AIAA JOURNAL
Vol. 53, No. 3, March 2015

Topological Degree Applied to Flutter Equations

Edward E. Meyer*
Boreal Racing Shells, Seattle, Washington 98119

DOI: 10.2514/1.J053455

The solution of linear flutter equations carries the risk of missing important aeroelastic modes, particularly with large, complex models or control systems. A mathematical technique known as the topological degree of a system of nonlinear equations tells how many roots the equations have within a given region. Using the degree to determine the number of roots of the flutter determinant, regions, such as a velocity–frequency interval, can be tested for missed neutral-stability points. A generalized-bisection technique, together with the topological degree, provides a globally convergent nonlinear-equation solver requiring only function values. Applied to flutter equations, it is a safe way to narrow regions containing neutral-stability points or start points for aeroelastic mode-tracking processes in preparation for more efficient techniques, such as Newton's method.

Nomenclature

$\mathbb{C}^n$, $\mathbb{C}^{m\times n}$	=	complex n-vectors and m by n matrices
$\boldsymbol{D}$	=	dynamic matrix
d	=	structural damping coefficient
$\boldsymbol{J}_f$	=	Jacobian matrix of a function $\boldsymbol{f}$
M	=	Mach number
$\boldsymbol{M}$, $\boldsymbol{G}$, $\boldsymbol{K}$	=	mass, stiffness, gyroscopic matrices
n_s	=	order of the dynamic matrix
$\boldsymbol{P}_n$	=	region in $\mathbb{R}^n$
p	=	s/V, complex reduced frequency
q	=	dynamic pressure
$\boldsymbol{q}$	=	generalized-coordinate amplitudes
$\mathbb{R}^n$, $\mathbb{R}^{m\times n}$	=	real n-vectors and m by n matrices
$\Re z$, $\Im z$	=	real and imaginary parts of z
s	=	characteristic exponent $\sigma + i\omega$
sgn z	=	-1, 0, or 1 if z is negative, zero, or positive, respectively
V	=	velocity (true airspeed)
$\boldsymbol{V}$, $\boldsymbol{U}$, $\boldsymbol{T}$	=	viscous-damping, unsteady aerodynamic, and control-system matrices
$\boldsymbol{x}$	=	independent variables
x_i	=	ith element of vector $\boldsymbol{x}$
$\Delta\boldsymbol{J}$	=	determinant of matrix $\boldsymbol{J}$
$\boldsymbol{\rho c}$	=	real vector comprising the real and imaginary parts of c
σ	=	real part of the characteristic exponent
ω	=	oscillation frequency

I. Introduction

CONTINUATION methods provide a powerful method of solving flutter equations, including nonlinear and control-system terms [1]. Among the advantages are the ability to treat most formulations, such as the p-, p-k, and g-methods [2–4], and the near absence of mode-switching issues. More important, aeroelastic modes are tracked independently, aiding parameter studies and allowing selective mode tracking. Typically, a limited number of low-frequency modes are computed to save time, and because unsteady aerodynamic theories lose accuracy at higher frequencies. However, doing so runs the risk of missing important modes. Aircraft control surfaces are especially prone to selective mode tracking, having high natural frequencies, large aerodynamic forces, and possible interactions with control-system equations.

Received 7 March 2014; revision received 7 July 2014; accepted for publication 14 August 2014; published online 1 December 2014.

*Consultant; edward.e.meyer@borealracingshells.com.

Another way in which modes can be missed is when computing start points for a continuation method. A search for neutral stability usually begins with a search for solutions to the flutter equations at zero (or small) velocity. Without control-system equations and at zero velocity, start points are readily found using eigenvalue techniques. With control-system equations, linear eigenvalue techniques do not work and nonlinear eigenvalue techniques must be used, which are much less robust. Nonlinearities in the control-system equations can result in highly damped modes, creating difficulties for traditional nonlinear eigenvalue techniques [5].

Yet another way an important mode can be missed is if it does not extend to zero velocity, making it impossible to track in a traditional flutter analysis starting at zero velocity. Highly unlikely without control-system equations, this phenomenon is not only possible, but as an example will show, can hide an important mode.

What is needed is a way to determine if roots to flutter equations exist within a given region: if solutions exist in regions of $((V,\omega))$ with $\sigma = 0$ (flutter crossings), or regions of (σ,ω) at $V = 0$ (start points).

The topological degree is an integer related to the number of roots within a region of the domain of a system of nonlinear equations. Also known as the Brouwer degree, it has recently been used to study bifurcations [6] and boundary-value problems [7], extended to infinite-dimensional nonlinear problems [8], and used to study the stability of differential equations [9]. Applied to flutter equations, it can be used to ensure that regions in the flight regime do not contain flutter instabilities, and to search for start points in a more robust fashion.

II. Flutter Equation

The flutter equation considered here is a general form of the flutter characteristic equation:

$$\boldsymbol{D}\boldsymbol{q} = \boldsymbol{0} \in \mathbb{C}^{n_s}$$

$$\boldsymbol{D} = s^2\boldsymbol{M} + s(\boldsymbol{G}+\boldsymbol{V}) + (1+id)\boldsymbol{K} - q\boldsymbol{U}(p,M) + \boldsymbol{T} \in \mathbb{C}^{n_s\times n_s} \qquad (1)$$

in which $\boldsymbol{M}$, $\boldsymbol{G}$, $\boldsymbol{V}$, $\boldsymbol{K}$, $\boldsymbol{U}$, and $\boldsymbol{T}$ are the (n_s by n_s) mass, gyroscopic, viscous-damping, stiffness, unsteady aerodynamic, and control-system matrices, respectively; d is the structural damping coefficient; q is the dynamic pressure; $p = s/V$ is the complex reduced frequency; V is the airspeed; M is the freestream Mach number; $\boldsymbol{q}$ is a complex vector of generalized-coordinate amplitudes; and $\boldsymbol{D}$ is the (complex) dynamic matrix. Interest usually centers on the real part σ of the characteristic exponent $s = \sigma + i\omega$: negative values indicate stability and decaying oscillations, positive values indicate growing oscillations and instability, and zero means the mode is neutrally stable. The search for conditions of neutral stability, the boundary between stability and instability, and the behavior of the points of neutral stability with various parameters are the primary focus on the flutter equation.

Nontrivial ($\boldsymbol{q} \neq \boldsymbol{0}$) solutions to Eq. (1) require the determinant of $\boldsymbol{D}$ to be zero. In the past, the determinant has been used in various solution techniques [2]. Searching for solutions to the flutter equation as determinant zeros cannot be recommended in general because the determinant exhibits rapidly changing, extremely large and extremely small values, even for small ($n_s < 10$) problems. Regardless, the determinant serves well for the procedure presented here because the magnitude is not important, only the complex argument (angle).

