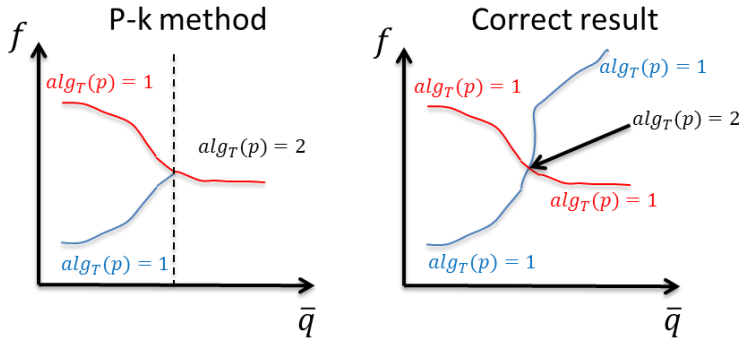

Figure 6: Algebraic multiplicity of the eigenvalues in the merging of modes (left) and the correct result (right).

In order to automatically detect the merging of modes, the Modal Assurance Criterion (or MAC) can be used to track collinearity between the eigenvectors. The p-k method computes the eigenvectors solution based only on the eigenvalues (thanks to the inverse iteration method), hence if two modes merges, then they have the same eigenvectors. The MAC is defined by the following formula:

$$MAC = \frac{\|(x,y)\|}{\sqrt{(x,x)(y,y)}} = \|\cos(\widehat{x,y})\| \tag{9}$$

Where (x,y) denotes the hermitian product of the vectors x and y. The MAC can be applied directly on the generalised eigenvectors or on their expression on the FE model. A threshold has to be defined on this criterion in order to detect the collinearities. For more robustness in the detection process, it is possible to add criterion on the similitudes between the eigenvalues if the eigenvectors are detected to be collinear.

Using the MAC it is possible to detect merging of modes solved by the p-k method and then to adapt the computation strategy to try to resolve this aspect. In some cases, it can be done by changing a little bit the flight points (Mach, altitude, sweeping step in dynamic pressure, etc.) or the assumptions used to initialise the first guess from the previous solution. However, even for the cases for which such a strategy works, it adds a manual intervention which reduces greatly the efficiency of the automated process.

5 THE INVARIANT PAIRS THEORY

5.1 Some Definitions and Properties around the Invariant Pairs

To introduce the invariant pairs theory (exposed and developed in [4], [6]), it is necessary to consider a particular expression of the problem (5). The NLEVP needs to be put in a formulation similar to:

$$\left[\sum_{j=1}^{n} A_j f_j(p)\right] x = 0 \tag{10}$$

In which A_j are constant matrices and f_j are holomorphic scalar functions. Let $\Lambda \in \mathbb{C}^{k\times k}$ be a matrix with eigenvalues in $\Omega \subset \mathbb{C}$ and $X \in \mathbb{C}^{n\times k}$. The pair (X, Λ) is called invariant pair of the problem (10) if and only if:

$$\sum_{j=1}^{n} A_j X f_j(\Lambda) = 0 \tag{11}$$

In this relation, f_j are the extensions to the matrix case of the scalar f_j in (10). For instance, given a set of eigenvalues $(p_1, \dots, p_i)$ and eigenvectors $(x_1, \dots, x_i)$ solutions of the NLEVP (10), the pair (X, Λ) with $X = [x_1 \dots x_i]$ and $\Lambda = \begin{bmatrix} p_1 & & 0 \\ & \ddots & \\ 0 & & p_i \end{bmatrix}$ is a solution of the problem (11).

Let $(X, \Lambda) \in \mathbb{C}^{n\times k} \times \mathbb{C}^{k\times k}$ be an invariant pair. (X, Λ) is said to be minimal if there is an integer l such that:

$$V_l(X, \Lambda) = \begin{bmatrix} X \\ X\Lambda \\ \dots \\ X\Lambda^{l-1} \end{bmatrix} \tag{12}$$

has full column rank. The smallest integer verifying this property is called minimality index of (X, Λ). The minimality of a pair ensures that there is no null column in X. There are three important features about minimal invariant pairs:

- For any invertible matrix $Z \in \mathbb{C}^{k\times k}$, $(XZ, Z\Lambda Z^{-1})$ is also a minimal invariant pair solution of (11). This means that there is a basis in which Λ is diagonal.
- The eigenvalues of Λ are also eigenvalues of the NLEVP (10).
- If Z denotes the matrix of the eigenvectors of Λ, then XZ is the matrix of the eigenvectors solution of (10) associated to the eigenvalues of Λ.

