Unified Approach to Flutter Equations

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

DOI: 10.2514/1.J052554

A general formulation of the flutter equation includes gyroscopics, viscous damping, active controls, Mach and complex reduced-frequency-based unsteady aerodynamics, quasi-linear approximated structural nonlinearities, and parameterized matrices. Solving the flutter equation usually involves a search for neutral stability points, the boundary between stability and instability, followed by variations in the neutral stability point with various parameters. A technique for solving the flutter equation under these conditions must necessarily be capable of solving systems of nonlinear parameterized equations over a range of parameter values. Continuation methods are specifically designed for solving such problems. A continuation method is presented that is capable of solving general formulations of the flutter equation for neutral stability, parameter variations, optimization, model tuning, and describing function flutter analyses, including an efficient technique for assessing limit cycle stability.

Nomenclature

$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}_c$	=	control system matrices
a	=	sonic velocity
$\boldsymbol{D}$	=	dynamic matrix
d	=	structural damping coefficient
$\boldsymbol{J}$	=	Jacobian matrix
$\boldsymbol{J}_{:j}$	=	column j of matrix $\boldsymbol{J}$
$\boldsymbol{M}, \boldsymbol{K}, \boldsymbol{G}$	=	mass, stiffness, gyroscopic matrices
M	=	V/a, Mach number
n_s	=	order of the dynamic matrix
p	=	s/V, complex reduced frequency
$\|\boldsymbol{x}\|$	=	2-norm of vector $\boldsymbol{x} = \sqrt{\boldsymbol{x}^t\boldsymbol{x}}$
q	=	$\rho V^2/2$, dynamic pressure
$\boldsymbol{q}$	=	generalized coordinate amplitudes
$\mathbb{R}^n$, $\mathbb{R}^{m\times n}$	=	the real n vectors and m-by-n matrices
$\Re()$, $\Im()$	=	real and imaginary parts of a complex variable
s	=	$\sigma + i\omega$, characteristic exponent
$\boldsymbol{t}$	=	tangent vector
V	=	velocity (true airspeed)
$\boldsymbol{V}, \boldsymbol{U}, \boldsymbol{T}$	=	viscous damping, unsteady aero, and control-system matrices
$\boldsymbol{w}$	=	projection vector
$\boldsymbol{x}$	=	independent variables
$\boldsymbol{x}_j^i$	=	ith iteration at the jth continuation step
x_i	=	ith element of vector $\boldsymbol{x}$
$\boldsymbol{y}$	=	residual vector
ρ	=	fluid density
$\boldsymbol{\rho}(z)$	=	real vector comprising the real and imaginary parts of z
σ	=	real part of the characteristic exponent
$\boldsymbol{\Phi}$	=	transformation from generalized to physical coordinates
ω	=	oscillation frequency

Superscript

t	=	transpose

I. Introduction

THROUGHOUT the design of a modern aircraft, many calculations are performed to ensure that the aircraft will be free of flutter. In the preliminary stages of design, various combinations of design parameters are considered. As the design becomes fixed, flutter calculations are required at various combinations of fuel, payload, altitude, and other parameters that define flight conditions. The flutter analyst is therefore concerned with two basic types of calculations: a search for the lowest (critical) flutter speed under a given set of conditions, and the variation of the critical flutter speed with various system parameters. Flutter equations commonly encountered in modern structures contain terms that make them difficult or impossible to treat with traditional approaches [1,2]. A partial list includes active controls, structural nonlinearities, Mach dependent aerodynamics, and gyroscopics. By treating the flutter equation as a system of nonlinear algebraic equations, it is possible to account for all of these terms and solve both neutral-stability and parameter-variation problems in a unified manner. Various types of parameter variations are possible; curves and contours are considered here, and surfaces are possible but not treated here.

Continuation methods are a well known class of techniques for the solution of systems of underdetermined nonlinear equations, maintaining continuity over a specified range of the parameters, making them ideally suited to solving flutter equations. Continuation methods have been used previously [3,4] to solve flutter equations. Here, a more general method is presented that allows for optimization and model tuning solutions in addition to neutral stability and parameter variations.

II. Flutter Equations

The flutter equation considered here is a system of characteristic equations following the assumption that the actual motion of the structure (z) as a function of time is related to a set of generalized coordinates ($\boldsymbol{q}$) by

$$z(t) = \boldsymbol{\Phi}\Re(\boldsymbol{q}e^{st}) \tag{1}$$

where $s = \sigma + i\omega$ is the complex characteristic exponent, ω is the frequency, $i^2 = -1$, and the real part σ determines the rate of growth or decay of the oscillations and hence is of primary interest to the flutter analyst. $\boldsymbol{\Phi}$ is a transformation from the generalized coordinates to physical degrees of freedom, often a set of vibration modes, and $\Re$ denotes the real part.

With this assumption, a general form of the flutter characteristic equations can be written as

$$[s^2\boldsymbol{M} + s\boldsymbol{G} + s\boldsymbol{V} + (1+id)\boldsymbol{K} - q\boldsymbol{U}(p, M) + \boldsymbol{T}]\boldsymbol{q} = \boldsymbol{D}\boldsymbol{q} = \boldsymbol{0} \tag{2}$$

where $\boldsymbol{M}$, $\boldsymbol{K}$, $\boldsymbol{G}$, $\boldsymbol{V}$, $\boldsymbol{U}$, and $\boldsymbol{T}$ are the (n_s-by-n_s) mass, stiffness, gyroscopic, viscous damping, unsteady aerodynamic, and control-system matrices, respectively; d is the structural damping coefficient; $q = \rho V^2/2$ is the dynamic pressure; $p = s/V$ is the complex

Received 11 January 2013; revision received 29 June 2013; accepted for publication 6 August 2013; published online 31 January 2014.

*Consultant.

reduced frequency; $M = V/a$ is the freestream Mach number; a is the sonic velocity; and $\boldsymbol{D}$ is the dynamic matrix.