The complex determinant can be written as a system of two real equations in two unknowns:

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} = \boldsymbol{\rho}\Delta\boldsymbol{D}(\boldsymbol{x}) = \begin{Bmatrix} \Re\Delta\boldsymbol{D}(\boldsymbol{x}) \\ \Im\Delta\boldsymbol{D}(\boldsymbol{x}) \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \tag{2}$$

in which $\Delta\boldsymbol{D}(\boldsymbol{x})$ is the complex determinant of the dynamic matrix evaluated at $\boldsymbol{x}$, $\boldsymbol{\rho}$ is a real vector comprising the real and imaginary parts of its complex argument, and $\boldsymbol{x}$ comprises two parameters. Any two independent parameters are acceptable; for simplicity, two of the three parameters, velocity, frequency, and σ, will be used in examples, and the third held constant using the shorthand notation:

$$\boldsymbol{x} = \begin{Bmatrix} V \\ \omega \end{Bmatrix}_{\sigma=0} \qquad \boldsymbol{x} = \begin{Bmatrix} \sigma \\ \omega \end{Bmatrix}_{V=0} \tag{3}$$

for flutter crossings and start points, respectively.

Regular solutions to Eq. (2) are those in which $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$, and the Jacobian matrix

$$\boldsymbol{J}_f = \boldsymbol{f}'(\boldsymbol{x}) = \left[\frac{\partial f_i}{\partial x_j}\right] \in \mathbb{R}^{2\times 2} \tag{4}$$

is nonsingular, that is, the Jacobian determinant $\Delta\boldsymbol{J}_f \neq 0$. A singular Jacobian matrix indicates a bifurcation, and will not be considered here. Note also that, although $\Delta\boldsymbol{D}$ is complex, $\Delta\boldsymbol{J}_f$ is real.

A further restriction is necessary because the generalized coordinates are not included in Eq. (2); the dynamic matrix may be a function of at most the magnitude of one generalized coordinate using a quasi-linear technique [1].

Under certain conditions, the determinant of the dynamic matrix is an analytic function of the characteristic exponent s, satisfying the Cauchy–Riemann equations ([10] p. 39):

$$\frac{\partial f_1}{\partial \sigma} = \frac{\partial f_2}{\partial \omega} \quad \text{and} \quad \frac{\partial f_1}{\partial \omega} = -\frac{\partial f_2}{\partial \sigma} \tag{5}$$

If the elements of the dynamic matrix are analytic, the determinant, computed by simple arithmetic operations on the elements, also must be. In the absence of aerodynamics $\boldsymbol{U}$ and control systems $\boldsymbol{T}$, the flutter equation is simply a polynomial in s, and the dynamic matrix is analytic. Unsteady aerodynamics are, in general, analytic; however, the usual practice is to compute unsteady aerodynamic matrices at a few values of $p = s/V$ or $k = \omega/V$, and fit interpolation or approximation functions for intermediate values. Interpolation on k is not analytic, except with the g-method modification [3], and then only at $\sigma = 0$. Rational-function approximation [11] is analytic, as are certain types of interpolation on p. Control-system matrices are usually analytic, for example, when derived from a linear time-invariant state-variable form or rational polynomials in s [1].

III. Topological Degree

The topological degree (or simply degree) of a system of n equations in n unknowns

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0} \in \mathbb{R}^n \tag{6}$$

relative to a region $P_n \in \mathbb{R}^n$ is an integer, which, if it is nonzero, indicates there is at least one root within the region satisfying Eq. (6). More precisely, it is the number of roots with positive Jacobian determinant, $\Delta\boldsymbol{J}_f$, minus the number of roots with negative Jacobian determinant, or ([12] p. 148)

$$\deg(\boldsymbol{f}, P_n) = \sum_{k=1}^{p} \operatorname{sgn}\Delta\boldsymbol{J}_f(\bar{\boldsymbol{x}}^k) \tag{7}$$

in which the function sgn t is $+1$, -1, or zero, if t is positive, negative, or zero, respectively; and the $\bar{\boldsymbol{x}}^k$ are the roots within the region P_n. The topological degree is a generalization of the winding number ([13] p. 41) also known as the argument principle in complex-variable theory ([10] p. 298), the number of roots minus the number of poles in the region.

Equation (7) describes the degree in terms of derivatives at roots of the equations, thus is not useful for computing the degree. More useful are descriptions in terms of function values and derivatives on the region boundary. The topological degree was first formulated by Kronecker as an $(n-1)$-dimensional integral over the boundary of the region, and is sometimes referred to as the Kronecker or degree integral. Provided no roots are on the boundary $b(P_n)$ of the region P_n, the degree of $\boldsymbol{f}$ in P_n is [14]

$$\deg(\boldsymbol{f}, P_n) = \frac{(n/2-1)!}{2\pi^{n/2}} \int_{b(P_n)} \sum_{i=1}^{n} a_i \, \mathrm{d}x_1 \ldots \mathrm{d}x_{i-1}\mathrm{d}x_{i+1} \ldots \mathrm{d}x_n \tag{8}$$

in which the a_i are determinants of modifications of the Jacobian matrix

$$a_i = \Delta\frac{(-1)^{n(i-1)}}{(\boldsymbol{f}^T\boldsymbol{f})^{n/2}}\left[\boldsymbol{f}\frac{\partial \boldsymbol{f}}{\partial x_1}\cdots\frac{\partial \boldsymbol{f}}{\partial x_{i-1}}\frac{\partial \boldsymbol{f}}{\partial x_{i+1}}\cdots\frac{\partial \boldsymbol{f}}{\partial x_n}\right] \tag{9}$$

A. Computing the Degree

The fact that the degree is an integer makes it possible to greatly simplify Eq. (8). Stenger [15] presents a simple method for computing the degree, which requires only function values on the boundary of the region. The boundary is divided into m $(n-1)$-dimensional oriented simplexes. Two-dimensional (2-D) ($n = 2$) boundary simplexes are line segments between two points ordered clockwise around the region, and three-dimensional (3-D) ($n = 3$) boundary simplexes are triangles with surface normals facing out of the region.

Given m oriented simplexes, the degree is

$$\deg(\boldsymbol{f}, P_n) = \frac{1}{2^n n!}\sum_{k=1}^{m} \Delta \operatorname{sgn}\boldsymbol{B}^k \tag{10}$$

in which the (n, n) matrices $\boldsymbol{B}^k$ are

$$\boldsymbol{B}^k = [\boldsymbol{f}(\boldsymbol{x}_1^k) \ldots \boldsymbol{f}(\boldsymbol{x}_n^k)] \in \mathbb{R}^{n\times n} \tag{11}$$

$\boldsymbol{x}_i^k$ is the ith point of the kth simplex, sgn $\boldsymbol{B}^k$ is a matrix with each element B_{ij}^k of $\boldsymbol{B}^k$ replaced with sgn B_{ij}^k, and Δ sgn $\boldsymbol{B}^k$ is an integer ranging from $-n!$ to $+n!$.