Assuming that Λ has an eigenvalue of multiplicity 2 associated to the same eigenvector, then $V_l(X, \Lambda)$ does not have full column rank, hence (X, Λ) is not minimal. This demonstrates that the minimality of the invariant pair ensures that there is no merging of modes.

Let $\mathbb{T}$ and $\mathbb{V}$ be two matrix operators defined as follows:

$$\begin{aligned} \mathbb{T} : \mathbb{C}^{n\times k} \times \mathbb{C}^{k\times k} &\to \mathbb{C}^{n\times k} \\ (X, \Lambda) &\to \sum_{j=1}^{n} A_j X f_j(\Lambda) \\ \mathbb{V} : \mathbb{C}^{n\times k} \times \mathbb{C}^{k\times k} &\to \mathbb{C}^{k\times k} \\ (X, \Lambda) &\to W^H V_l(X, \Lambda) - I_k \end{aligned} \tag{13}$$

With W a normalisation matrix defined in (14) and I_k the identity matrix of size k.

$$W = V_l(X,\Lambda)[V_l(X,\Lambda)^H V_l(X,\Lambda)]^{-1} \tag{14}$$

Using the definition in (13), it appears that finding a minimal invariant pair solution of (11), is the same as finding a couple $(X,\Lambda) \in \mathbb{C}^{n\times k} \times \mathbb{C}^{k\times k}$ such that:

$$\mathbb{F}(X,\Lambda) = \begin{bmatrix}\mathbb{T}(X,\Lambda)\\ \mathbb{V}(X,\Lambda)\end{bmatrix} = 0 \tag{15}$$

Once a solution of (15) has been obtained the eigenpairs solution of the NLEVP (10) are determined thanks to a diagonalisation of Λ and a base transformation of X.

5.2 A Block Newton Method to Solve the Invariant Pairs Problem

In [9], a block Newton method is presented to solve (15). The main steps are recalled in this article. Let (X,Λ) be an estimate of the pair solution. To solve the problem one needs to determine $(\Delta X, \Delta\Lambda)$ in order that:

$$\begin{cases}\mathbb{T}(X+\Delta X,\Lambda+\Delta\Lambda) = 0\\ \mathbb{V}(X+\Delta X,\Lambda+\Delta\Lambda) = 0\end{cases} \tag{16}$$

By manipulating these equations, one can express through the Fréchet derivative of $\mathbb{T}$ and $\mathbb{V}$ at (X,Λ) toward $(\Delta X,\Delta\Lambda)$ a new system of matrix equations with the unknown $(\Delta X,\Delta\Lambda)$ in the left-hand side and only known parameters in the right-hand side. These relations hold after the small orders terms in ΔX and $\Delta\Lambda$ have been neglected (equivalent to the cut of the Taylor expansion in the p-k method):

$$\mathbb{D}\mathbb{F}(\Delta X,\Delta\Lambda) = \begin{bmatrix}\mathbb{D}\mathbb{T}(\Delta X,\Delta\Lambda)\\ \mathbb{D}\mathbb{V}(\Delta X,\Delta\Lambda)\end{bmatrix} \approx -\begin{bmatrix}\mathbb{T}(X,\Lambda)\\ \mathbb{V}(X,\Lambda)\end{bmatrix} \tag{17}$$

With $\mathbb{D}\mathbb{F}(\Delta X,\Delta\Lambda)$ denoting the Fréchet derivative of $\mathbb{F}$ at (X,Λ) in the direction of $(\Delta X,\Delta\Lambda)$ and so on for $\mathbb{D}\mathbb{T}$ and $\mathbb{D}\mathbb{V}$. From the first equation of (16) and the definition of the Fréchet derivative (see chapter 3 of [10]), it is possible to show that:

$$\mathbb{D}\mathbb{T}(\Delta X,\Delta\Lambda) = \mathbb{T}(\Delta X,\Lambda) + \sum_{j=1}^{n} A_j X \mathbb{D}f_j(\Lambda,\Delta\Lambda) \tag{18}$$

