C.P. No. 1046

C.P. No. 1046

MINISTRY OF TECHNOLOGY

AERONAUTICAL RESEARCH COUNCIL

CURRENT PAPERS

Methods of Solving the Flutter Equations in use at R.A.E.

by

Ll. T. Niblett

Structures Dept., R.A.E., Farnborough

LONDON. HER MAJESTY'S STATIONERY OFFICE

1969

PRICE 5s 6d NET

U.D.C. 533.6.013.422 : 532.511 : 512.831 : 518.5

C.P. No.1046*
February 1968

METHODS OF SOLVING THE FLUTTER EQUATIONS IN USE AT R.A.E.

by

Ll. T. Niblett

SUMMARY

Details are given of some of the methods of solving the flutter equations which have been programmed in Mercury Autocode at the R.A.E. Included is a method of following the variation of critical speed with a linear variation of the structural inertia, damping or stiffness.

*Replaces R.A.E. Technical Report 68025 - A.R.C. 30502.

CONTENTS

1 INTRODUCTION

Flutter calculations involve the solution of a matrix equation which can be put in the form

$$[A \lambda^2 + (\sigma^{\frac{1}{2}} B \upsilon + D) \lambda + C \upsilon^2 + E] q = \{0\} \qquad (1)$$

where A, D and E are real matrices associated with the structural properties of the system, B and C are real matrices associated with its aerodynamic properties, σ is the relative air density, $-i\lambda$ is a scaled complex frequency, q the latent vector and υ is a scaled airspeed. The values of υ for which one of the possible λ is purely imaginary are of particular interest as they correspond to the critical flutter speeds. During the past few years various forms of the solution of these equations have been programmed in Mercury CHLF3 Autocode, a subset of Extended Mercury Autocode. This Report records details of a few of the programmes and also gives their specifications.

For convenience and to reduce the amount of transcription necessary, data common to all programmes is specified in a common form. For the same reason it has been arranged that particular rows and columns of the matrices can be selected so that any set of equations whose coefficients are included in those of the full set can be solved with the minimum of data changes.

2 PROGRAMME R.A.E. 272A - SOLUTION OF FLUTTER EQUATIONS

This programme finds all the roots λ of equation (1) for specified values of υ. It can also be used to find a value of υ for which the modulus of the real part of any of the λ is smaller than a specified fraction of its imaginary part.

2.1 Basis of solution

The method used is one of quadratic interpolation which Muller[1] developed for the solution of polynomial equations but which is equally applicable to the solution of determinantal equations. The determinant of the square matrix of equation (1) is evaluated, with υ constant, for three complex values of λ giving complex Δ_0, Δ_1, Δ_2 say. A parabolic curve fitted to the three points (λ_0, Δ_0) etc. will intersect the λ-plane at two points and the point of intersection nearer λ_2 is taken as the next approximation, λ_3, to the solution. Δ_3 is evaluated, the first point (λ_o, Δ_o) is discarded and the process is continued until the difference between successive approximations, $|\lambda_r - \lambda_{r-1}| / |\lambda_r|$, is sufficiently small. Subsequent to the first solution it is imperative that the solutions already found are suppressed so that they will not be found again and this is done by dividing the values of the determinant by $\prod_{j=1}^{r-1} (\lambda - \bar{\lambda}_j)$ where the $\bar{\lambda}_j$ are the solutions already found. Since the matrices A etc. are real, complex solutions occur in conjugate pairs and after a complex solution has been found both it and its conjugate are suppressed. Should the last solution of the equation be real it is found by evaluating $\Delta_{\lambda=i} \Big/ \prod_{j=1}^{2n-1} (i - \bar{\lambda}_j)$ since $\bar{\lambda}_{2n}$ it is the negative of the real part of this function divided by the imaginary part.

2.2 Scaling

Failure will occur if the value of the determinant becomes so large that it overflows the accumulator of the computer or so small that it is insignificant. Calculations can be made satisfactorily for a wide range of absolute values of λ, υ and the elements of the matrices but the risk of failure will be eliminated if the values of these are of the order of unity. Since the scaling of the coefficient matrices relative to each other depends on the scaling of λ and υ, only an overall factor can be applied to them and one which brings the elements of the leading diagonal of A to the order of unity will be suitable in most cases.

2.3 Preliminaries

The matrices are read into backing stores of the computer and the selected rows and columns are subsequently copied into further backing stores. Should $\sigma^{\frac{1}{2}}$ be other than unity the B matrix is multiplied by it during transit.

The time taken to find a root depends, to a large extent, on how closely it can be estimated. The roots vary continuously with υ so that the roots for one value of υ can be used as estimates for the next value. Initially, however, estimates have to be provided in some other way. With flutter coefficients, the roots of the individual equations $A_{rr} \lambda^2 + D_{rr} \lambda + E_{rr} = 0$ are generally reasonably good estimates and these equations are solved to provide the starting values for the first υ. These estimates are sorted according to whether they are real or complex and then arranged in order of modulus.