Returning to the flutter problem [Eq. (2)], $n = 2$ and the degree is

$$\deg(\boldsymbol{f}, P_2) = \frac{1}{8}\sum_{k=1}^{m} \Delta \operatorname{sgn}[\,\boldsymbol{f}(\boldsymbol{x}_1^k) \quad \boldsymbol{f}(\boldsymbol{x}_2^k)\,] \tag{12}$$

in which $\boldsymbol{x}_j^k$, $j = 1, 2$ are the two points of segment k.

It remains to determine how many segments, or how small each segment must be, to ensure the correct value of degree. Stenger [15] shows that, for 2-D problems, each segment must be small enough that at most one of f_1 or f_2 changes sign on each segment, equivalent to moving at most one quadrant. Furthermore, Eq. (12) must yield an integer, and so the summation must yield a multiple of 8; anything else is an indication that the segments are too large. Given these

requirements, a reasonable strategy for dividing the boundary is shown in Algorithm 1.

B. Creating a 2-D Grid

Algorithm 1 shows how to create a 2-D grid as follows:

Algorithm 1 Two-dimensional grid creation

1: The four sides of the region are the initial segments.
2: Proceed counterclockwise around the region.
3: For each segment
  a) Compute $\boldsymbol{f}(\boldsymbol{x})$ at the endpoints of the segment.
  b) Test the signs of f_1 and f_2 at the endpoints; if both change sign, divide the segment and repeat steps 3a and 3b for both new segments.
4: Sum the determinants of the signs of the function values [Eq. (12)].
5: If the sum is not a multiple of 8, divide each segment, and repeat steps 4 and 5.

C. Example

HA145b is a demonstration problem included in MSC NASTRAN [16], originally presented as example 9-1 in [17]. This tapered aircraft wing is often used to demonstrate flutter formulations and solution techniques. Here, the model is built using 10 free-vibration modes as the generalized-coordinate basis ($n_s = 10$), and unsteady aerodynamic matrices at seven values of reduced frequency ω/V interpolated with cubic splines. The results of tracking only modes starting between 4 and 12 Hz (two modes) are shown in Fig. 1, with the lowest flutter crossing at point F. Two modes that were not tracked but should have been are shown dashed: a 3 Hz mode, which crosses at point A, and a 15 Hz hump mode going unstable at B and stable at C. Velocity has been normalized with respect to the lowest computed flutter speed V_F.

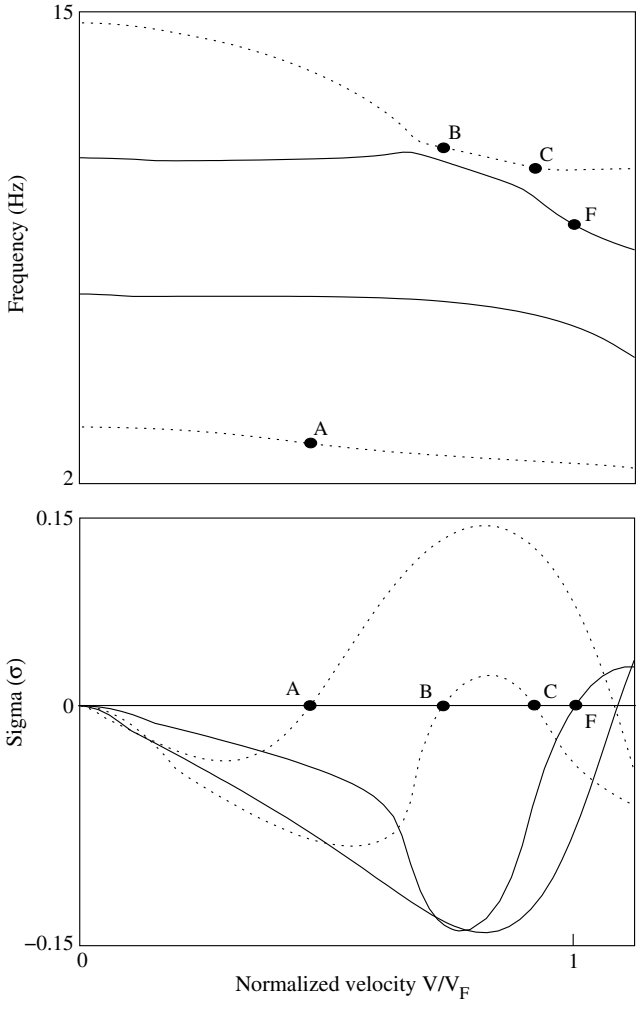

Fig. 1 Flutter solution with missed modes.

Modeling a real-life scenario, the lowest flutter speed has been computed as V_F by tracking a limited number of modes. To verify that no lower flutter crossings have been missed, the degree is computed for a region

$$P_2 = \begin{Bmatrix} V/V_F \\ \omega \end{Bmatrix}_{\sigma=0} = \begin{Bmatrix} 0.1{:}0.9 \\ 2{:}12 \end{Bmatrix}_{\sigma=0} \tag{13}$$

with normalized velocity ranging from 0.1 to 0.9 and frequency from 2 to 12 Hz, as shown in Fig. 2.

Applying Algorithm 1 to this region divides the region boundary into segments shown in Fig. 3. The resulting degree is 1, indicating at least one root in the region, when, in fact, there are three.

Evidently, the degree failed to account for two of the roots in the region. The reason goes back to Eq. (7), which states that the degree is the number of roots with positive Jacobian determinant minus the number with negative determinant. The two roots for the hump mode (points B and C) cancel each other with Jacobian determinants of opposite signs.

To see why this is so, consider Eq. (7) with a region P_2 containing a single root $\bar{\boldsymbol{x}} = \left[\bar{V}, \bar{\omega}\right]^T$:

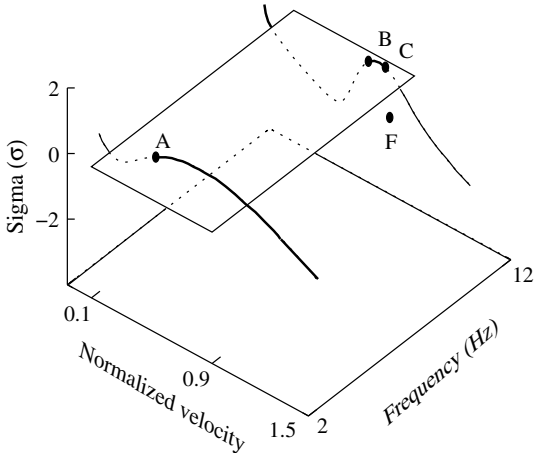

Fig. 2 Region for flutter search.