The matrices in this equation are, in general, not constant; rather, they are assumed to be functions of problem-dependent parameters using various matrix parameterizations. Nonlinear structural elements such as freeplay and nonlinear stiffness can be approximated using describing functions ([5] p. 121), which conform to the harmonic motion assumption [Eq. (1)].

Treating the flutter equation in the frequency domain has tremendous computational advantages over the time domain: the characteristic equation is a system of nonlinear algebraic equations instead of a system of differential equations that must be integrated at each flight condition. In contrast, the algebraic equations can be solved over a continuous range of flight conditions. The disadvantage is that nonlinearities must be approximated to conform to the harmonic motion assumption [Eq. (1)].

III. Parameterized Matrices

Various possibilities exist for creating matrices that are functions of parameters, either those explicit in Eq. (2) or new problem-dependent parameters. A few are discussed here, and others are easily imagined. In the continuation method that follows, a requirement for any type of matrix parameterization is the ability to compute derivatives of the matrix with respect to its parameters. In addition, the matrix elements should be continuous with respect to the parameters.

A simple parameterization is to set an element of the matrix to a function. For example, nonlinear structural elements can be approximated using the describing function method ([5] p. 121) by scaling a diagonal element of the stiffness matrix by a function of the absolute value of the corresponding generalized coordinate. Structural elements with freeplay can be modeled [6] by multiplying the diagonal of the stiffness matrix corresponding to the relevant generalized coordinate by

$$c(q_k, \delta) = \begin{cases} \frac{1}{\pi}[\pi - 2\sin^{-1}(u) - \sin(2\sin^{-1}(u))] & \text{if } |q_k| \geq \delta \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$u = \delta/|q_k| = \delta/\sqrt{\Re(q_k)^2 + \Im(q_k)^2} \tag{4}$$

where δ is the freeplay, and q_k is the generalized coordinate associated with the nonlinear element. Note that this involves three parameters: δ, and the real and imaginary parts of q_k. The flutter equations are no longer linear in the generalized coordinates, but the motion is constrained by the harmonic assumption [Eq. (1)], which is why this is often called a quasi-linear approximation. Similar describing functions can approximate other types of structural nonlinearities such as bilinear stiffness or quadratic damping. The use of describing functions to approximate nonstructural nonlinearities like control systems and unsteady aerodynamics needs further investigation.

A commonly used technique for parameterizing matrices is interpolation. Given a collection of matrices at various values of one or more parameters numerous techniques are available for fitting each element with, for example, cubic splines. Unsteady aerodynamic matrices are often interpolated with respect to reduced frequency (ω/V), complex reduced frequency ($p = s/V$), or complex reduced frequency and Mach number. The mass matrix can be parameterized by creating matrices at various mass distributions and interpolated with respect to the relevant parameter, similarly for the stiffness matrix.

A powerful way to parameterize a matrix is to use a custom computer code to generate or modify an existing matrix based on various parameters. A matrix modified in this manner can be an almost arbitrary function of parameters. The problem with this kind of parameterization is computing derivatives of the matrix with respect to its parameters. One solution is to write computer code to evaluate the derivatives, possibly an onerous task. Alternatively, finite-difference approximations could be used, eliminating the effort to code derivatives but introducing a possibly unacceptable approximation. A better solution, one that requires no extra coding yet gives the exact derivatives, is to write the code or transform it to use automatic differentiation ([7] p. 15). The control-system $\boldsymbol{T}$ is an example of a matrix that is well suited to implementing in a custom computer code as it is often a complicated function of several parameters such as gains, phases, and time delays. The control-system matrix usually has more rows and columns than the mass matrix. The extra rows and columns correspond to control-system degrees of freedom. $\boldsymbol{T}$ typically has the following form:

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{T}_{11} & \boldsymbol{T}_{12} \\ \boldsymbol{T}_{21} & \boldsymbol{T}_{22} \end{bmatrix} \tag{5}$$

where $\boldsymbol{T}_{11}$ is the size of the mass matrix, $\boldsymbol{T}_{21}$ transforms generalized coordinates to physical sensor motions, $\boldsymbol{T}_{22}$ contains control-system equations, and $\boldsymbol{T}_{12}$ provides feedback to the structure from the control-system degrees of freedom. The extra control-system degrees of freedom must be accounted for when forming the dynamic matrix.

If the controls equations are available in linear, time-invariant state variable form ([8] p. 172):

$$\dot{\boldsymbol{x}}_c = \boldsymbol{A}\boldsymbol{x}_c + \boldsymbol{B}\boldsymbol{u} \in \mathbb{R}^{n_c}$$
$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}_c + \boldsymbol{D}_c\boldsymbol{u} \in \mathbb{R}^{n_o}$$

where $\boldsymbol{u}$ is an n_i vector of inputs, $\boldsymbol{y}$ is an n_o vector of outputs, $\boldsymbol{x}_c$ is an n_c vector of state variables, and $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}_c$ are real matrices, then it can be shown that

$$\boldsymbol{T}_{11} = -\boldsymbol{K}\boldsymbol{E}\boldsymbol{D}_c\boldsymbol{S}\boldsymbol{\Phi}_c$$
$$\boldsymbol{T}_{12} = -\boldsymbol{K}\boldsymbol{E}\boldsymbol{C}$$
$$\boldsymbol{T}_{21} = \boldsymbol{B}\boldsymbol{S}\boldsymbol{\Phi}_c$$
$$\boldsymbol{T}_{22} = \boldsymbol{A} - s\boldsymbol{I}$$

where $\boldsymbol{\Phi}_c$ are the rows of the transformation matrix corresponding to the control system inputs; and $\boldsymbol{S}$ is a diagonal matrix consisting of either 1, s, or s^2, depending on whether the corresponding input is a displacement, velocity, or acceleration. $\boldsymbol{E}$ is a real matrix that transforms the control system output $\boldsymbol{y}$ to generalized coordinates.