Where $\mathbb{D}f_j(\Lambda,\Delta\Lambda)$ is the standard notation for the Fréchet derivative of f_j at Λ in the direction $\Delta\Lambda$. Let W from (14) be $\begin{bmatrix}W_0\\ W_1\\ \vdots\\ W_{l-1}\end{bmatrix}$ with $W_i \in \mathbb{C}^{n\times k}$. The same can be done for the second line of (16), yielding:

$$\mathbb{D}\mathbb{V}(\Delta X,\Delta\Lambda) = W_0^H \Delta X + \sum_{i=1}^{l-1} W_i^H \left(\Delta X \Lambda^i + X \mathbb{D}\Lambda^i(\Lambda,\Delta\Lambda)\right) \tag{19}$$

The computation of ΔX and $\Delta\Lambda$ out of (17), (18) and (19) is possible due to the fact that Λ is at least upper triangular and for the flutter problem it even is diagonal which make some of the steps a little simpler than the ones developed in [9]. The main idea is to project the equations

on the vector $e_1 = [1\ 0\ ...\ 0]$ of the vector base allowing solving for the first eigenpair. Then, one needs to update the right-hand side of the problem and to project it on the remaining vectors of the base. It is possible to show that the structure of this reduced problem is the same as the main one, hence an iterative process along all the directions of the base to determine completely ΔX and $\Delta\Lambda$. Thanks to Λ being diagonal, one can even show that the update of the right-hand side is not necessary as opposed to the general case.

For each column, after the projection, the problem to solve takes the following form (expressed for the first eigenpair):

$$\begin{bmatrix} T(p_1) & \sum_{j=1}^{n} A_j X\left[\mathbb{D}f_j(\Lambda)\right]_{11} \\ \sum_{i=0}^{l-1} W_i^H p_1^i & \sum_{i=1}^{l-1} W_i^H X\left[\mathbb{D}\Lambda^i(\Lambda)\right]_{11} \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta p_1 \end{bmatrix} = -\begin{bmatrix} \mathbb{T}(X,\Lambda) \\ \mathbb{V}(X,\Lambda) \end{bmatrix} e_1 \tag{20}$$

With $\left[\mathbb{D}f_j(\Lambda)\right]_{11}$ defined by the matrix operator relation (the demonstration of this result is available in [9]):

$$f_j\left(\begin{bmatrix} \Lambda & I_k \\ 0 & p_1 I_k \end{bmatrix}\right) = \begin{bmatrix} f_j(\Lambda) & \left[\mathbb{D}f_j(\Lambda)\right]_{11} \\ 0 & f_j(p_1) I_k \end{bmatrix} \tag{21}$$

From an algorithmic point of view the resolution of (17) can be summarised as follows:

Algorithm 4: Resolution of the matrix equations (17)

For each column i of ΔX, $\Delta\Lambda$

1. Building of $\widetilde{\Lambda} = \begin{bmatrix} \Lambda & I \\ 0 & \Lambda_{ii} I \end{bmatrix}$
2. Resolution of (20)

Reassembly of the columns of ΔX and $\Delta\Lambda$

The algorithm for the block Newton method is then given below:

Algorithm 5: Block Newton method for given flight conditions

1. Choice of an initial pair (X_0, Λ_0)

Repeat

2. Minimality index computation l for the current pair
3. Computation of the residual $R_T = -\mathbb{T}(X_p, \Lambda_p)$
4. Resolution of $\left(\mathbb{D}\mathbb{T}(\Delta X, \Delta\Lambda), \mathbb{D}\mathbb{V}(\Delta X, \Delta\Lambda)\right) = (R_T, 0)$ [Algorithm 4]
5. Update $X_{p+1} = X_p + \Delta X$ and $\Lambda_{p+1} = \Lambda_p + \Delta\Lambda$
6. QR-factorisation of $V_l(X_{p+1}, \Lambda_{p+1}) = WR$
7. Normalisation $X_{p+1} = X_{p+1} R^{-1}$ and $\Lambda_{p+1} = R\Lambda_{p+1} R^{-1}$
8. Diagonalisation of $\Lambda_{p+1}$to get the eigenvectors matrix Z
9. Base transformation $X_{p+1} = X_{p+1} Z$ and $W = WZ$

Until convergence is reached

6 APPLICATION TO THE FLUTTER PROBLEM

6.1 Expression of the Flutter Problem into the Invariant Pairs Formalism

The flutter equation is detailed equation (3) with the standard notations. To use the invariant pairs theory, one needs to identify the matrices and functions to rewrite this equation in the formalism of equation (10).

