https://ntrs.nasa.gov/search.jsp?R=19860015547 2020-03-20T15:30:45+00:00Z

1060

NASA Technical Memorandum 86081

Statistical Correlation Analysis for Comparing Vibration Data From Test and Analysis

T. G. Butler, R. F. Strang,
L. R. Purves, and D. J. Hershfeld
Goddard Space Flight Center
Greenbelt, Maryland

National Aeronautics
and Space Administration

**Scientific and Technical
Information Branch**

1986

TABLE OF CONTENTS

PRECEDING PAGE BLANK NOT FILMED

STATISTICAL CORRELATION ANALYSIS FOR COMPARING VIBRATION DATA FROM TEST AND ANALYSIS

INTRODUCTION

Up until the last 30 years, structural analysis consisted largely of abbreviated and approximate calculations. Typically, the calculations represented empirical knowledge more than an understanding of structural behavior. Limited analytical capabilities also lead to limitations in testing. Without the knowledge of how a structure should respond to loads, the testing of a new design often consisted simply of putting it to its intended use. If it worked sufficiently well for a sufficiently long time, it was deemed successful and became a model for future structures of a similar nature.

This situation changed drastically as the result of three major developments. With the advent of the digital computer in the 1940s came the possibility of doing the large number of calculations required to determine accurately and in detail the stresses and deformations that a structure would experience when serving its intended functions. Next, the development of the finite element method in the 1950s provided algorithms, executable on digital computers, which could indeed calculate stresses and deformations in complex structures. Finally, software systems were developed to combine structural theory, finite element methods, and computer hardware into useful tools for structural analysis.

The NASTRAN software system (NASA STRuctural ANalysis), development of which was begun by NASA in 1965, is an example of such a system which has received wide usage. There were a number of reasons for its acceptance. First, it integrated into a unified set of software a sufficient number of algorithms and data types to carry out the majority of structural analyses. Second, NASTRAN included software to overcome many of the deficiencies of early computers, such as limited memory size, which would have otherwise limited the complexity of finite element models. This meant that the limitation on the complexity of an analyst's model shifted to the amount of computer time he could obtain. Finally, extensive effort went into providing good user interfaces, complete documentation, and wide dissemination. To enable wide use, NASTRAN is written mainly in a computer independent subset of FORTRAN, which permits it to run on 5 different makes of computers. The total capabilities provided by NASTRAN have contributed significantly to the present situation, where finite element analysis using digital computers is not only the dominant means of accomplishing structural analysis, but it is also widely used for analysis in heat transfer, aerodynamics, acoustics, control systems, potential theory, and actually any problem which can be solved by sets of linear equations.

The overall effect of this new tool has been to improve the quality of structural designs. With finite element techniques, the designer can be much more certain of reaching a desired tradeoff of structural efficiency, cost, and reliability than previously. However, the volumes of data produced by complex finite element analyses, now routinely performed, have created problems of interpretation, because of the human time required to assimilate large amounts of data. For the same reason, test data is often not completely compared to predicted analytical results. Often, if a structure goes through static and dynamic testing with no apparent problems, such as breakage or pronounced deflection or vibration, it is deemed adequate. When test data obviously disagrees with results predicted by finite element analysis, a too frequent practice is to assume that the analytical data is wrong and to adjust its parameters until satisfactory agreement with test data is achieved.

There are a number of ways to increase the value of combined analytical and test data on structural behaviour. The work described here was undertaken to derive some of these improvements. Fundamentally, the two sets of data provide a means of cross checking, thereby reducing the probability of undetected errors in either set. A principal problem in doing cross checking is that there is not a commonly accepted measure of comparison between analytical and test data. Often, this means that a few hundred data points from both sources, regarded as critical, are plotted in some way, and if the data plots appear reasonably alike to the eye, or can be made to look alike by reasonable alterations to the analytical model, then agreement is decreed.

With the techniques described below, it is possible to use NASTRAN to place analytical results in a format in which they can be statistically compared as a group to equivalent groups of test data gathered by automated test systems. Statistical routines can then provide criteria describing overall agreement and identify areas of significant disagreement. This means that far more data can be evaluated with far less human labor. This technique can help identify whether disagreement is due to faulty test procedures or faulty modeling. However, another promising long range benefit of the statistical correlation of analytical and test data is that structural design can be further improved. There is presently no widely accepted and effective way of validating analytical results on real problems. Statistical comparisons with equivalent test data appear to provide an acceptable means for this validation. If this technique becomes widely used and improved, then it will gradually lead to improved analytical techniques, thus leading in turn to improved designs and eventually to improved structures.

OBJECTIVE

When designs need to qualify under narrow allowances, as is normally the case with aerospace structures, it is often necessary to compare structural vibration properties obtained from testing with those obtained from analysis.