IV. Neutral Stability

The interest in the flutter equation is primarily in the behavior of the real part of the characteristic exponent (s) with velocity; negative values indicate decaying oscillations, positive values indicate growing oscillations, and zero indicates neutral stability. From Eq. (2), n_s physically meaningful aeroelastic modes can be calculated at each set of system parameters. Normally, the flutter analyst is interested in only a few of these modes.

Continuity in these aeroelastic modes is important if the neutral-stability point falls between two solution points and interpolation must be used to get the neutral-stability point. In addition, continuity is necessary in understanding the evolution of aeroelastic modes as the velocity is increased.

As an example of the results from such an analysis, Fig. 1 shows σ plotted against velocity for five aeroelastic modes, only one of which goes unstable (point a). This and all subsequent examples considered here are results from analyses of a simple cantilevered wing with five vibration modes used for the generalized coordinate basis.

V. Parameter Variations

Once the lowest neutral-stability velocity (critical flutter speed) has been found, interest usually turns to studying the variation in critical flutter speed with various model parameters such as mass and stiffness distributions, altitude, and control-system gain. Two possible parameter variations are shown in Figs. 2 and 3. Figure 2 shows the variation in critical flutter speed with a stiffness parameter

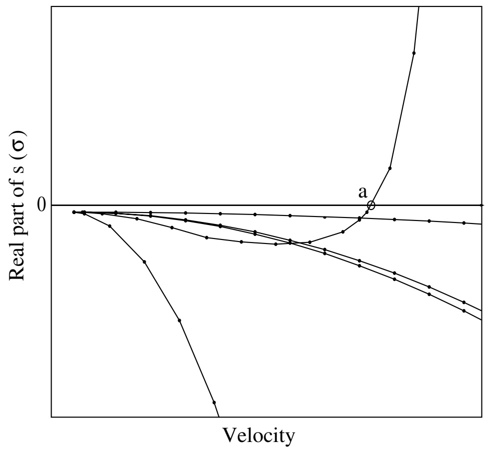

Fig. 1 Typical neutral-stability plot.

p_1, and Fig. 3 is a contour of constant flutter speed with p_1 and a mass parameter p_2.

With parameter variations, it is even more important to track modes in a continuous fashion. The alternative, to do a neutral-stability analysis at each incremental value of the parameters would be prohibitively expensive. Moreover, the character of aeroelastic modes can change rapidly, making it difficult to relate modes at different steps, which makes it necessary to do a neutral-stability analysis tracking more than just the critical flutter mode at the previous step.

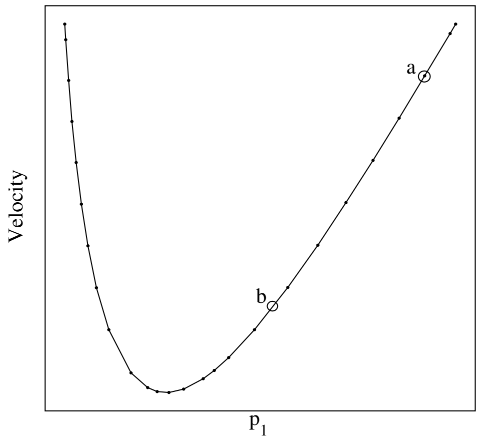

Fig. 2 Flutter speed as a function of p_1.

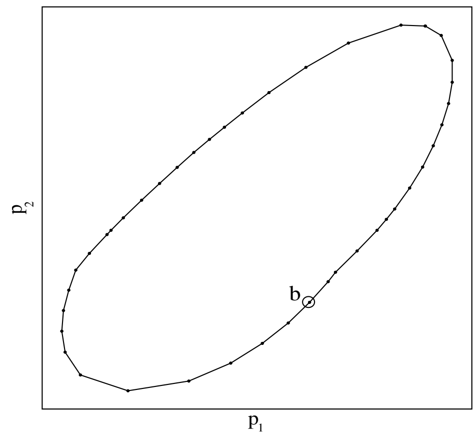

Fig. 3 Constant flutter speed as a function of p_1 and p_2.

VI. Continuation Method

Continuation methods ([9] p. 13, [10,11]) are a class of techniques for solving underdetermined systems of nonlinear equations:

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

$$\boldsymbol{x} \in \mathbb{R}^n \tag{7}$$

where the number of equations m is fewer than the number of independent variables n. Because there are fewer equations than unknowns, there are an infinite number of solutions. Depending on the difference $n - m$, the solution space consists of curves ($n - m = 1$), surfaces ($n - m = 2$), and so on.

Continuation methods solve the system of equations so that solutions form continuous curves, an important consideration for the solution of flutter equations where aeroelastic modes often become close, making them difficult to keep separated.

There are a variety of continuation methods, most of which are predictor–corrector schemes reminiscent of ordinary differential equation solvers. Many methods ([3], [9] p. 3) choose a particular parameter to increment throughout the tracing of curves, but these methods can fail if the curve has limit points in the continuation parameter. Locally parameterized continuation methods [11] avoid this problem but require choosing the best parameter at each step. These methods constrain the corrector steps to be at constant values of the continuation parameter. In contrast, pseudoarclength methods take steps along the tangent to the curve and constrain the corrector steps to be perpendicular to the tangent, thus avoiding the problems with continuation parameters. The method presented here is a pseudoarclength method with minimum-norm corrector steps; the corrector steps are computed in a way that gives the smallest step instead of being constrained to be normal to the tangent.