For the structural terms, the identification is quite straightforward:

$$\begin{cases} A_1 = M, & f_1(p) = p^2 \\ A_2 = C, & f_2(p) = p \\ A_3 = K, & f_3(p) = 1 \end{cases} \tag{22}$$

As seen in Section 5.1, the f_j functions have to be holomorphic functions (it is the case for the functions in (22)) and it is necessary that a function of matrices can be derived from the scalar formulation. That second part is not a problem for the structural terms for which the matrix functions definition is as follows:

$$\begin{aligned} f_1 &: \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n} \\ & X \to X^2 \\ f_2 &: \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n} \\ & X \to X \\ f_3 &: \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n} \\ & X \to I_n \end{aligned} \tag{23}$$

The main difficulty arises from the aerodynamic terms. They have been computed at given Mach numbers and given reduced frequencies. Thus, some kind of interpolation has to be performed to get the GAF values for a specific Mach number and reduced frequency. For the sake of clarity, it is assumed that previous to a flutter computation at a given Mach number, the GAF are linearly interpolated as described in (24) (with $M_1 \leq Mach < M_2$):

$$GAF(k, Mach) = \frac{Mach - M_1}{M_2 - M_1} GAF(k, M_2) + \left(1 - \frac{Mach - M_1}{M_2 - M_1} GAF(k, M_1)\right) \tag{24}$$

This assumption allows reducing the dependencies of the GAF matrices to solely the reduced frequency k. In the following, $GAF(k_i)$ denotes the GAF matrices interpolated at the Mach number of interest. The linear interpolation required to get the GAF values at the reduced frequency of interest k (with $k_1 \leq k < k_2$) is given in (25), where the dependency of k with the Laplace variable has been expressed.

$$GAF(k) = \frac{\frac{Im(p)c}{V} - k_1}{k_2 - k_1} GAF(k_2) + \left(1 - \frac{\frac{Im(p)c}{V} - k_1}{k_2 - 1_1}\right) GAF(k_1) \tag{25}$$

From (25), and by defining A_j for j from 4 to $N_f + 3$ (N_f being the number of reduced frequencies available) such as $A_j = GAF\left(k_{j-3}\right)$, one can identify the functions f_j to adopt the invariant pairs formalism for the aerodynamic terms:

$$f_j(p) = \begin{cases} 1 \quad if \quad \dfrac{Im(p)c}{V} \le k_1 \quad and\ j = 4 \\ 1 \quad if \quad \dfrac{Im(p)c}{V} > k_{N_f}\ and\ j = N_f + 3 \\ 0 \quad if \quad 4 < j < N_f + 3\ and\ \left(\dfrac{Im(p)c}{V} < k_{j-4}\ or \dfrac{Im(p)c}{V} \ge k_{j-2}\right) \\ \dfrac{\dfrac{Im(p)c}{V} - k_{j-4}}{k_{j-3} - k_{j-4}} \quad if \quad k_{j-4} < \dfrac{Im(p)c}{V} \le k_{j-3}\ and\ 4 < j \\ 1 - \dfrac{\dfrac{Im(p)c}{V} - k_{j-3}}{k_{j-2} - k_{j-3}} \quad if \quad k_{j-3} < \dfrac{Im(p)c}{V} \le k_{j-2}\ \ and\ j < N_f + 3 \end{cases} \tag{26}$$

These functions are continuous and piecewise differentiable. Figure 7 illustrates some of them for $j = 4$ (middle), $j = N_f + 3$ (bottom) and a value of j between 5 and $N_f + 2$.