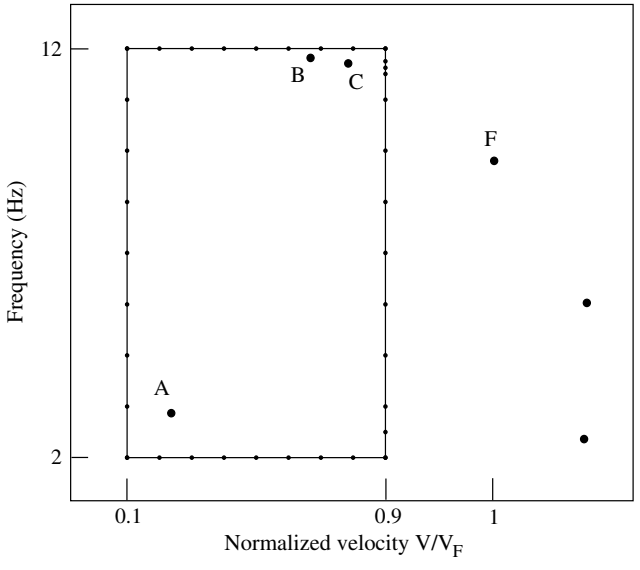

Fig. 3 Boundary points.

$$\deg(\boldsymbol{f}, P_2) = \operatorname{sgn}\left[\frac{\partial f_1}{\partial V}\frac{\partial f_2}{\partial \omega} - \frac{\partial f_1}{\partial \omega}\frac{\partial f_2}{\partial V}\right]_{\bar{\boldsymbol{x}}}$$
$$= \operatorname{sgn}\left[\frac{\partial f_1}{\partial \sigma}\frac{\partial \sigma}{\partial V}\frac{\partial f_2}{\partial \omega} - \frac{\partial f_1}{\partial \omega}\frac{\partial f_2}{\partial \sigma}\frac{\partial \sigma}{\partial V}\right]_{\bar{\boldsymbol{x}}} \tag{14}$$

If the dynamic matrix is an analytic function of $s = \sigma + i\omega$, the determinant is also an analytic function, and $\boldsymbol{f}$ satisfies the Cauchy–Riemann equations [Eq. (5)], which, when substituted into Eq. (14)

$$\deg(\boldsymbol{f}, P_2) = \operatorname{sgn}\left[\frac{\partial \sigma}{\partial V}\left(\frac{\partial f_1}{\partial \sigma}\right)^2 + \frac{\partial \sigma}{\partial V}\left(\frac{\partial f_2}{\partial \sigma}\right)^2\right]_{\bar{\boldsymbol{x}}}$$
$$= \operatorname{sgn}\frac{\partial \sigma}{\partial V}(\bar{\boldsymbol{x}}) \tag{15}$$

and so the degrees for unstable crossings ($\partial\sigma/\partial V > 0$) and stable crossings ($\partial\sigma/\partial V < 0$) have opposite signs, and they cancel each other in the summation of Eq. (7). Even though the dynamic matrix is not quite analytic in this example due to the aerodynamic matrices, it is close enough that this analysis holds.

Clearly, this is unacceptable: two important flutter crossings have not been accounted for by the degree. The solution is to use a modification of the system of equations, which makes the Jacobian determinant always positive, and so the degree always gives the total number of roots in a region.

IV. Total Number of Roots

As presented previously, the topological degree is inconclusive with regard to the number of roots in a region. A nonzero value means the region contains at least one root, but a zero value does not ensure there are no roots, only that the number with positive Jacobian determinants equals the number with negative Jacobian determinants.

This limitation can be overcome with a simple addition to the equations, which ensures that the Jacobian determinant is positive at all points in the region, in which $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$. Originally, due to Picard [18], and more recently promoted by Hoenders and Slump [19], the idea is to add an equation, $z\Delta\boldsymbol{J}_f = 0$, which has a solution only at $z = 0$, to the system of equations:

$$\boldsymbol{g}(\boldsymbol{y}) = \begin{Bmatrix} \boldsymbol{f}(\boldsymbol{x}) \\ z\Delta\boldsymbol{J}_f \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{0} \\ 0 \end{Bmatrix} \in \mathbb{R}^{n+1} \tag{16}$$

with

$$\boldsymbol{y} = \begin{Bmatrix} \boldsymbol{x} \\ z \end{Bmatrix} \in \mathbb{R}^{n+1} \tag{17}$$

At a regular solution of Eq. (16), $z = 0$, and the Jacobian matrix of $\boldsymbol{g}(\boldsymbol{y})$ is

$$\boldsymbol{J}_g = \boldsymbol{g}'(\boldsymbol{y}) = \begin{bmatrix} \boldsymbol{J}_f & \boldsymbol{0} \\ \boldsymbol{0} & \Delta\boldsymbol{J}_f \end{bmatrix} \in \mathbb{R}^{n+1\times n+1} \tag{18}$$

The determinant of $\boldsymbol{J}_g$ is $(\Delta\boldsymbol{J}_f)^2$, which is always positive provided $\boldsymbol{J}_f$ is not singular, and so by Eq. (7), the topological degree of $\boldsymbol{g}(\boldsymbol{y})$ in a region containing $z = 0$ is the total number of roots.

Some confusion might arise from the fact that three determinants are involved with the application of degree to the flutter problem: the complex determinant of the (n_s, n_s) dynamic matrix $\Delta\boldsymbol{D}$, the integer determinant of a (2,2) matrix of signs of the dynamic-matrix determinant ($\Delta \operatorname{sgn} \boldsymbol{B}^k$), and the determinant of the Jacobian matrix of the determinant of the dynamic matrix ($\Delta\boldsymbol{J}_f$).

The Jacobian matrix of the determinant of the dynamic matrix [Eq. (2)]

$$\boldsymbol{J}_f = \boldsymbol{f}'(\boldsymbol{x}) = \left[\boldsymbol{\rho}\frac{\partial(\Delta\boldsymbol{D})}{\partial x_1}\boldsymbol{\rho}\frac{\partial(\Delta\boldsymbol{D})}{\partial x_2}\right] \in \mathbb{R}^{2\times 2} \tag{19}$$

requires derivatives of the determinant of the dynamic matrix ([20] p. 360)

$$\frac{\partial(\Delta\boldsymbol{D})}{\partial x_i} = \Delta\boldsymbol{D}\,\operatorname{tr}\left[\boldsymbol{D}^{-1}\frac{\partial\boldsymbol{D}}{\partial x_i}\right] \qquad i = 1, 2 \tag{20}$$

in which tr, the trace of a matrix, is the sum of the diagonals.

Applying the Picard extension to the flutter problem increases the number of equations from two to three, and Eq. (10) becomes

$$\deg(\boldsymbol{g}, P_3) = \frac{1}{48}\sum_{k=1}^{m}\Delta \operatorname{sgn} \boldsymbol{B}^k \tag{21}$$

in which the (3,3) matrices $\boldsymbol{B}^k$ are

$$\boldsymbol{B}^k = [\,\boldsymbol{g}(\boldsymbol{y}_1^k) \quad \boldsymbol{g}(\boldsymbol{y}_2^k) \quad \boldsymbol{g}(\boldsymbol{y}_3^k)\,] \in \mathbb{R}^{3\times 3} \tag{22}$$

and $\boldsymbol{y}_1^k$, $\boldsymbol{y}_2^k$, and $\boldsymbol{y}_3^k$ are the vertices of simplex k. Thus, the natural discretization of the surface is a set of triangles, oriented with outward normals.