The flutter equations are often such that equation (1) has roots $\lambda = 0$. Should this be the case, there is a danger that accumulator overflow will occur if the methods outlined in section 2.1 are used. This is possible either in the test for convergence, since this is based on the absolute value of the root, or, in the suppression of the roots if there are multiple roots $\lambda = 0$ and at least one of these has already been found. The behaviour of the determinant near zero is therefore examined and the number of zero roots determined (see section 2.4) before solution by the normal method is attempted. Multiple non-zero roots are found successfully by the normal method.

If the equation has multiple zero roots when υ is not zero it will almost certainly have at least one zero root when υ is zero and hence E will be singular. E is taken to be singular either if a first estimate of a root is zero or if the determinant of E is considerably less than the product of the elements of the leading diagonal. Determinants are evaluated by triangular decomposition with row interchanges[2] using single length arithmetic throughout.

2.4 Main Programme

First the value of υ is substituted in $(B\,\upsilon + D)$ and $(C\,\upsilon^2 + E)$. If E has been adjudged singular the determinant is evaluated for a real λ which is a small fraction of the imaginary part of the estimated complex λ of smallest modulus and also for an argument of twice this. The logarithm to the base two, if positive, of the ratio of the second determinant to the first gives an approximation to the multiplicity of zero roots. This method has worked well in practice even correctly divining in one instance that an equation with 12th-order matrices had 18 zero roots.

The remaining roots are then found using the estimates, complex values first, in order of increasing modulus. Should the number of zero roots found be less than the number estimated, zero estimates are replaced by small complex numbers to avoid accumulator overflow. The three values taken to commence interpolation are λ close to the estimate and on either side and the estimate itself. At each iteration the increment in λ is compared with the increment at the last iteration and if it is greater than it by a factor which varies directly as the number of roots already found its size is reduced to that of the last increment. Also the increment is reduced by half if the value of the determinant for the full increment is not less than ten times the value of the last determinant accepted. Iteration is terminated either when the difference between successive approximations is less than a small fraction of the latest approximation or when the difference between successive approximations increases after first being reduced to a reasonably small fraction of the approximation. If iteration is terminated in the latter way the printed root is marked by an asterisk.

Roots are stored for reference and output as they are found. Real roots will be printed floating point with six decimal places. The imaginary parts of complex roots will be printed in the same form but the real part will be printed as $-100\mu\,|\lambda|^{-1}$, $(\lambda = \mu + i\nu)$, fixed point with four decimal places and also as μ itself in floating point with six decimal places if they are large. Only one of each conjugate complex pair is output and the original estimates provide a bias in favour of the ones with a positive imaginary part. Occasionally however ones with negative imaginary parts are found. When all the roots have been found the sum of their real parts is output to provide a check on the working of the programme. This sum is proportional to the coefficient of λ^{2n-1} in the characteristic polynomial and is linear with υ.

2.5 Approximate critical υ

Solutions can be found for a series of υ input individually or for υ at regular intervals between two limits. Also the smallest critical υ, i.e. the smallest υ at which, μ, the real part of one of the λ changes from negative to positive, can be found approximately. In this last case three values of υ, υ_o, υ_1 and υ_2 say, are input together with two further numbers ε and δ. Solutions are obtained for a series of values $\upsilon = \upsilon_o$, $\upsilon_o + \upsilon_1$, $\upsilon_o + 2\upsilon_1$ etc. up to or just beyond υ_2 (actually $\upsilon < \upsilon_2 + 0.9\,\upsilon_1$) until, for one root, both $|\lambda|$ and $\mu/|\lambda|$ are greater than ε. The purpose of the reference to ε is to enable one to ignore near-zero spurious roots and roots with small spurious real parts due to round-off or similar causes. When a positive real part is found for a υ other than υ_o interpolation is commenced with the object of finding a value of υ at which $|\mu/\lambda| < \delta$ if this has not been done already. The roots for the previous value of υ are always retained to facilitate this on the assumption that the order in which the roots are found does not change with υ. The next υ to be tried is the result of linear interpolation between the positive μ and the μ which has the same position among the roots found at the previous speed. Subsequent interpolation is by the second-order Newton formula. If no μ is positive at a particular υ the μ in the same position as the last positive μ to be found is used. When a sufficiently small value of μ has been found the rest of the range of υ is covered in steps of υ_1 recommencing at the υ above the one at which the first positive μ was found but no attempt is made to find further critical speeds.

2.6 Further comments

The use of Muller's method on the determinantal equation itself involving as it does the repeated evaluation of complex determinants is a comparatively slow method of solving the equation. However, all roots are found directly from the original data and single-length arithmetic has been accurate enough to provide reliable results. Further the order of the equations is not doubled by replacement of the original equations by linear ones as is the case when a library latent root programme is used. The method becomes even more attractive if the elements of the determinant are higher-order polynomials in λ.