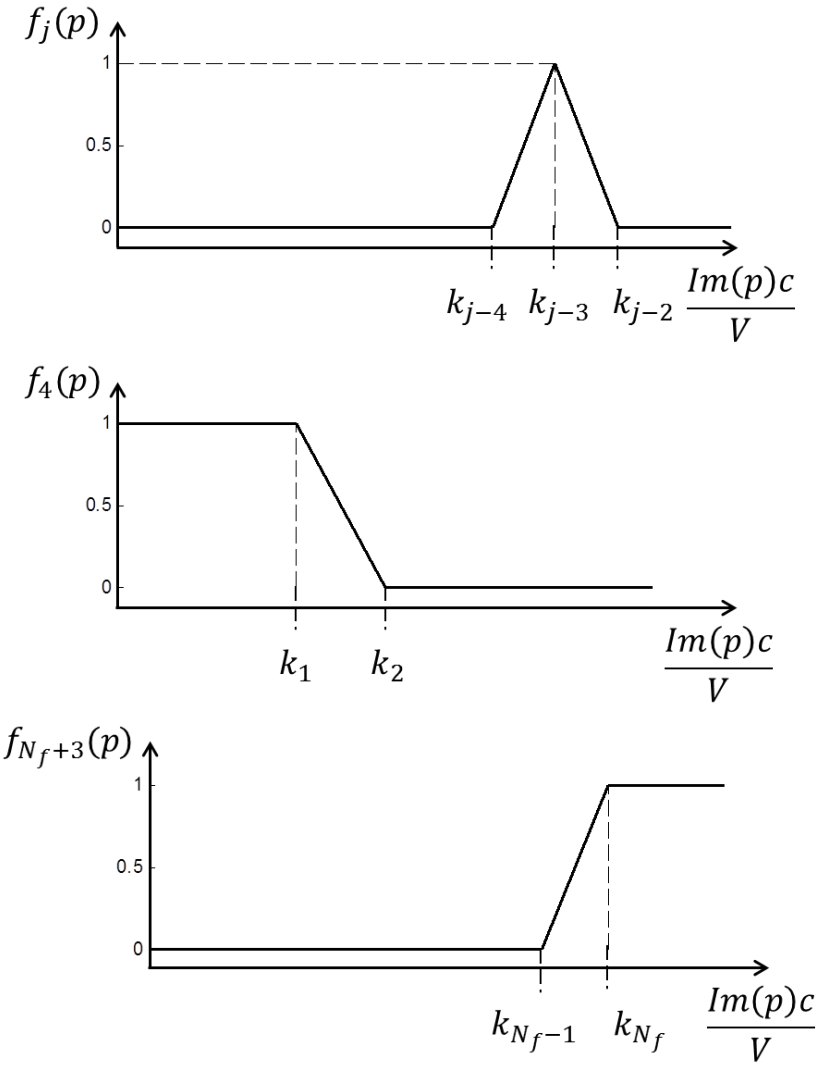

Figure 7: Shapes of various f_j functions associated to GAF terms.

The definition of the matrix functions from these scalar functions is not obvious. To overcome this difficulty, one needs to look at the methods to compute matrix functions from

scalar functions. The retained method to do so is based on the Jordan normal form of a matrix and the property presented in [10]. Every matrix $X \in \mathbb{C}^{n\times n}$ possess a Jordan normal form J:

$$Z^{-1}XZ = J = \begin{bmatrix} J_1 & & 0 \\ & \ddots & \\ 0 & & J_r \end{bmatrix} \tag{27}$$

With J_k the Jordan blocks:

$$J_k = J_k(p_k) = \begin{bmatrix} p_k & 1 & & 0 \\ & p_k & \ddots & \\ & & \ddots & 1 \\ 0 & & & p_k \end{bmatrix} \in \mathbb{C}^{m_k\times m_k} \tag{28}$$

This matrix is unique up to the order of Jordan blocks. Then, the property of interest for computing the matrix functions is as follows:

$$f(X) = Zf(J)Z^{-1} = Z\begin{bmatrix} f(J_1) & & 0 \\ & \ddots & \\ 0 & & f(J_r) \end{bmatrix}Z^{-1}$$

$$f(J_k) = \begin{bmatrix} f(p_k) & f'(p_k) & \dots & \frac{f^{(m_k-1)}(p_k)}{(m_k-1)!} \\ & f(p_k) & \ddots & \vdots \\ & & \ddots & f'(p_k) \\ 0 & & & f(p_k) \end{bmatrix} \tag{29}$$

Through the aerodynamic terms of the flutter equation, it appears that the invariant pairs formalism is rather constraining. This is one of the reason for which the application of this theory to complex cases (flutter of systems with FCS, non-linearities, etc.) can become tedious, hence slowing down its spread to industrial cases. Furthermore, the strategy based on the Jordan normal form developed above requires some properties on the scalar functions. The mathematic assumptions on this topic found in the literature are somewhat unequal from one source to another (see for instance the assumptions in [10] and the ones in [11], §11.1). For a more robust method to compute matrix functions from scalar functions, one might want to use the Parlett algorithm for matrix in the Schur block form. However, this strategy would require a more complex algorithm and imply a longer computational time.