Creating this 3-D grid is greatly simplified by observing that the first two equations of $\boldsymbol{g}(\boldsymbol{y})$ are independent of z, and the third equation ($z\Delta\boldsymbol{J}_f$) is a linear function of z. Because the limits of z are arbitrary as long as they contain $z = 0$, take the limits to be ± 1, and an approach to creating a 3-D grid can be outlined as Algorithm 2 (see Fig. 4).

A. Creating a 3-D Grid

Algorithm 2 shows how to create a 3-D grid as follows:

Algorithm 2 Three-dimensional grid creation

1: Create a 2-D grid using Algorithm 1.
2: Add points as necessary to create a rectangular 2-D grid by connecting points on opposite sides.
3: Compute $\boldsymbol{J}_f$ [Eqs. (19) and (20)] at each point on the rectangular grid.
4: Duplicate this grid, take one as the top, the other as the bottom of the 3-D grid.
5: Scale the third element of $\boldsymbol{g}$ by -1 on the bottom grid.
6: Connect each point on the outer boundary of the bottom grid to the corresponding point on the top.
7: Connect a diagonal of each resulting rectangle to create triangular simplexes, ordering the points so that the surface normal points outward.
8: Sum the determinants of the signs of the function values [Eq. (21)].
9: If the sum is not a multiple of 48, divide each segment of the 2-D grid in step 1, and repeat steps 2–9.

B. Example

Returning to the previous example, which showed how stable and unstable crossings cancel in the count of roots, and using Eq. (21) with Picard's extension and the 3-D region in Fig. 4 result in a degree of 3, reflecting the fact that the region has three roots, and the stable and unstable have not canceled each other. Thus, it is necessary when testing regions for flutter crossings to use Picard's extension and a 3-D region.

V. Generalized Bisection

In addition to determining if a region contains roots to a set of equations, the topological degree can be used to locate roots to within any desired precision using a generalization of the well-known bisection method for single equations.

Bisection methods are used to find zeros of single-variable ($n = 1$) functions using the observation that, if the function changes sign between two points, there must be a root between them. Splitting the interval and testing the signs on each subinterval, and repeating with the subinterval containing the root eventually result in a sufficiently narrow interval. The number of sign changes in an interval is analogous to the topological degree for multidimensional functions. As with multidimensional functions, a nonzero degree means the interval contains a root, but a zero degree only means the function has

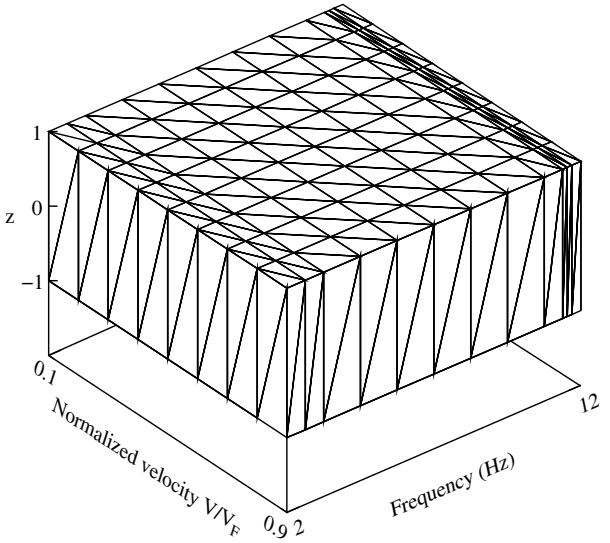

Fig. 4 Three-dimensional region for flutter search.

passed zero an equal number of times with positive and negative slopes. Associating the slope with the multidimensional Jacobian determinant completes the analogy.

Extending the bisection method to multidimensional functions, using the degree in place of sign changes, results in a generalized-bisection method. For the flutter problem, there are two cases to consider: two dimensions using the degree, and three dimensions using the degree and the Picard extension. In spite of the problem with stable and unstable flutter crossings discussed previously, the 2-D generalized bisection is useful for finding start points, as discussed below.

A. 2-D Generalized Bisection

The steps involved can be summarized in Algorithm 3.

Algorithm 3 Two-dimensional generalized bisection

1: Beginning with a 2-D region with nonzero degree and a tolerance ϵ,
2: Divide the region by dividing the largest side creating two new regions.
3: For each region
  a) Compute the degree of the function relative to the region.
  b) If the degree is zero, discard the region.
  c) If the largest side is smaller than ϵ and the degree is ± 1, save the region on a list of regions containing roots; otherwise, do steps 2 and 3 for this region.

B. 3-D Generalized Bisection

With Picard's extension, the flutter problem becomes 3-D, and generalized bisection takes the following form of Algorithm 4:

Algorithm 4 Three-dimensional generalized bisection

1: Given a tolerance ϵ and a 2-D region, build a 3-D region with Algorithm 2.
2: Compute the degree with Eq. (21).
3: If the degree is 1 and the region size is $\lesssim\epsilon$, save the region on a list of regions containing roots.
4: If the degree is not zero, divide the 2-D region by dividing the longest side, and repeat steps 1–4 for each new region.

C. Newton's Method

Generalized bisection, as applied to the flutter equation, has a couple of drawbacks. It is relatively expensive, and it does not produce the generalized coordinates at the root, required for a continuation process. Its advantage is global convergence, the ability to converge starting from large regions. With these factors in mind, a reasonable strategy is to use generalized bisection to reduce the region containing a root small enough that Newton's method converges. Just how small is difficult to predict, but a trial-and-error approach works well: if a Newton iterate falls outside the region, return to generalized bisection; otherwise, continue until the Newton iterates converge.

Newton's method applied to Eq. (1) can be used to solve for the generalized coordinates in addition to completing the convergence to the root sought with generalized bisection. Adding a normalization equation to the flutter equation, and combining the two real parameters used in the generalized bisection with the generalized coordinates converted to a real vector result in the following $2n_s + 2$ real equations in $2n_s + 2$ unknowns:

$$\boldsymbol{h}(z) = \rho \begin{Bmatrix} \boldsymbol{Dq} \\ \boldsymbol{q}^T\boldsymbol{q} - 1 \end{Bmatrix} = \boldsymbol{0} \in \mathbb{R}^{2n_s+2} \qquad z = \begin{Bmatrix} \boldsymbol{x} \\ \boldsymbol{\rho}(\boldsymbol{q}) \end{Bmatrix} \in \mathbb{R}^{2n_s+2} \tag{23}$$

Newton's method can then be written as

$$z^{i+1} = z^i - \boldsymbol{J}_h^{-1}\boldsymbol{h}(z^i) \qquad i = 0, 1, \ldots \qquad z^0 = \begin{Bmatrix} \boldsymbol{x}^0 \\ \boldsymbol{\rho}(0) \end{Bmatrix} \tag{24}$$

in which $\boldsymbol{x}^0$ is the center of the region, and the Jacobian matrix is

$$\boldsymbol{J}_h = \boldsymbol{h}'(z) = \left[\frac{\partial h_i}{\partial z_j}\right] \in \mathbb{R}^{2n_s+2\times 2n_s+2} \tag{25}$$

Bisection converges linearly, reducing the size of the region, hence the error by one-half each iteration. On the other hand, Newton's method is known to converge quadratically. In addition to faster convergence, Newton's method requires much less work for each iteration: the factorization of a $(2n_s + 2, 2n_s + 2)$ real matrix, in contrast to the factorization of a (n_s, n_s) complex matrix at each grid point, numbering perhaps 20 to over 100.