3 PROGRAMME R.A.E. 306A - CRITICAL FLUTTER CONDITIONS

The vectors associated with the roots of equation (1) are not found in the course of a solution by Muller's method. Sometimes the vector at the critical values of υ and λ is of interest however and a programme has been written which finds it from data in the form specified for R.A.E. 272A. The critical values of υ and λ do not have to be known accurately since an attempt is made to improve the approximation to the true values whatever the datum values used.

3.1 Basis of solution

The programme is based on inverse iteration with shift of origin using the linear equivalent of equation (1)

$$\begin{bmatrix} -A^{-1}(\sigma^{\frac{1}{2}} B \upsilon + D) & -A^{-1}(C \upsilon^2 + E) \\ I & 0 \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix} = \lambda \begin{bmatrix} p \\ q \end{bmatrix} . \qquad (2)$$

Iteration according to the equation

$$\begin{bmatrix} A^{-1}\bar{B} + \pi I & A^{-1}\bar{C} \\ -I & \pi I \end{bmatrix} \begin{bmatrix} p_{s+1} \\ q_{s+1} \end{bmatrix} = -(\lambda - \pi) \begin{bmatrix} p_s \\ q_s \end{bmatrix} , \qquad (3)$$

where $\bar{B} \equiv \sigma^{\frac{1}{2}} B\, \upsilon + D$ and $\bar{C} \equiv C\, \upsilon^2 + E$, results in convergence to the latent root nearest π. Equation (3) can be written

$$\pi\, q_{s+1} = p_{s+1} - (\lambda - \pi)\, q_s \qquad (4)$$

$$(\pi^2 A + \pi \bar{B})\, p_{s+1} + \pi \bar{C}\, q_{s+1} = -(\lambda - \pi)\, \pi A\, p_s \ . \qquad (5)$$

Substituting for q_{s+1} from equation (4) in equation (5)

$$(\pi^2 A + \pi \bar{B} + \bar{C})\, p_{s+1} = (\lambda - \pi)\, (\bar{C}\, q_s - \pi A\, p_s) \ . \qquad (6)$$

The equations used in the programme are (4) and (6). π is the imaginary number which is the known approximate critical value of λ and $\bar{B}$ and $\bar{C}$ are evaluated for υ_o the approximate critical value of υ. $(\pi^2 A + \pi \bar{B} + \bar{C})$ is decomposed with row interchanges to the product of lower and upper triangular matrices. Equation (6) is solved for p_1 and $(\lambda - \pi)$ using arbitrary vectors p_o and q_o as data. Equation (4) is solved for q_1 and equation (6) is solved again using p_1 and q_1 as data. The process is repeated until the sum of the squares of the differences between the elements of consecutive $\{p\ q\}$ is a minimum. $(\lambda - \pi)$ will, in general, be a complex number.

$(\pi^2 A + \pi \bar{B} + \bar{C})$ is evaluated for υ_1, a value of upsilon near υ_o, the iteration is repeated and the value of $(\lambda - \pi)$ found. The next value of υ for which $(\lambda - \pi)$ is found is chosen to be that at which a straight line passing through the real parts of λ already found passes through zero. Subsequently interpolation is by the second-order Newton formula and is terminated either when the real part of λ is sufficiently close to zero or ceases to be reduced. The values of υ and λ are the critical values and are output together with the vector.

The deflections in the flutter mode (which are complex) can be output if the deflection matrix for the coordinates is available.

The elements of the matrix

$$[A\,\lambda^2 + (\sigma^{\frac{1}{2}} B\,\upsilon + D)\,\lambda + C\,\upsilon^2 + E]\,{}^{\ulcorner}q_{\lrcorner} \;=\; F \text{ (say)} \qquad (7)$$

where ${}^{\ulcorner}q_{\lrcorner}$ is a diagonal matrix of the elements of the vector can also be output for the critical values of λ, υ and q.

F gives the generalised forces in each coordinate due to the displacements in all the coordinates. The product of F post-multiplied by a column of ones is a null column.

4 PROGRAMME R.A.E. 324A - FLUTTER SPEED LOCUS

The programme calculates the variation of υ with any one of the μ which satisfy the equation

$$\left|(A + \mu_a\,A_\mu)\,\lambda^2 + \{\sigma^{\frac{1}{2}} B + (D + \mu_d\,D_\mu)\,\upsilon^{-1}\}\,\lambda + C + (E + \mu_e\,E_\mu)\,\upsilon^{-2}\right| = 0 \qquad \ldots\ldots (8)$$

where all the matrices and scalars are real apart from λ which is purely imaginary. It will be noticed that λ is a frequency parameter rather than a frequency in this case. The choice of argument of the equation is influenced largely by the relative likelihood of υ tending to zero and to infinity. In the system principally in mind υ is far more likely to tend to infinity than zero and the frequency-parameter form has been chosen. The coefficient matrices are the same for both forms.