Fundamental to continuation methods is the Jacobian matrix of partial derivatives of $\boldsymbol{f}$ with respect to $\boldsymbol{x}$:

$$\boldsymbol{J}(\boldsymbol{x}) = \boldsymbol{f}'(\boldsymbol{x}) = \left[\frac{\partial f_i}{\partial x_j}\right] \in \mathbb{R}^{m\times n} \tag{8}$$

and a tangent vector $\boldsymbol{t} \in \mathbb{R}^n$ with the following properties:

$$\boldsymbol{J}\boldsymbol{t} = \boldsymbol{0} \quad \boldsymbol{t}^t\boldsymbol{t} = 1 \tag{9}$$

Starting from a known solution $\boldsymbol{x}_0$, a curve is traced through a range of the parameters. At step $j + 1$, the solution is predicted from the converged solution at step j and the tangent

$$\boldsymbol{x}^0_{j+1} = \boldsymbol{x}_j + \alpha\boldsymbol{t} \tag{10}$$

where α is the step size, which must be chosen carefully. Too large a step, and the corrector may not converge or may converge to the wrong root; too small a step means extra work. Den Heijer and Rheinboldt [12] studied the problem of computing step sizes, and Rheinboldt [11] produced a continuation code using their algorithm, taking into account the curvature and the rate and quality of corrector convergence.

Starting with the prediction $\boldsymbol{x}^0_{j+1}$, the corrector iterates to find the solution $\boldsymbol{x}_{j+1}$ using a Newton-like method computing corrections $\boldsymbol{h}^i$ satisfying

$$\boldsymbol{J}(\boldsymbol{x}^i)\boldsymbol{h}^i = -\boldsymbol{f}(\boldsymbol{x}^i) \tag{11}$$

and setting

$$\boldsymbol{x}^{i+1} = \boldsymbol{x}^i + \boldsymbol{h}^i \tag{12}$$

Equation (11) is an underdetermined linear system that has an infinity of solutions; the smallest such solution can be computed using a QR factorization of the transposed of Jacobian which splits it into an orthogonal matrix ($\boldsymbol{Q}$) times an upper-triangular matrix $\boldsymbol{R}$:

$$\boldsymbol{J}^t = \boldsymbol{QR} = [\,\boldsymbol{Q}_1 \quad \boldsymbol{Q}_2\,]\begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{0} \end{bmatrix} \tag{13}$$

which when substituted into Eq. (11) yields the minimum-norm solution ([13] p. 300)

$$\boldsymbol{h}^i = -\boldsymbol{Q}_1 \boldsymbol{R}_1^{-t} \boldsymbol{f}(\boldsymbol{x}^i) \tag{14}$$

Corrector iterations continue using Eqs. (11–14) until suitable convergence criteria are met, for example for some absolute tolerance ε_a and relative tolerance ε_r:

$$\|\boldsymbol{f}\| < \epsilon_a \quad \text{and} \quad \|\boldsymbol{h}\| < \epsilon_a + \epsilon_r \|\boldsymbol{x}\| \tag{15}$$

The $(n, n-m)$ matrix $\boldsymbol{Q}_2$ in Eq. (13), a byproduct of the QR factorization, is an orthogonal matrix spanning the null space of the Jacobian, meaning

$$\boldsymbol{J}\boldsymbol{Q}_2 = \boldsymbol{0} \in \mathbb{R}^{m \times n-m}$$
$$\boldsymbol{Q}_2^t \boldsymbol{Q}_2 = \boldsymbol{I} \in \mathbb{R}^{n-m \times n-m} \tag{16}$$

An arbitrary vector $\boldsymbol{w}$ is projected onto the null space with

$$\bar{\boldsymbol{t}} = \boldsymbol{Q}_2 \boldsymbol{Q}_2^t \boldsymbol{w} \in \mathbb{R}^n \tag{17}$$

and normalized with $\boldsymbol{t} = \bar{\boldsymbol{t}}/\|\bar{\boldsymbol{t}}\|$ to satisfy Eq. (9). If $n = m + 1$, $\boldsymbol{Q}_2$ consists of only one column, the projection only determines the sign of the tangent, and a logical choice for the vector to project is the tangent at the previous continuation step to keep the curve tracking in the right direction. For the first step, any vector with the desired sign will do.

VII. Flutter Equations as a System of Nonlinear Real Equations

The complex flutter equation can be converted to an equivalent system of real equations of twice the size, opening up the vast amount of effort that has been put in to developing real continuation methods ([9] p. 146). Many of the parameters in the flutter equation are real, and no comparable theory has been developed for complex systems dependent on real variables. The drawback to this conversion is a penalty in space (twice the computer memory) and time to factorize the Jacobian (again a factor of 2).

The flutter equation in a form suitable for solution using a continuation method is a system of $m = 2n_s$ real equations comprising the real and imaginary parts of the residual vector $\boldsymbol{y} = \boldsymbol{Dq}$:

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{\rho}(\boldsymbol{y}) \in \mathbb{R}^m \tag{18}$$

where $\boldsymbol{\rho}$ is a notational convenience that creates a real vector of length $2k$ from a complex vector of length k:

$$\boldsymbol{\rho}(\boldsymbol{z}) = [\,\Re(z_1) \quad \Im(z_1) \ldots \Re(z_k) \quad \Im(z_k)\,]^t \in \mathbb{R}^{2k}$$

The vector of independent variables $\boldsymbol{x} \in \mathbb{R}^n$ comprises the real and imaginary parts of elements of the generalized-coordinate amplitudes ($\boldsymbol{q}$), flutter equation parameters such as V, ω, σ, and problem-dependent parameters. Depending on which parameters are included in $\boldsymbol{x}$, a variety of solutions are possible. Three examples are shown here, all of which have one more independent variable than equations resulting in curves. Each of these examples and some subsequent examples include a reduced set of generalized coordinates:

$$\boldsymbol{q}_r = [\boldsymbol{\rho}(q_1)^t \ldots \boldsymbol{\rho}(q_{k-1})^t \quad \boldsymbol{\rho}(q_{k+1})^t \ldots \boldsymbol{\rho}(q_{n_s})^t\,]^t \in \mathbb{R}^{m-2}$$

where $\boldsymbol{q}_r$ is a real vector comprising the real and imaginary parts of the generalized coordinates, excluding the kth element, which is held constant for two reasons: to normalize the generalized coordinates, and to prevent convergence to the trivial solution $\boldsymbol{q} = \boldsymbol{0}$. The index k is usually chosen as the largest generalized coordinate component. With these $m - 2$ components, three more parameters are needed to trace curves. In addition to the three parameters, it is usually necessary to set other parameters to constants. In the following examples, density ρ, speed of sound a, and the structural damping coefficient d are held constant.