D. Example

Missed modes are not always because of a limited number of start points, as in the example of Fig. 1. A remarkable, and fortunately, rare example of a missed instability that can be found with the topological degree comes from a complex 190 degree-of-freedom (DOF) model, including a sixth-order rational-polynomial control system with six poles. Figure 5 shows 17 modes starting between 0 and 4 Hz frequency, and the corresponding V-σ curves with velocity normalized to the lowest computed flutter crossing at point F and the mode that was missed, shown as a dashed curve.

Except for the missed mode, these appear to be ordinary plots of velocity vs σ and frequency. What makes the missing mode remarkable is that it does not extend to zero velocity. Tracing the aeroelastic modes from zero velocity misses this mode; worse, it is an unstable mode. The highly nonlinear set of control equations in the $\boldsymbol{T}$ matrix is the likely cause of this strange mode.

The missing mode was found using 3-D generalized bisection and degree based on the 2-D region:

$$P_2 = \begin{Bmatrix} V/V_F \\ \omega \end{Bmatrix}_{\sigma=0} = \begin{Bmatrix} 0{:}0.6 \\ 0.5{:}4 \end{Bmatrix}_{\sigma=0} \tag{26}$$

shown in Fig. 6.

Generalized bisection progresses toward the root, as shown in Fig. 7. Bisection stops when the box is small enough that iterations of Newton's method [Eq. (24)] stay within the box and converge to the root.

Once the root is converged, it is used as a start point for a continuation process [1], tracking the mode first in one direction, then the other, resulting in the dashed curve in Fig. 5.

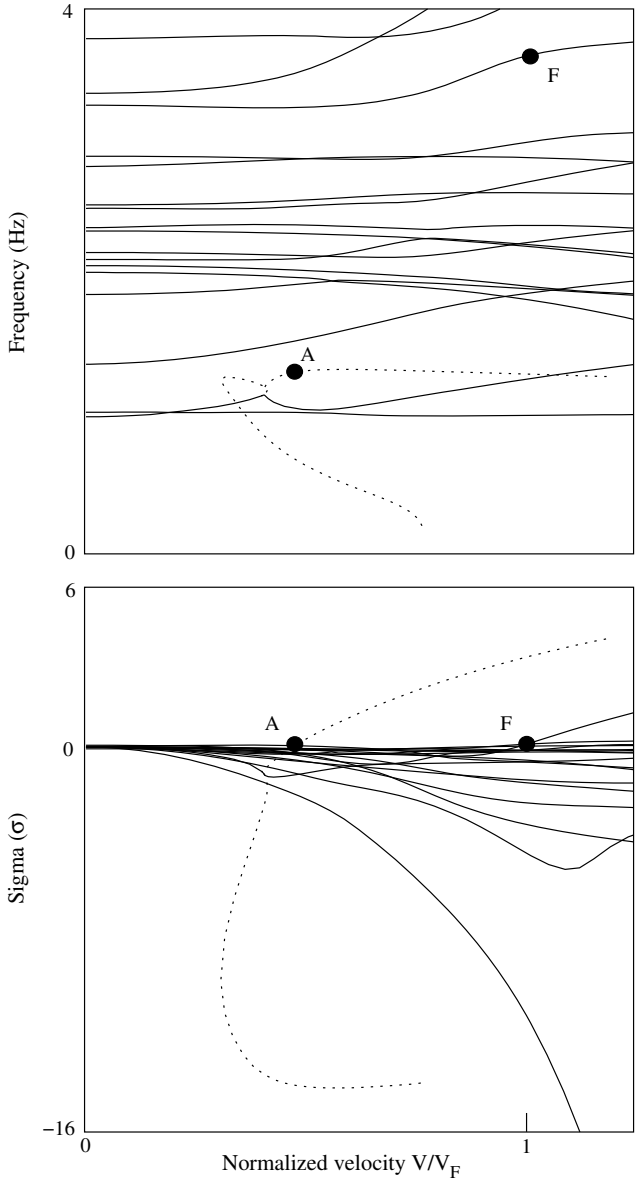

Fig. 5 Flutter solution with a missing mode.

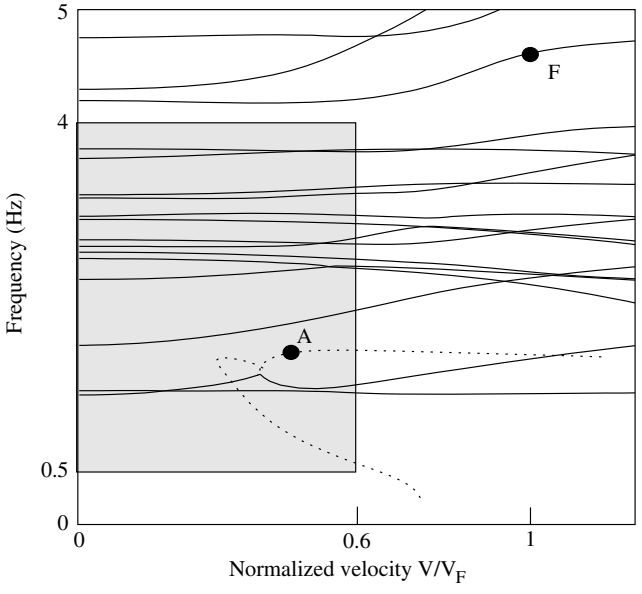

Fig. 6 Generalized-bisection search region.

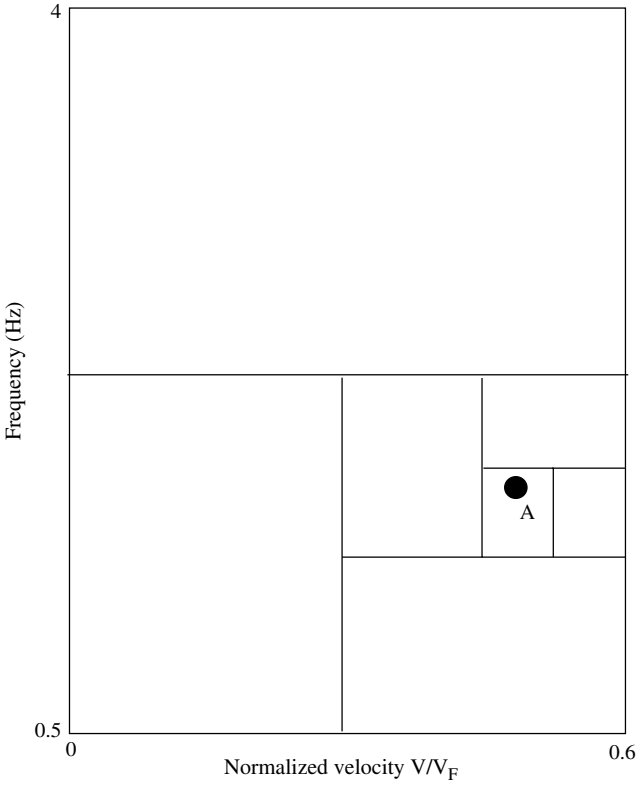

Fig. 7 Generalized-bisection evolution.