The graph of υ^{-1} against μ is followed by assuming its basic form and extrapolating an estimate for the next point from the positions of previous points rather than by basing the extrapolation on the derivatives at the last point. An accurate point on the graph is found by interpolation on the normal to the assumed curve at the estimated point.

4.1 Description of the method

The basic form of the graph of υ^{-1} against μ is assumed to be a circle. Once three points have been found (say A, B, C of Fig.1) a circle is put through them and the first approximation (D) to the next point is the mirror image of the penultimate point (B) in the radius through the last point (OC). Interpolation to critical values of υ^{-1} and λ is similar to that of R.A.E. 306A (see section 3.1) although not at constant μ but along the radius OD which is a line of constant $(\mu \cos \theta - \upsilon^{-1} \sin \theta)$, where θ is the angle from OD to the υ^{-1} axis and the increments are in $(\mu \sin \theta + \upsilon^{-1} \cos \theta)$ and not υ^{-1}. Iteration is terminated when the difference between successive approximations to $\mathscr{R}(\lambda)$ is either less than a preset value or not getting smaller. Interpolation along a particular line is stopped either if the angle between the tangent to the circle at C and the line joining J (see Fig.2) to the current approximation exceeds a preset value or if $\mathscr{I}(\lambda)$ differs by more than a preset amount from π, the approximate value of λ at the next point. J is a point on the tangent at C a short distance (depending on the first estimate of υ for the want of something better) from C and on the same side of C as the penultimate point. The test of $\mathscr{I}(\lambda) - \pi$ is omitted for the first few iterations to allow $\mathscr{I}(\lambda)$ a chance to settle. Interpolation is then tried on the line parallel to OD which bisects the chord CD. Should this also be unsuccessful subdivision of the chord is continued. The calculation is abandoned if no success is achieved with an increment which is 2^{-9} of the original. The calculation will also be abandoned if interpolation is still unsuccessful after a preset limit on the number of points tried at any particular value of $\mu \cos \theta - \upsilon^{-1} \sin \theta$ is exceeded.

The size of the increment can also be increased up to a maximum value specified in the data. After each point on the curve is found the radius of the latest circle is compared with that immediately previously and if it does not show a reduction of more than 20% the increment is increased. In Fig.1, F, the next approximate point, is at the intersection of DE and the circle and the distance BE is 50% greater than the distance BD. Should the increment be limited to the maximum it will be modified so that the next approximate point is on the circle (e.g. G to H). The size of the increment will not be increased if it was reduced during the finding of the last point.

To begin the calculation an approximate critical υ and frequency parameter π must be known for the first value of the parameter μ. The programme starts by refining these and uses the more accurate values as first approximations at the next value of μ which is a preset small fraction of the maximum increment from the initial μ. Interpolation at this value of μ is stopped and the increment halved when the same limits as for a general point are exceeded, the tangent at the first point being taken to be a line parallel to the μ axis.

The interpolation for the third point is started on the assumption that the curve is a straight line through the first two points. The increment between the first and second points is doubled to obtain the initial increment for finding the third point and similarly the increment between the second and third points is doubled to give the increment for the fourth point.

After the first two points have been found the pole, π, for the next point is based on the λs at and the relative positions of the last two points. Referring to Fig.1 the value of π for the approximate point F would be $\lambda_C + (\lambda_C - \lambda_B)\,\frac{\text{chord CF}}{\text{chord BC}}$. The π for the initial point is the frequency parameter in the data. For the second point it is the accurate frequency parameter at the first point.

The calculation is terminated when the value of μ is smaller than its initial value or larger than the nominal maximum value input as data or when υ is larger than the nominal maximum value input as data.

The programme has been developed using cases in which the range of μ was of the same size as that of υ^{-1} and it is advisable that the matrices should be scaled so that the likely $(\mu_{max} - \mu_{min})$ is of the same order as the likely $\left(\frac{\upsilon_{max} - \upsilon_{min}}{\upsilon_{max}\ \upsilon_{min}}\right)$ so that the limits incorporated in the programme are applicable.

4.2 Experience

This was the last of the programmes described to be written and little experience of it in general use has been obtained. However, the test cases used in its development were picked for their difficulty. One of them is shown in Fig.3. The presence of a 'knot' in one of the curves was unsuspected before the programme was used.

Appendix A

SPECIFICATION - R.A.E. 272A - SOLUTION OF FLUTTER EQUATIONS

A.1 The programme finds the roots, λ, of the equation

$$|A\,\lambda^2 + (\sigma^{\frac{1}{2}} B\,\upsilon + D)\,\lambda + C\,\upsilon^2 + E| = 0$$

where $A, B, C, D, E, \sigma, \upsilon$ are real.

A.2 Data

(1) An integer (p') which is limited to up to 15 in the case of the Mercury programme and 30 in the case of the Atlas programme and is the order of the matrices A, B etc. input.

(2) The matrices A, B etc. in any order preceded by the appropriate letter. They are overwritten only when further matrices are input.