The first example is for a neutral stability analysis as in Fig. 1:

$$\boldsymbol{x} = \begin{Bmatrix} V \\ \sigma \\ \omega \\ \boldsymbol{q}_r \end{Bmatrix} \in \mathbb{R}^{m+1} \tag{19}$$

The variation in flutter speed with a parameter (p_1) can be studied by setting $\sigma = 0$ and

$$\boldsymbol{x} = \begin{Bmatrix} V \\ \omega \\ p_1 \\ \boldsymbol{q}_r \end{Bmatrix} \in \mathbb{R}^{m+1} \tag{20}$$

resulting in the parameter-variation analysis as shown in Fig. 2. Another kind of parameter variation is possible with V held constant, $\sigma = 0$, and

$$\boldsymbol{x} = \begin{Bmatrix} \omega \\ p_1 \\ p_2 \\ \boldsymbol{q}_r \end{Bmatrix} \in \mathbb{R}^{m+1} \tag{21}$$

resulting in a constant-velocity contour, as shown in Fig. 3. Many more types of analyses are possible, provided the parameters included in $\boldsymbol{x}$ and those held constant form a consistent set of equations.

An important consequence to including the generalized coordinates in the vector of independent variables is that the continuation method is much more likely to maintain continuity when tracking curves. The generalized coordinates participate in both the predictions and the corrections equal to any other parameters. A step-size algorithm that takes into account the curvature of the curve being tracked will therefore adjust the step size based on changes in all parameters including the generalized coordinates. It may seem computationally more expensive to include the generalized coordinates in the prediction and correction; that this is not the case can be seen by considering the operation count ([13] p. 299) for the most widely used solution techniques. For the traditional V-g technique, the major expense is an eigensolution of an (n_s, n_s) complex matrix [2]; for the determinant iteration [1], it is the factorization of an (n_s, n_s) complex matrix; and for the continuation method, it is the factorization of the $(2n_s, 2n_s + 1)$ real Jacobian matrix. Each of these require some multiple of n_s^3 operations, but because it uses the generalized coordinates, only the continuation method can reliably maintain continuity in the solution curves.

A. Jacobian Matrix

The Jacobian matrix needed for the continuation method serves two purposes: iterating to find a solution at each continuation step using Newton's method, and computing the tangent vector for the prediction of the next step. It is therefore important to compute the Jacobian reasonably accurately both to ensure continuity and to speed convergence.

The Jacobian is an m $(= 2n_s)$-by-n real matrix, much of which is the dynamic matrix due to the partial of the residual with respect to the generalized coordinates. Column j of the Jacobian matrix is

$$\boldsymbol{J}_{:j} = \boldsymbol{\rho}(\boldsymbol{y}^j)$$
$$\boldsymbol{y}^j = \frac{\partial \boldsymbol{y}}{\partial x_j} = \frac{\partial \boldsymbol{D}}{\partial x_j}\boldsymbol{q} + \boldsymbol{D}\frac{\partial \boldsymbol{q}}{\partial x_j} \tag{22}$$

The second term on the right-hand side is zero unless x_j is the real or imaginary part of a generalized coordinate. If it is the real part of generalized coordinate k, then the term is the kth column of $\boldsymbol{D}$, and if it is the imaginary part, then it is i times the kth column of $\boldsymbol{D}$, where $i^2 = -1$.

For the set of variables in the example of Eq. (19),

$$\begin{aligned} \boldsymbol{y}^1 &= \frac{\partial \boldsymbol{y}}{\partial V} = \left[\rho V \boldsymbol{U} + \frac{q}{a}\frac{\partial \boldsymbol{U}}{\partial M} - \frac{qs}{V^2}\frac{\partial \boldsymbol{U}}{\partial p}\right]\boldsymbol{q} \\ \boldsymbol{y}^2 &= \frac{\partial \boldsymbol{y}}{\partial \sigma} = \left[2s\boldsymbol{M} + \boldsymbol{G} + \boldsymbol{V} + \frac{q}{V}\frac{\partial \boldsymbol{U}}{\partial p}\right]\boldsymbol{q} \\ \boldsymbol{y}^3 &= \frac{\partial \boldsymbol{y}}{\partial \omega} = \left[2isM + i(\boldsymbol{G} + \boldsymbol{V}) + \frac{iq}{V}\frac{\partial \boldsymbol{U}}{\partial p}\right]\boldsymbol{q} \end{aligned} \tag{23}$$

B. Start Points

Finding start points for parameter-variation analyses is straightforward; they are taken from previous neutral stability or parameter-variation analyses. For example, the start point for the curve in Fig. 2 is the neutral-stability point in Fig. 1 (point a). It may be necessary to interpolate results from the previous analysis to obtain the precise start point, for example if the neutral-stability point ($\sigma = 0$) lies between two solution points.

Start points for parameter-variation analyses can also be obtained from previous parameter-variation analyses. For example, a start point for the flutter speed contour of Fig. 3 could be obtained from the analysis of Fig. 2 by interpolating to get the desired flutter speed (point b).