VI. Start Points with Control-System Equations

Continuation methods require known solutions to the equations to start the tracking process. A search for neutral stability generally starts at zero velocity, and proceeds until some parameter limit, for example, velocity or frequency, is reached. In the absence of control-system equations, the flutter equation at zero velocity reduces to a polynomial eigenvalue problem if gyroscopic or viscous-damping terms are included, or a generalized eigenvalue problem if they are not, both of which have robust solution techniques.

If the flutter equation includes a control-system matrix $\boldsymbol{T}(s)$, which is a function of s, the problem to be solved is a nonlinear eigenvalue problem:

$$\boldsymbol{D}(s)\boldsymbol{q} = \boldsymbol{0} \in \mathbb{C}^{n_s} \tag{27}$$

in which the eigenvalues are the characteristic exponents s, and the eigenvectors are the generalized-coordinate amplitudes $\boldsymbol{q}$. The techniques for the nonlinear eigenvalue problem [5] are not nearly as robust as for the generalized or polynomial eigenvalue problem.

Generalized bisection with the degree provides an alternate technique for finding start points by searching for roots of Eq. (27) in a region of s. Furthermore, if the control-system equations are analytic in s, the dynamic matrix is an analytic function of s, and the topological degree with respect to s is the winding number, the number of roots in the region minus the number of poles.

It is not necessary to use the Picard extension to ensure that all roots are accounted for, if the dynamic matrix is analytic in s, because the Jacobian determinant is always positive, in contrast to the $[V, \omega]_{\sigma=0}$ regions discussed previously. Substituting the Cauchy–Riemann equations [Eq. (5)] into the expression for the Jacobian matrix [Eq. (4)]

$$\boldsymbol{J}_f = \begin{bmatrix} \frac{\partial f_1}{\partial \sigma} & \frac{\partial f_1}{\partial \omega} \\ \frac{\partial f_2}{\partial \sigma} & \frac{\partial f_2}{\partial \omega} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial \sigma} & \frac{\partial f_1}{\partial \omega} \\ -\frac{\partial f_1}{\partial \omega} & \frac{\partial f_1}{\partial \sigma} \end{bmatrix} \tag{28}$$

and the determinant

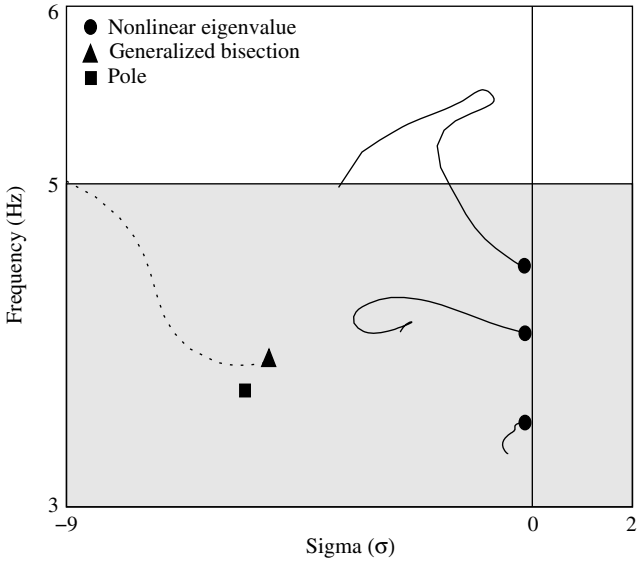

Fig. 8 Start points in the presence of a pole.

$$\Delta \boldsymbol{J}_f = \left(\frac{\partial f_1}{\partial \sigma}\right)^2 + \left(\frac{\partial f_1}{\partial \omega}\right)^2 \geq 0 \tag{29}$$

is always positive.

When using generalized bisection with the degree for locating the roots of the nonlinear eigenvalue problem, it is necessary to know the location of all the poles, and add the number of them in the region being considered to the degree to get the correct number of roots. On the other hand, generalized bisection can locate roots that may be missed with more traditional methods, particularly in the presence of poles, as the following example shows. Figure 8 shows the results of computing three start points with a nonlinear eigenvalue-solution technique (circles) for the preceding 190 DOF example. Also shown are a root missed by the nonlinear solution technique (triangle) and a control-system pole (square). Applying Algorithm 1 to the shaded region

$$P_2 = \left\{ \begin{matrix} \sigma \\ \omega \end{matrix} \right\}_{V=0} = \left\{ \begin{matrix} -9{:}2 \\ 3{:}5 \end{matrix} \right\}_{V=0} \tag{30}$$

gives a degree of 3, which seems to agree with the number of roots from the nonlinear eigenvalue solution. However, when the pole is accounted for by adding to the degree, there are four roots in the region. A generalized-bisection procedure (Algorithm 3) isolates the missing root (triangle). The proximity of this pole to the start point missed by the nonlinear eigenvalue solution is probably the reason for the failure, and for the strange behavior discussed in the previous example.

VII. Computational Aspects

Determinants are often very large or very small numbers, and can easily overflow or underflow in practice, that is, the determinant may be so large it may not be representable in computer arithmetic. For example, the determinants of the 190th-order dynamic matrix in the preceding example have orders of magnitude 10^{840}, far exceeding the common double-precision range of $10^{\pm 308}$ [21]. For this reason, the approach taken here is to compute the determinant as two numbers in the form $t \times 10^{\phi}$, in which t ranges between 1 and 10 in absolute value, and ϕ is an integer. Because the degree only depends on the signs of the components of the function [see Eq. (10)], it suffices to use only t, ignoring the factor 10^{ϕ}, and avoiding overflow or underflow.