In some rare cases, the method presented in this article might not converge due to some approximations made on the differentiability of the functions f_j. The addition of new terms to model feedback, FCS or some other kind of external forces to the flutter equation brings along an increase in the number of f_j functions, hence an increase in the risk to deal with non-verified assumptions degrading the convergence of the process.

6.2 Main Algorithm, Results and Performances

To solve the flutter problem by means of the invariant pairs theory and in particular of the block Newton method, the main algorithm is given below.

Algorithm 6: Resolution of the flutter problem by block Newton method

1. Computation of the modal characteristics "on ground" (with $\bar{q} = 0$)

For each flight point considered

2. Determination of (X_0, Λ_0) from previous points
3. Computation of (X, Λ) solution at the current flight point [Algorithm 5]
4. Sorting of modes by ascending frequencies

5. Sorting of modes over all the flight points
6. Flutter detection

This algorithm has been implemented in Elfini (in-house aero-structural solver platform at Dassault Aviation) and tested against several flutter cases displaying convergence problems, merging of modes and validation cases without any problem when solved with the p-k method. The cases considered range from small ones (2 modes) to industrial ones (up to 250 modes).

On cases without issue, the block Newton method finds the same results and diagrams as the p-k method. This observation allows validating the algorithms and their implementation in Elfini. As for the cases with issues, Figure 4 highlights the problem of merging modes when solving the flutter problem with the p-k method. This problem does not occur when using the block Newton method as shown Figure 8 (and Figure 9 for larger case).

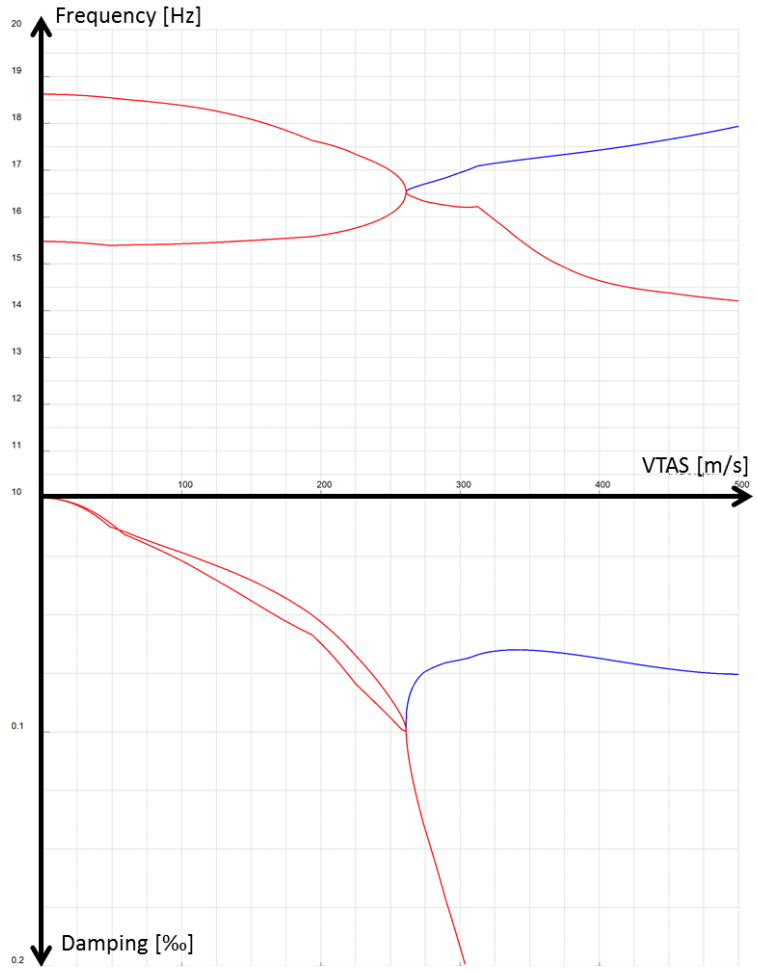

Figure 8: Flutter diagram computed with p-k method (red) and block Newton method (blue) on a small case.