Start points for neutral-stability analyses are more problematic. Generally, the task is to solve Eq. (2) at zero (or small) velocity, giving rise to the following nonlinear eigenvalue problem:

$$\boldsymbol{D}(s)\boldsymbol{q} = \boldsymbol{0} \tag{24}$$

Solutions to Eq. (24) are eigenpairs $(s_i, \boldsymbol{q}_i)$ that can be used to form start points using Eq. (19). A straightforward technique due to Ruhe [14] solves Eq. (24) with a series of linear, generalized eigenvalue problems starting with an initial guess s^0 for the first eigenvalue.

1) Solve the complex generalized eigenvalue problem

$$\boldsymbol{D}(s^j)\boldsymbol{q} = -\lambda \frac{\partial \boldsymbol{D}}{\partial s}(s^j)\boldsymbol{q}$$

for the smallest (absolute value) λ.

2) Set $s^{j+1} = s^j + \lambda$.

3) If $\lambda > \epsilon|s^{j+1}|$, then repeat.

Then choose the second smallest λ, set $s^0 = s^{j+1} + \lambda$, and repeat the process for the next eigenvalue. This algorithm is based on a Taylor series expansion of Eq. (24), keeping only two terms. If instead three terms are retained,

$$\boldsymbol{D}(s+\lambda) \approx \boldsymbol{D}(s) + \lambda\frac{\partial \boldsymbol{D}(s)}{\partial s} + \frac{\lambda^2}{2}\frac{\partial^2 \boldsymbol{D}(s)}{\partial s^2} + \ldots \tag{25}$$

the eigenvalue problem becomes a polynomial eigenvalue problem

$$\left[\boldsymbol{D}(s^j) + \lambda\frac{\partial \boldsymbol{D}}{\partial s}(s^j) + \lambda^2\frac{1}{2}\frac{\partial^2 \boldsymbol{D}(s^j)}{\partial s^2}\right]\boldsymbol{q} = \boldsymbol{0} \tag{26}$$

that takes more effort to solve but should result in faster convergence. In fact, in the absence of control-systems, gyroscopics, and viscous damping, at zero velocity and $s^0 = 0$, Eq. (26) becomes the usual free-vibration eigenvalue problem $[\lambda^2\boldsymbol{M} + \boldsymbol{K}]\boldsymbol{q} = \boldsymbol{0}$, and all eigenpairs are valid start points with no iteration required. Note also that s^0 should be small but not zero with the first algorithm because $(\partial \boldsymbol{D}/\partial s)(0)$ is zero in this case.

VIII. Guided Tangent

Normally, to track curves with a continuation method, the number of unknowns must be exactly one greater than the number of equations; two greater yields surfaces, three yields volumes, and so on. Likewise, the tangent is a vector for curves, a plane for surfaces, and so on for higher dimensions. By restricting the tangent to being a vector in the higher-dimensional spaces, it is possible, and useful, to trace a curve. The method used to compute the tangent vector [Eq. (17)] allows for this restriction simply by choosing the projected vector $\boldsymbol{w}$.

The dimension of the Jacobian null space (columns in $\boldsymbol{Q}_2$) is $n - m$. If this difference is greater than 1, there are an infinite number of tangent vectors possible instead of just one, which means that there is more freedom to guide the tangent in particular directions by choosing the vector, $\boldsymbol{w}$ in Eq. (17), that is projected onto the null space. A few examples will illustrate the technique.

A. Optimization

If the projection vector is one of the n coordinate directions, that is if it is all zeros except for one of the elements, say the kth element, then that component of the resulting tangent will be maximal. That is, moving along this tangent will have the greatest change in x_k of all possible tangent vectors, and at each continuation step the curve will head toward maximally increasing or decreasing x_k, depending on the sign of the projection vector.

Returning to the example of Eq. (21), to optimize flutter speed with respect to the two parameters p_1 and p_2, let

$$\boldsymbol{x} = \begin{Bmatrix} V \\ \omega \\ p_1 \\ p_2 \\ \boldsymbol{q}_r \end{Bmatrix} \in \mathbb{R}^{m+2} \tag{27}$$

and if the projection vector is all zeros but for ± 1.0 in the element corresponding to velocity:

$$\boldsymbol{w} = [\pm 1 \quad 0 \ldots 0]^t \in \mathbb{R}^{m+2} \tag{28}$$

Equation (17) will drive the continuation curve toward maximally increasing or decreasing velocity (depending on the sign); thus, it is a method for optimizing the model flutter speed with respect to model parameters. Only two parameters p_1 and p_2 were used in this example, but any number greater than one could be used.

B. Contour Start Points

Figure 3 is a constant-velocity contour of two parameters p_1 and p_2. More such contours may be traced, giving an outline of the cone-shaped $p_1 - p_2 - V$ surface shown in Fig. 4. A problem arises when searching for start points for these contours in that, if a parameter

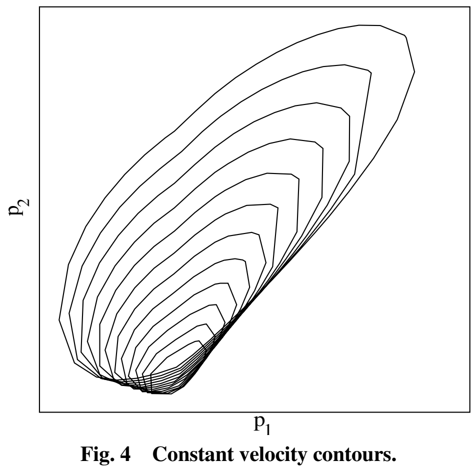

Fig. 4 Constant velocity contours.

variation as in Fig. 2 is used to find start points, the velocities may not extend the full range desired for the contours. What is needed is a curve that traverses the surface toward increasing (or decreasing) contour values: in this case, velocity. That is exactly what the optimization technique does using a set of independent variables as in Eq. (27) and a projection vector as in Eq. (28). Figure 5 shows this optimization curve with the contours and the p_1 variation curve (from Fig. 2). Clearly, the p_1 variation curve is capable of providing start points for only three contours; the guided tangent curve can provide start points for them all.