To compute the components t and ϕ of the determinant of the complex dynamic matrix, first factor it into lower $\boldsymbol{L}$ and upper $\boldsymbol{U}$ triangular factors using, for example, LAPACK subroutine ZGETRF ([22] p. 241) which returns the triangular factors in $\boldsymbol{D}$, except for the diagonals of the lower triangle, which are implicitly all ones. Using the fact that the determinant of a triangular matrix is the product of the diagonals, and the lower factor has unit diagonals

```
complex*16 D(n,n)
integer ipiv(n)
call zgetrf(n,n,D,n,ipiv)
```

$$\Delta \boldsymbol{D} = \Delta(\boldsymbol{LU}) = \Delta \boldsymbol{L} \Delta \boldsymbol{U} = \Delta \boldsymbol{U} = \prod_{i=1}^{n_s} U_{ii} \tag{31}$$

t and ϕ are computed as shown in the following pseudo-code, accounting for the fact that pivoting changes the sign of the determinant:

```
complex t
integer phi
t = 1.0
phi = 0
do i = 1, n
  if [ipiv(i) ≠ i] t = -t
  t = D(i,i)*t
  do while [abs(t) < 1.0]
    t = 10.0*t
    phi = phi - 1
  enddo
  do while [abs(t) > 10.0]
    t = t/10.0
    phi = phi + 1
  enddo
enddo
```

The Picard technique for ensuring that all roots are identified [Eq. (16)] increases the order of the system of equations from 2 to 3, and dramatically increases the amount of work required. The example shown in Fig. 3 requires 36 evaluations of the determinant of the (n_s, n_s) dynamic matrix. Extending to three dimensions (shown in Fig. 4) increases the number of function evaluation points to 117. In addition, the derivatives of the dynamic-matrix determinant must be computed, requiring the solution of $\boldsymbol{D}^{-1}(\partial \boldsymbol{D}/\partial x_1)$ and $\boldsymbol{D}^{-1}(\partial \boldsymbol{D}/\partial x_2)$. However, the dynamic matrix was factorized to compute the determinant, and so all that is needed is two back-substitutions, one for the partial with respect to x_1 and one for the partial with respect to x_2, using, for example, the LAPACK routine ZGETRS ([22] p. 242). The matrix factorization is the most expensive portion of the work at each of the 117 points, roughly equivalent to tracking one aeroelastic mode through a reasonable velocity range with a continuation method, in which the major effort is the factorization of a $(2n_s, 2n_s + 1)$ real Jacobian. Thus, even the 3-D technique is cheap insurance against the potentially catastrophic consequences of missing an important mode.

VIII. Conclusions

Topological degree, a mathematical technique for determining if a system of equations has roots within a given region of the independent variables, solves some important problems in flutter analysis. Applied to the linear flutter equation, it provides a method for ensuring that important roots have not been missed. Roots to the flutter equation include, but are not limited to, flutter crossings (points of neutral stability) and closed-loop control-system start points.

A modification of the equations known as the Picard extension is necessary when searching for flutter crossings, because stable crossings and unstable crossings cancel in the count of crossings. This extension adds considerably to the computational effort, but the effort is small compared with the overall expense of flutter calculations, and is justified, considering the potentially severe consequences of missing important flutter modes.

References

[1] Meyer, E. E., "Unified Approach to Flutter Equations," *AIAA Journal*, Vol. 52, No. 3, 2014, pp. 627–633. doi:10.2514/1.J052554

[2] Hassig, H. J., "An Approximate True Damping Solution of the Flutter Equation by Determinant Iteration," *Journal of Aircraft*, Vol. 8, No. 11, 1971, pp. 885–889. doi:10.2514/3.44311

[3] Chen, P. C., "Damping Perturbation Method for Flutter Solution: The g-Method," *AIAA Journal*, Vol. 38, No. 9, 2000, pp. 1519–1524. doi:10.2514/2.1171

[4] Edwards, J. W., and Wieseman, C. D., "Flutter and Divergence Analysis Using the Generalized Aeroelastic Analysis Method," *Journal of Aircraft*, Vol. 45, No. 3, 2008, pp. 906–915. doi:10.2514/1.30078

[5] Ruhe, A., "Algorithms for the Nonlinear Eigenvalue Problem," *SIAM Journal on Numerical Analysis*, Vol. 10, No. 4, 1973, pp. 674–689. doi:10.1137/0710059

[6] Fečkan, M., *Topological Degree Approach to Bifurcation Problems*, Springer, New York, 2008, pp. 2–5.

[7] Amster, P., *Topological Methods in the Study of Boundary Value Problems*, Springer, New York, 2014, pp. 145–184. doi:10.1007/978-1-4614-8893-4

[8] Chang, K.-C., *Methods in Nonlinear Analysis*, Springer, New York, 2005, pp. 155–163.

[9] Ortega, R., "Topological Degree and Stability of Periodic Solutions for Certain Differential Equations," *Journal of the London Mathematical Society*, Vol. s2-42, No. 3, 1990, pp. 505–516. doi:10.1112/jlms/s2-42.3.505

[10] Churchill, R. V., Brown, J. W., and Verhey, R. F., *Complex Variables and Applications*, 3rd ed., McGraw–Hill, New York, 1974.

[11] Hoadley, S. T., and Karpel, M., "Application of Aeroservoelastic Modeling Using Minimum-State Unsteady Aerodynamic Approximations," *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 6, 1991, pp. 1267–1276. doi:10.2514/3.20783

[12] Ortega, J. M., and Rheinboldt, W. C., *Iterative Solutions of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970, p. 148.

[13] Fonseca, I., and Gangbo, W., *Degree Theory in Analysis and Applications*, Oxford Univ. Press, New York, 1995, p. 41.

[14] Mourrain, B., Vrahatis, M. N., and Yakoubsohn, J. C., "On the Complexity of Isolating Real Roots and Computing with Certainty the Topological Degree," *Journal of Complexity*, Vol. 18, No. 2, 2002, pp. 612–640. doi:10.1006/jcom.2001.0636

[15] Stenger, F., "Computing the Topological Degree of a Mapping in R^n," *Numerische Mathematik*, Vol. 25, No. 1, 1975, pp. 23–38. doi:10.1007/BF01419526

[16] *MSC/NASTRAN Aeroelastic Analysis User's Guide: MSC/NASTRAN Version 70*, Vol. 1, MacNeal–Schwendler Corp., Santa Ana, CA, Sept. 2009, pp. 357–372.

[17] Bisplinghoff, R. L., Ashley, H., and Halfman, R. L., *Aeroelasticity*, Addison Wesley, Reading, MA, 1955, p. 565.

[18] Picard, E., "Sur le nombre des racines communes à plusieurs équations simultanées," *Journal de Mathématiques Pures et Appliquées*, Vol. 4, No. 8, 1892, pp. 5–24.

[19] Hoenders, B. J., and Slump, C. H., "On the Calculation of the Exact Numbers of Zeroes of a Set of Equations," *Computing*, Vol. 30, No. 2, 1983, pp. 137–147. doi:10.1007/BF02280784

[20] Schott, J. R., *Matrix Analysis for Statistics*, 2nd ed., Wiley, Hoboken, NJ, 2005, p. 360.

[21] Goldberg, D., "What Every Computer Scientist Should Know About Floating-Point Arithmetic," *ACM Computing Surveys*, Vol. 23, No. 1, 1991, pp. 5–48. doi:10.1145/103162.103163

[22] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D., *LAPACK Users' Guide*, 3rd ed., Soc. for Industrial and Applied Mathematics, Philadelphia, 1999.

B. Epureanu
Associate Editor