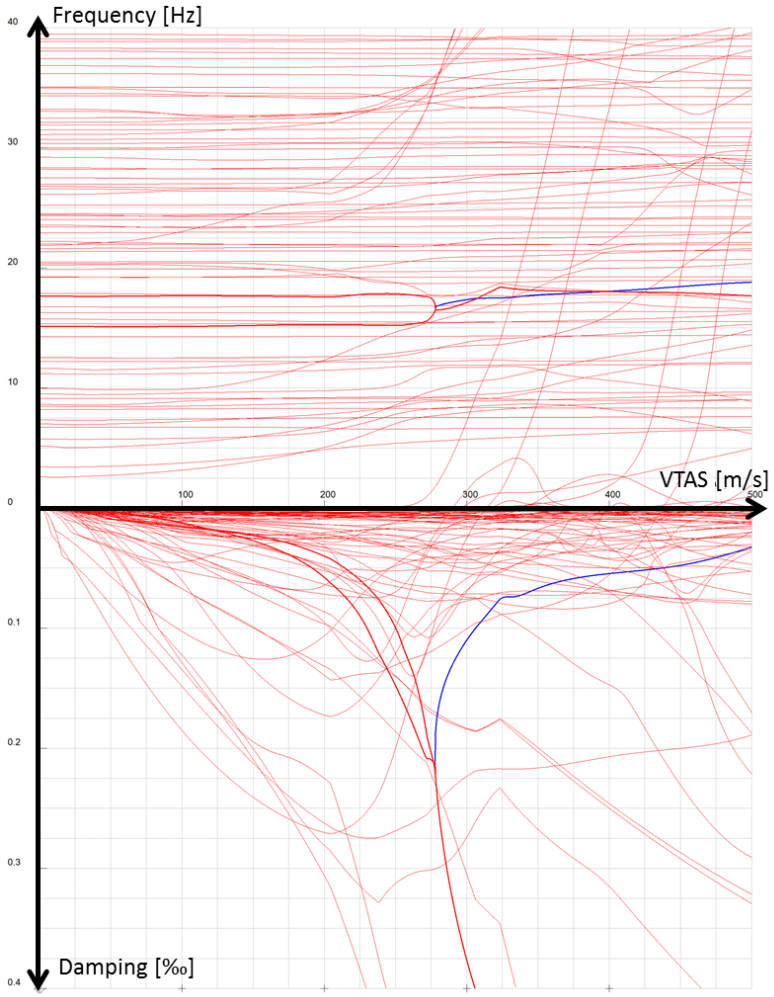

Figure 9: Flutter diagram obtained with p-k method (red) and block Newton method (blue) on an industrial case.

The main drawback of the block Newton method is its computational time which is longer than the p-k method one. A study on this aspect has been performed by increasing the number of modes computed and comparing the time between p-k method and block Newton method (Figure 10, the time of reference is the one obtained with the p-k to compute 10 modes). It appears that, as expected, the p-k method has a linear increase in the computational time with the number of modes computed. On the other hand, the block Newton method exhibits a quadratic or cubic increase. This highlights that this method might not be suitable for large cases in which numerous modes are computed.

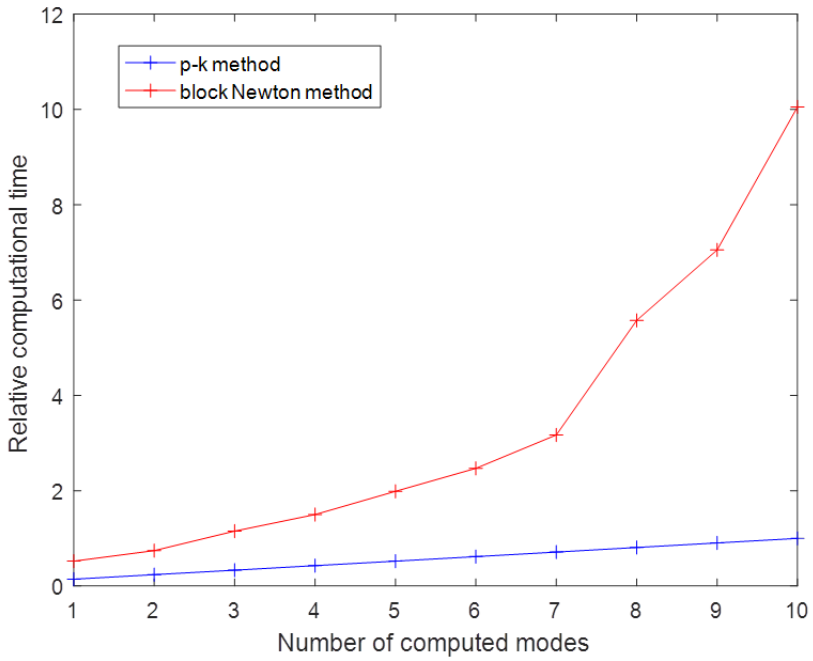

Figure 10: Comparison of relative computational times between p-k (blue) and block Newton (red) methods.