(3) The n row - and - column numbers, in the range 1 to p', of the elements of the matrices input which are the elements of the submatrices on which the calculation is to be made, enclosed in brackets, e.g. (1 3 4) remembering that each number must be terminated.

The value of $\sigma^{\frac{1}{2}}$.

(4) An integer t - followed by:-

If $t = 0$:- υ_1, υ_2 etc. terminated by * if the programme is to be re-entered.

The roots for $\upsilon = \upsilon_1, \upsilon_2$ etc. will be calculated.

If $t = 1$:- $\upsilon_o, \upsilon_1, \upsilon_2$.

The roots for $\upsilon = \upsilon_o, \upsilon_o + \upsilon_1$ etc., $\upsilon < \upsilon_2 + 0.9\,\upsilon_1$, will be calculated.
If $t = -1$:- $\upsilon_o, \upsilon_1, \upsilon_2, \varepsilon, \delta$.

The roots are found as for $t = 1$ but if a complex root, $\lambda = \mu + i\nu$, is found with $|\lambda| > \varepsilon$, $\mu/|\lambda| > \varepsilon$, an υ will be found at which $|\mu/\lambda| \leqslant \delta$.

If $t = -2$:- $\upsilon_o, \upsilon_1, \upsilon_2, \varepsilon, \delta, i', i'$ values of ν and $-100\,\mu/|\lambda|$, $2(n - i')$ values of μ.

This allows the introduction of estimates of the roots, i' of which are complex, other than those supplied by the programme. The values of $-100\ \mu/|\lambda|$ follow their respective ν immediately. Estimates that are zero must be input as the last real roots.

More data may be input starting with any of (1) to (4). If the character > is input the rest of the characters on its line will be output as a title and will be followed by a new line.

If the character ≏ is input an end instruction will be reached.

A.3 Output

(1) D. OF F. followed by the n row - and - column numbers.

(2) $\sigma^{\frac{1}{2}}$.

(3) $\underline{V} = \upsilon$.

(4) The roots of the equation. Complex roots are printed ν, floating point, followed by $-100\ \mu/|\lambda|$, fixed point and, if large, μ floating point. Real roots are printed floating point. If ℓ roots are zero the non-zero roots are preceded by 'ℓ' ZEROS.

(5) REAL SUM sum of real parts of roots. This should be linear with upsilon.

A.4 Data faults

If an illegal spurious character is found in the data an end instruction is reached. Before this happens FAULT K is output followed by a integer (k) and the spurious character except that a letter shift character is changed to an erase and a line-feed character will be followed by an erase.

k = 1 , illegal character is between the main items of section 2 above,
k = 12, character is in data of sub-section 2(4),
k = 13, character is in data of sub-section 2(3),
k = 20, character follows a letter-shift,
k = 21, character is in one of the matrices of 2(2) and the appropriate letter will be output before the illegal character.

Appendix B

SPECIFICATION - R.A.E. 306A - CRITICAL FLUTTER CONDITIONS

B.1 Given good approximations to an imaginary λ and real υ for which

$$[A\,\lambda^2 + (\sigma^{\frac{1}{2}} B\,\upsilon + D)\,\lambda + C\,\upsilon^2 + E]\,q = 0 \quad ,$$

where A, B, C, D, E, σ are real, the programme finds accurate values of λ and υ together with the corresponding q.

Given a matrix $Z(p' \times s)$ of the displacements in the generalised coordinates of a set of s points, it will output the displacements of the points in the flutter mode, z.

It will also find the matrix of generalised forces

$$[A\,\lambda^2 + (\sigma^{\frac{1}{2}} B\,\upsilon + D)\,\lambda + C\,\upsilon^2 + E]\ulcorner q \lrcorner \equiv F \quad .$$

B.2 Data

(1) An integer (p') which is limited to up to 15 in the case of the Mercury programme and 30 in the case of the Atlas programme and is the order of the matrices A, B etc. input.

(2) The matrices A, B, C, D, E, Z in any order preceded by the appropriate letter. They are overwritten only when further matrices are input. The matrix Z is a $p' \times s$ rectangular matrix and the integer s ($p'\,s \leqslant 2{,}250$ Mercury, 4,500 Atlas) must be input immediately after the letter Z.

(3) The n row - and - column numbers, in the range 1 to p', of the elements of the matrices input which are the elements of the submatrices on which the calculation is to be made enclosed in brackets, e.g. (1 3 4) remembering that each number must be terminated.

The value of $\sigma^{\frac{1}{2}}$.

(4) An integer ℓ followed by the approximate critical values of υ and $\vartheta(\lambda)$.

If $\ell = -1$ only υ and $\vartheta(\lambda)$ will be output

$\ell = 0$ υ, $\vartheta(\lambda)$, q and z will be output

$\ell = 1$ υ, $\vartheta(\lambda)$, q, z and F will be output

z will only be output if a Z has been input and can be suppressed by the subsequent input of p'.