C. Tuning

The guided tangent technique can also be used to tune model parameters to force an aeroelastic mode to match certain criteria, for example to match test results. If $\hat{\boldsymbol{x}}$ is a set of parameter values, a subset of the problem parameters ($\boldsymbol{x}$), by adjusting the rest of the problem parameters it may be possible to match the subset of parameter values. The idea is to form a projection vector that is the difference between the current parameter values and the target values. For example, to match $\hat{\omega}$ and $\hat{p}_2$ in Eq. (27), the following projection vector, used to guide the tangent with Eq. (17), will tend toward the target parameter values:

$$\boldsymbol{w} = \begin{Bmatrix} 0 \\ \hat{\omega} \\ 0 \\ \hat{p}_2 \\ 0 \\ \vdots \\ 0 \end{Bmatrix} - \begin{Bmatrix} 0 \\ \omega \\ 0 \\ p_2 \\ 0 \\ \vdots \\ 0 \end{Bmatrix} \in \mathbb{R}^{m+2} \tag{29}$$

Of course, there may be no set of parameter values that will cause the mode to match the target, but this procedure will bring the values as close as possible, and intuitively, including more parameters in $\boldsymbol{x}$ would make for a closer match.

IX. Nonlinear Flutter

Nonlinearities can be approximated using describing functions that replace matrix elements with terms that are in some sense equivalent for a given amplitude of oscillation, resulting in flutter equations that are nonlinear in the generalized coordinate amplitudes. Because the flutter equations are already nonlinear in other parameters, this does not change the solution technique.

A basic assumption made with linear flutter equations is that the motion is infinitesimally small. A neutral stability condition found with linear flutter equations means that, given an infinitesimally small perturbation, the structure will oscillate with sinusoidal motion that neither decays nor grows. In contrast, a neutral stability condition found with nonlinear flutter equations represents a limit cycle oscillation (LCO) because the oscillations, although still not growing or decaying, are at a finite amplitude. More important is what happens to the limit cycle if the amplitude is changed slightly; if the limit cycle is stable, then the amplitude will return to the limit cycle amplitude, but if it is an unstable limit cycle, then the amplitude will increase or decrease to the closest stable limit cycle. Determining whether a limit cycle is stable or unstable requires the partial derivative of σ with respect to the nonlinear amplitude [6], information that is available from the Jacobian.

Approximating nonlinearities with a quasi-linear technique like describing functions complicates finding the lowest neutral-stability point by the fact that the generalized coordinates cannot be normalized arbitrarily because the flutter equation is no longer linear in the generalized coordinates. As in Eq. (19), one generalized coordinate could be left out of $\boldsymbol{x}$ and the critical flutter speed found with this one generalized coordinate amplitude held constant. Beyond this, it is also desirable to study the variation in flutter speed with nonlinear amplitude, perhaps with the goal of finding the combination of nonlinear amplitudes that gives the lowest flutter speed. This can be done by replacing σ with the real part of the excluded generalized coordinate; the imaginary part remains excluded and set to a constant value:

$$\boldsymbol{x} = \begin{Bmatrix} V \\ \omega \\ \Re(q_k) \\ \boldsymbol{q}_r \end{Bmatrix} \in \mathbb{R}^{m+1} \tag{30}$$

Tracing a curve with these variables gives the variation in flutter speed and frequency with generalized coordinate amplitudes. Figure 6 shows a typical variation in velocity with nonlinear amplitude. At one particular speed, arrows show a stable limit cycle on the upper branch and an unstable limit cycle on the lower branch. At this speed, perturbations of amplitude less than a decay to zero, but with amplitudes greater than a, the oscillations will grow to b and remain at this stable LCO.

At each solution point, it is possible to determine if the curve represents a stable or unstable limit cycle from the tangent vector of a modified Jacobian. Partial derivatives of variables on a curve can be obtained from the tangent vector simply by dividing the appropriate elements; thus, for example, if V were replaced in Eq. (30) with σ and the nonlinear amplitude is $\zeta = \Re(q_k)$, then

$$\frac{\partial\sigma}{\partial\zeta} = \frac{t_1}{t_{2k+1}} \tag{31}$$

If this quantity is positive, then the limit cycle is unstable; otherwise, it is stable. All that is needed is the tangent corresponding to the modified set of variables, and that can be obtained by doing a rank-one update to the Jacobian:

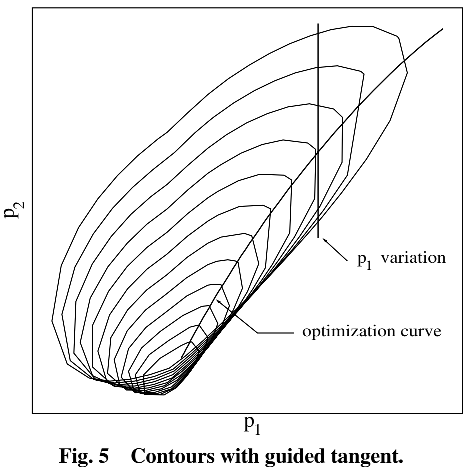

Fig. 5 Contours with guided tangent.