This flaw could be overcome by the use of more complex algorithms to solve the matrix system in the invariant pairs theory. Some of these algorithms are detailed in [6].

Another way to keep industrial computational times is to switch between p-k method and block Newton method along the various flight conditions to be computed. As long as no convergence issues or merging modes is detected, it is fine to use the p-k method and its low computational cost. Then, as soon as an issue is detected, the block Newton method can be used locally to solve only the non-converging or merging modes at a given flight point. Afterwards, for further flight points for which the modes are not close anymore, the p-k method is used again. This strategy produces a negligible increase in computational time.

7 CONCLUSION AND PERSPECTIVES

From the understanding of the reasons behind the merging of modes happening with the p-k method while solving the flutter problem, it has been possible to adapt a computational strategy based on the invariant pairs theory to avoid this issue. This new algorithm to solve the flutter problem has been implemented in Elfini, the in-house aero-structural solver platform at Dassault Aviation, and tested on several cases (both “academic” and industrial). It showed good promises to suppress the merging of modes despite an increased computational time. From this realisation, a hybrid solution has been derived in order to keep the computational time within industrial standard and still benefit from the features of the new algorithm regarding the problem at hand.

This hybrid method is now the standard method used at Dassault Aviation to solve the flutter problem, both on civilian and military aircraft. It allows automating the post-processing of the flutter computation with a larger confidence in the results and a smaller need for “manual” intervention.

Some improvements could be made to the developed method in order to help its convergence on complex cases (for instance the ones with feedback systems or FCS models). These improvements can be of two kinds, either on internal sub-algorithms (see the discussion

Section 6.1 on Jordan and Schur forms), or by implementing new method to solve the matrix system (method by extension, Arnoldi method, Jacobi-Davidson method, etc. as mentioned in [6] but requiring greater mathematical skills). Moreover, some developments regarding the formulation to use the invariant pairs theory on flutter problems might help to fully establish some mathematical proofs to sustain its accuracy and robustness.

8 REFERENCES

[1] Garrick, I. E. and Reed III, W. H. (1981). Historical Development of Aircraft Flutter. *Journal of Aircraft*, 18(11), 897-912.

[2] Garrigues E. (2018). A Review of Industrial Aeroelasticity Practices at Dassault Aviation for Military Aircraft and Businees Jets. *AerospaceLab Journal*, 14(09).

[3] Mehrmann, V. and Voss, H. (2004). Nonlinear eigenvalue problems: a challenge for modern eigenvalue methods. *GAMM-Mitteilungen*, 27(2), 121-152.

[4] Effenberger C. (2012). Robust successive computation of eigenpairs for nonlinear eigenvalue problems.

[5] Karpel M. and Strul E. (1996).Minimum-State Unsteady Aerodynamics with Flexible Constraints. *Journal of Aircraft*, 33(6), 1190-1196.

[6] Effenberger C. (2013). Robust solution methods for nonlinear eigenvalue problems. *Ecole polytechnique fédérale de Lausanne*.

[7] Nicot P. and Petiau C. (1989). Aeroelastic Analysis using Finite Element Models. *Proceedings from European Forum on Aeroelasticity and Structural Dynamics*.

[8] Panju M. (2011). Iterative Methods for Computing Eigenvalues and Eigenvectors. *The Waterloo Mathematics Review*, 1 (2011), 9-18.

[9] Kressner D. (2009). A block Newton method for nonlinear eigenvalue problems. *Research Report No. 2009-05, Seminar für Angewandte Mathematik Eidgenössische Technische Hochschule Zürich.*

[10] Higham N. J. (1961). Function of Matrices – Theory and Computation. *SIAM*.

[11] Golub G. H. and Van Loan C. F. (1996). Matrix Computations – Third Edition. *Baltimore and London: The Johns Hopkins University Press*.

COPYRIGHT STATEMENT