More data may be input starting with any of (1) to (4). If the character > is input the rest of the characters on its line will be output as a title and followed by a new line.

If the character ≏ is input an end instruction will be reached.

B.3 Output

(1) D. OF F. followed by the n row - and - column numbers.

(2) $\sigma^{\frac{1}{2}}$.

(3) CRITICAL SPEED υ , FREQUENCY $\vartheta(\lambda)$.

(4) VECTOR

q

(5) DISPLACEMENTS

z

(6) FORCES

F

B.4 Data faults

If an illegal spurious character is found in the data an end instruction is reached. Before this happens FAULT K is output followed by an integer (k) and the spurious character except that a letter-shift character is changed to an erase and a line-feed character will be followed by an erase.

k = 1 , illegal character in between the main items of section 2 above,

k = 12, illegal character is in the data of sub-section 2(4),

k = 13, illegal character is in the data of sub-section 2(3),

k = 20, character follows a letter shift,

k = 21, character is in one of the matrices 2(2) and the appropriate letter will be output before the illegal character.

Appendix C

SPECIFICATION - R.A.E. 324A - FLUTTER SPEED LOCUS

C.1 The flutter equation is taken to be in the form

$$\left|(A + \mu_a[\psi A])\,\lambda^2 + (\sigma^{\frac{1}{2}} B + \upsilon^{-1}\,\{D + \mu_d[\psi D]\})\,\lambda + C + (E + \mu_e[\psi E])\,\upsilon^{-2}\right| = 0 .$$

Given reasonably good approximations to the critical speed υ and the frequency parameter λ at the initial value of the structural parameter μ, the programme enables the variation of υ with μ to be traced until either μ exceeds a value input or is less than its initial value or υ exceeds a value input. Only one of μ_a, μ_d, μ_e can be used at a time. The programme is written in Mercury CHLF 3/4 Autocode and will be compiled by an EMA compiler.

C.1.1 Scaling

The speed and frequency-parameter scales should be such that λ and υ are of order unity. For safety the structural parameter scale should be such that $\mu_{max} - \mu_{min} \approx \dfrac{\upsilon_{max} - \upsilon_{min}}{\upsilon_{max}\ \upsilon_{min}}$.

C.2 Data tape

(1) An integer $p' \leqslant 30$ which is the order of the matrices A, B etc. on the tape.

(2) The matrices A, B, C, D, E, ψA (or ψD or ψE) in any order preceded by the appropriate letters. The only characters permitted between the letter shift (or ψ) character and the letter character are the shift characters and erase.

(3) The n row - and - column numbers in the range 1 to p' of the elements of the matrices already read which are the elements of the submatrices on which the calculations are to be made enclosed in brackets, e.g. (1 3 4) remembering that all the numbers must be terminated.

The value of $\sigma^{\frac{1}{2}}$.

(4) An asterisk * followed by the initial value of μ, an approximate value for the initial υ, ditto for λ, the maximum increment in μ desired, the maximum μ of interest and the maximum υ of interest.

After (4) is read computation is commenced. At its conclusion more data can be read starting at any of (1) to (4).

C.2.1 Titles

Characters on the data tape which follow the character > are output immediately up to and including the next carriage-return character and are followed by a line-feed character. Titles may be put before (1), (3), (4) or any of (2).

C.2.2 End

If the character ≏ is read the caption END is output and an instruction end is reached.

C.3 Output

D. OF F. followed by the n row - and - column numbers

$\sigma^{\frac{1}{2}}$

A (or D or E)	V	NU
Values of μ	Values of υ	Values of $\phi(\lambda)$

Lost

If the programme fails to find the continuation of the curve at any point the caption LOST is output and computation is concluded.

Data faults

If an illegal spurious character is found in the data the caption FAULT K is output together with the value of the index k and the spurious character except that a letter shift character will be changed to an erase and a line-feed character will be followed by an erase.

If $k = 1$, character is before (1), (3), (4) or any of (2),
$k = 12$, character is in the number after the asterisk,
$k = 13$, character is between the brackets,
$k = 19$, character follows ψ,
$k = 20$, character follows a letter shift,
$k = 18$, character is in the ψ matrix,
$k = 21$, character is in one of the matrices of the data (2) apart from the ψ and the appropriate letter will be output.