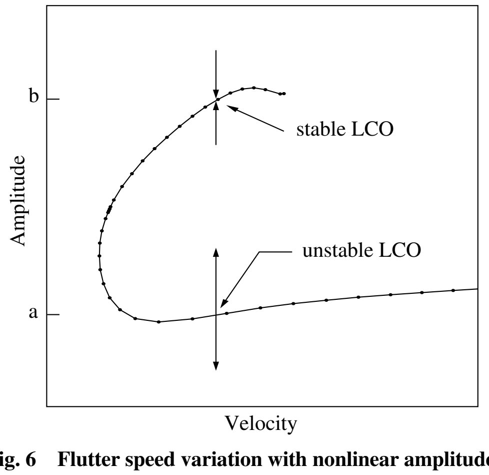

Fig. 6 Flutter speed variation with nonlinear amplitude.

$$\hat{\boldsymbol{J}} = \boldsymbol{J} + \boldsymbol{u}\boldsymbol{v}^t \tag{32}$$

where

$$\begin{aligned} \boldsymbol{u} &= \rho\left(\frac{\partial \boldsymbol{y}}{\partial \sigma}\right) - \boldsymbol{J}_{:1} \in \mathbb{R}^m \\ \boldsymbol{v} &= [1 \quad 0\ldots0]^t \in \mathbb{R}^n \end{aligned} \tag{33}$$

effectively replacing column one of the Jacobian with the partial of the residual with respect to σ. It is not necessary to actually make this replacement; only the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are of interest because, with them, the QR factorization of the Jacobian can be updated ([13] p. 334), and from that the desired tangent for Eq. (31) is $\boldsymbol{Q}_2$ [Eq. (13)]. Thus, the determination of limit cycle stability requires little additional effort.

X. Speeding-Up Curve Tracking

An advantage in tracking each aeroelastic mode separately as the continuation method does is that it is almost trivial to parallelize the computation because the tracking of each mode is entirely independent of the tracking of other modes, and so they may be done in parallel. Modern multicore, multi-CPU, and GPU architectures offer great potential for speeding up flutter analyses using continuation methods.

Much of the computational expense in the continuation method is in factorizing the Jacobian. Two things can be done to mitigate this expense: a modified Newton-type corrector, and updating the Jacobian factorization. Modified Newton's method simply uses a Jacobian for multiple corrector iterations; more important is the reduction in factorizations. Convergence for a modified Newton's method is slower, but whether that is compensated by the factorization savings is problem-dependent.

Updating the Jacobian factorization is an improvement on the modified Newton's method. The Jacobian is evaluated and factorized at the predicted point and used for the first corrector iteration. For subsequent corrector iterations, the QR factors are updated using the changes in $\boldsymbol{x}$ and $\boldsymbol{f}$. Broyden's update formula ([9] p. 61) is a rank-one update to the Jacobian:

$$\boldsymbol{J}(\boldsymbol{x}^{i+1}) \approx \boldsymbol{J}(\boldsymbol{x}^{i}) + \boldsymbol{u}\boldsymbol{v}^t \tag{34}$$

where

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{f}(\boldsymbol{x}^{i+1})/\|\boldsymbol{h}^i\| \in \mathbb{R}^m \\ \boldsymbol{v} &= \boldsymbol{h}^i/\|\boldsymbol{h}^i\| \in \mathbb{R}^n \end{aligned} \tag{35}$$

Given $\boldsymbol{u}$ and $\boldsymbol{v}$, the QR factorization of $\boldsymbol{J}^t(\boldsymbol{x}^i)$ can be updated ([13] p. 334) and used for corrector step $i+1$ in Eq. (14).

XI. Conclusions

Formulating the flutter equation as a system of real nonlinear equations in which all variables, including the generalized coordinates, are treated equally and solving with a continuation method is a very effective way of including most terms encountered in modern flutter analyses and (almost) guarantees continuous curves. A predictor–corrector continuation method using a QR factorization allows for more unknowns than equations and can be used to guide curves toward optimal or target sets of model parameters.

Several simple examples illustrate how these techniques might be used as well as the tremendous flexibility gained by this formulation of the flutter equation. Although the techniques were illustrated by a very small model, they have been used routinely on models approaching 300 degrees of freedom.

References

[1] 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

[2] Rodden, W. P., "Handbook for Aeroelastic Analysis," MSC/NASTRAN Ver. 65, Santa Ana, CA, Nov. 1987.

[3] Cardani, C., and Mantegazza, P., "Continuation and Direct Solution of the Flutter Equation," *Computers and Structures*, Vol. 8, No. 2, 1978, pp. 185–192. doi:10.1016/0045-7949(78)90021-4

[4] Meyer, E. E., "Application of a New Continuation Method to Flutter Equations," *29th AIAA Structures, Structural Dynamics, and Materials Conference*, AIAA Paper 1988-2350, April 1988.

[5] Šiljak, D. D., *Nonlinear Systems*, Wiley, New York, 1969.

[6] Gordon, J. T., Meyer, E. E., and Minogue, R. L., "Nonlinear Stability Analysis of Control Surface Flutter with Freeplay Effects," *Journal of Aircraft*, Vol. 45, No. 6, 2008, pp. 1904–1916. doi:10.2514/1.31901

[7] Griewank, A., and Walther, A., *Evaluating Derivatives: Principles and Techniques of Automatic Differentiation*, Vol. 105, Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, 2008.

[8] Brogan, W. L., *Modern Control Theory*, 3rd ed., Prentice–Hall, Upper Saddle River, NJ, 1991.

[9] Allgower, E. L., and Georg, K., *Numerical Continuation Methods*, Springer–Verlag, New York, 1990, pp. 3, 13.

[10] Rheinboldt, W. C., "Numerical Analysis of Continuation Methods for Nonlinear Structural Problems," *Computers and Structures*, Vol. 13, Nos. 1–3, 1981, pp. 103–113. doi:10.1016/0045-7949(81)90114-0

[11] Rheinboldt, W. C., "A Locally Parameterized Continuation Process," *ACM Transactions on Mathematical Software*, Vol. 9, No. 2, 1983, pp. 215–235. doi:10.1145/357456.357460

[12] Den Heijer, C., and Rheinboldt, W. C., "On Steplength Algorithms for a Class of Continuation Methods," *SIAM Journal on Numerical Analysis*, Vol. 18, No. 5, 1981, pp. 925–947. doi:10.1137/0718066

[13] Golub, G. H., and Van Loan, C. F., *Matrix Computations*, 4th ed., Johns Hopkins Univ. Press, Baltimore, MD, 2013.

[14] 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

W. Silva
Associate Editor