REFERENCES

No.	Author	Title, etc.
1	D.E. Muller	A method for solving algebraic equations using an automatic computer. Math. Tab. 10, 208-215 (1956)
2	J.H. Wilkinson	The algebraic eigen value problem. Oxford, Clarendon (1965)

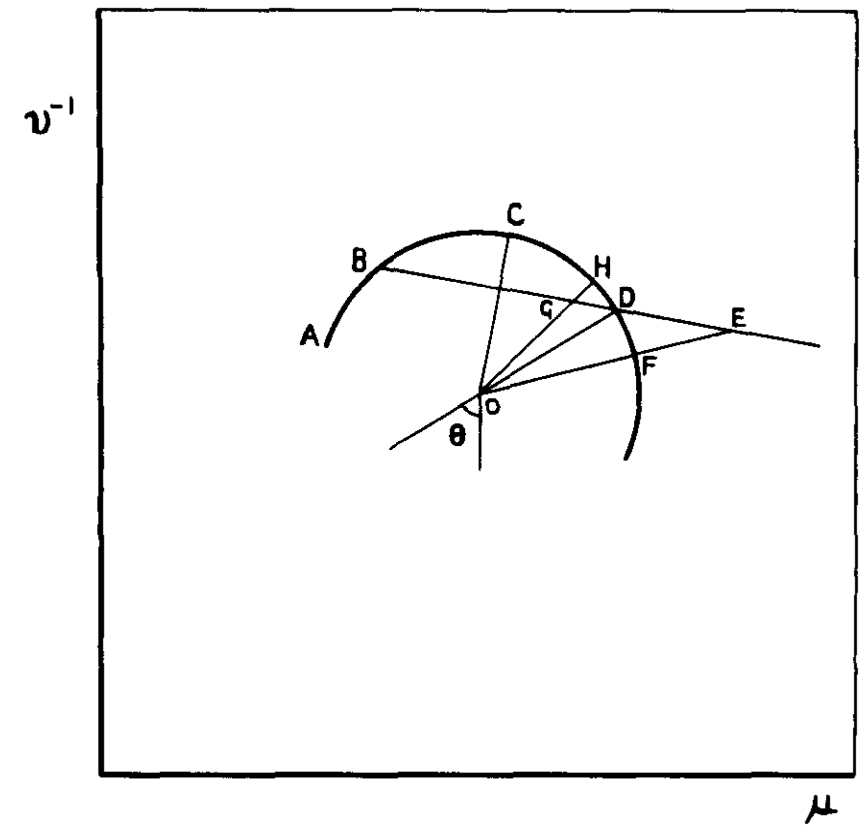

Fig. 1 Construction for first approximation to next point

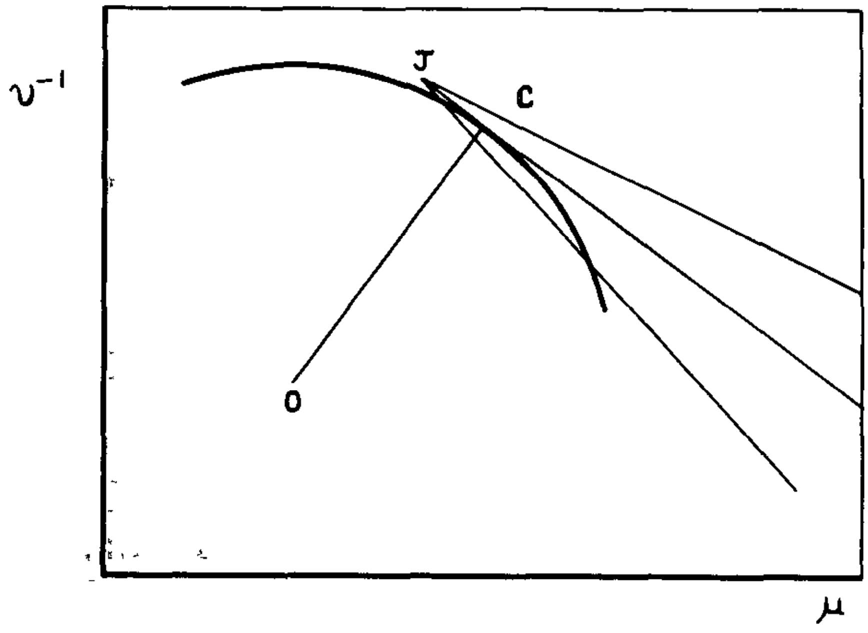

Fig. 2 Range within which next point can be found

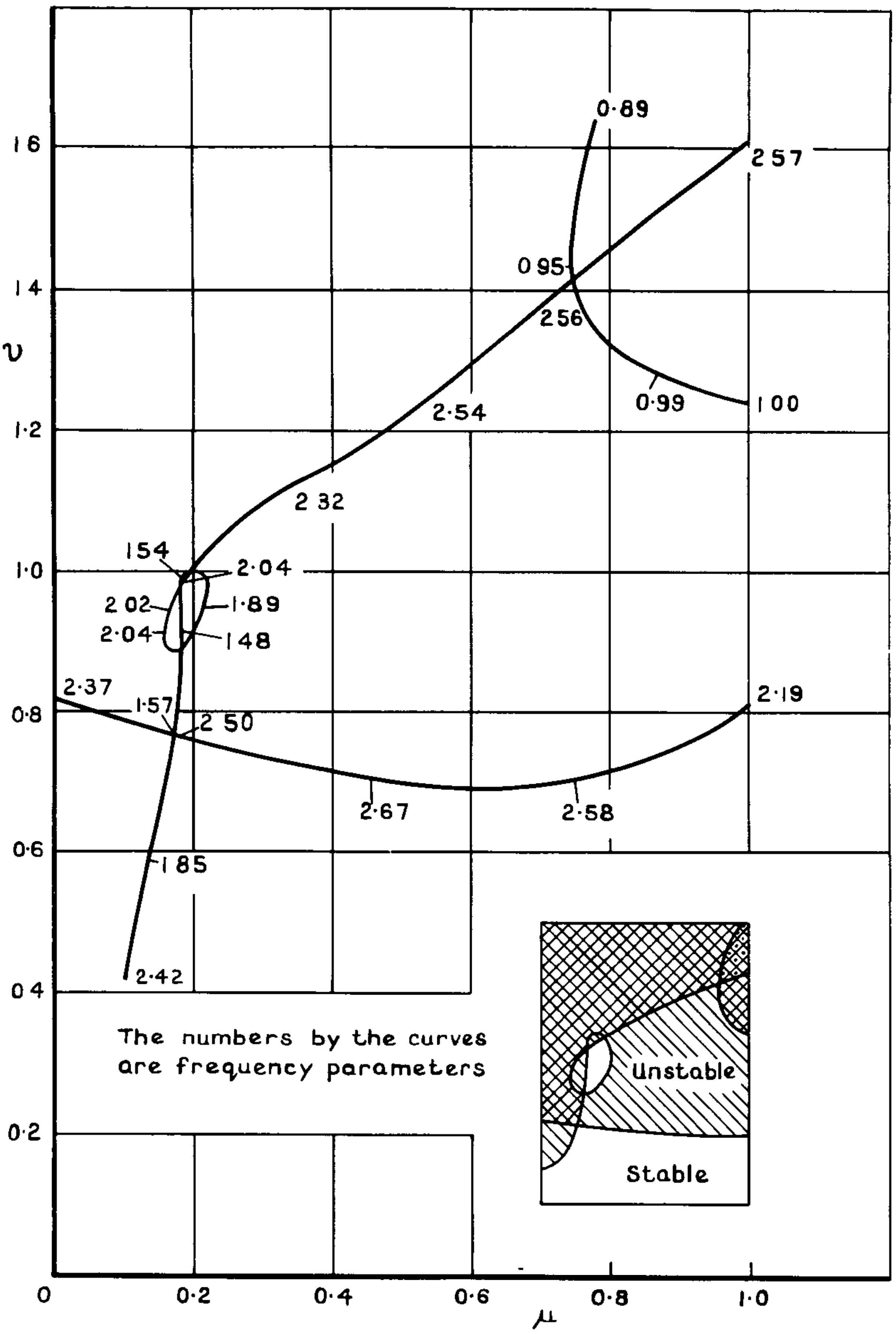

Fig.3 Example of variation of flutter speed with control stiffness

Printed in England for her Majesty's Stationery Office by the Royal Aircraft Establishment, Farnborough. Di.146915. K.3.

A.R.C. C.P. No.1046
February 1968

Niblett, Ll. T.

533.6.013.422 :
532.511 :
512.831 :
518.5

METHODS OF SOLVING THE FLUTTER EQUATIONS IN USE AT R.A.E.

Details are given of some of the methods of solving the flutter equations which have been programmed in Mercury Autocode at the R.A.E. Included is a method of following the variation of critical speed with a linear variation of the structural inertia, damping or stiffness.

A.R.C. C.P. No.1046
February 1968

Niblett, Ll. T.

533.6.013.422 :
532.511 :
512.831 :
518.5

METHODS OF SOLVING THE FLUTTER EQUATIONS IN USE AT R.A.E.

Details are given of some of the methods of solving the flutter equations which have been programmed in Mercury Autocode at the R.A.E. Included is a method of following the variation of critical speed with a linear variation of the structural inertia, damping or stiffness.

A.R.C. C.P. No.1046
February 1968

Niblett, Ll. T.

533.6.013.422 :
532.511 :
512.831 :
518.5

METHODS OF SOLVING THE FLUTTER EQUATIONS IN USE AT R.A.E.

Details are given of some of the methods of solving the flutter equations which have been programmed in Mercury Autocode at the R.A.E. Included is a method of following the variation of critical speed with a linear variation of the structural inertia, damping or stiffness.

A.R.C. C.P. No.1046
February 1968

Niblett, Ll. T.

533.6.013.422 :
532.511 :
512.831 :
518.5

METHODS OF SOLVING THE FLUTTER EQUATIONS IN USE AT R.A.E.

Details are given of some of the methods of solving the flutter equations which have been programmed in Mercury Autocode at the R.A.E. Included is a method of following the variation of critical speed with a linear variation of the structural inertia, damping or stiffness.

DETACHABLE ABSTRACT CARDS

C.P. No. 1046

C.P. No. 1046

SBN 11 470173 3