In order for a comparison to be meaningful, it is necessary to define the basis on which the data will be compared. In general, they will be compared on the basis of like modes. A test mode and an analytical mode are alike if the amplitude of the standing wave pattern is "exactly" the same for both at all sample points. The bothersome qualifier in this definition of like modes is the word 'exactly'. Even if one were to assign a reasonable tolerance for defining the word 'exact', it would be rare to find a test mode that measured exactly the same as an analytical mode. A more realistic definition of modes being "alike" is to require that the pair correlate closely. This definition allows some engineering judgement for prescribing the tolerance of "closeness."

There are no exact solutions to the vibration characteristics of complex three dimensional structures. Highly capable analytical techniques are available, but they are liable to human error. Highly capable experimental equipment is also available, but human error can enter the experimental side from two sources: in fabricating the test article and in conducting the test. Once a like pair is found, it can be expected that the frequency at which this mode appears is different for analysis than it is for test. Therefore, the purpose of this enterprise is not to judge the correctness of either set of data to represent a given design, but to point up differences and equip the responsible structural engineers with enough measures for them to be able to assess the areas of disagreement.

To illustrate how statistical comparison can help the structural engineer, consider the following example. After testing a collection of modes for likeness in a given frequency range, say 8 from test and 11 from analysis, it might be that 7 matches are found. Also, consider that one test mode and four analytical modes are not matched. In those that match, it is possible that the frequencies of the analytical modes are all higher than their test counterparts. It may also be that the one test mode without a match measures higher in frequency than all other test modes, while two of the analytical modes without matches measure lower in frequency than all other analytical modes, and the other two are in the middle range. Figure 1 depicts this example.

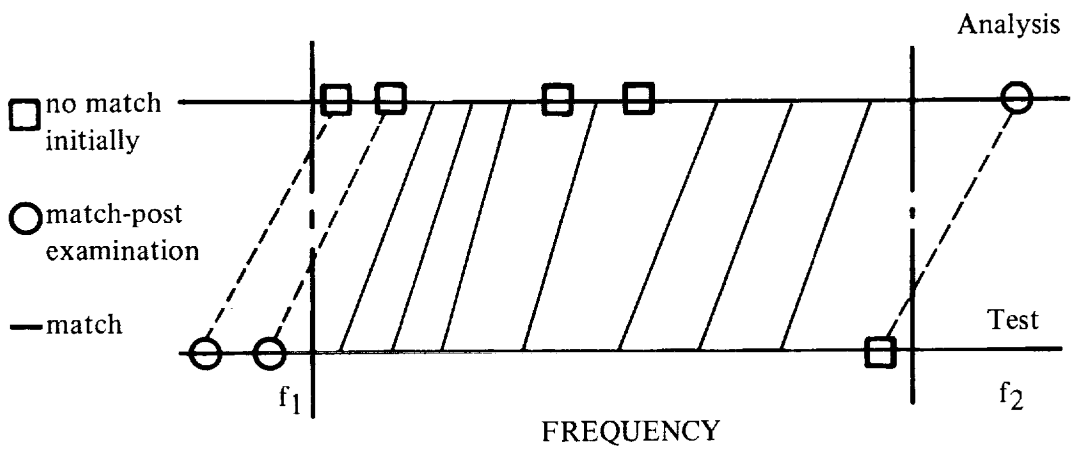


Figure 1

One would be inclined to look above the upper frequency bound of the analytical modes to find a match for the spinster test frequency, and to look below the lower frequency bound of the test modes to find matches for the mateless analytical modes. As for the two unpaired analytical modes in the middle frequency range, there is a possibility that analysis uncovered two modes that test failed to find. It might also be that one grid point stands out in five of the matching pairs of modes as the place where the modes show the greatest difference in their shapes. One might be suspicious of the analysis as having too low a value of mass at that point.

The happy ending to this story might be that the analytical mass defect was found, and the wanting test instrumentation was supplied so that the overall shift in frequency was erased, two middle range test modes were revealed, and 10 out of 11 matches were found in the frequency range on a second trial.

The limited amount of experimental data and the limited access in which to install sensors at interior points are factors that constrain the points at which comparisons can be made. There is a disproportionately large number of points at which analytical data is available for comparisons. The objective will be to bring about useful comparisons within the constraints of available data. The intent is to be able to identify the mode shapes from both of the sources; to say how closely any two mode shapes are related; to say at what frequencies the test mode and the analytical mode of a closely related pair appear; to give the locations at which two modes diverge beyond a threshold; and to isolate the greatest divergence.

A correlary objective is to establish a communication basis between the records from analysis and the records from test. Once such a channel is established, then standardized data can be exchanged by means of a program. For instance, the arrays of generalized mass and generalized stiffness can be passed from analysis to test while the array of modal damping can be passed from test to analysis.

METHOD

Structural properties determined by either analysis or test can properly be regarded as nondeterministic and therefore random. For instance, in analysis the factors liable to random variation are: the interpretation of drawings, the modeling of elastic relations, the representation of joints, the modeling of mass properties, entering of data into the computer, transfer of data into, out of, and within a computer, computer behavior, and debugging methods. There will be as many analytical results as there are analysts. Similarly, variability in testing can arise from the quality of materials in fabrication, the machining of parts, the assembly of components, the mounting of sensors, the calibration of recorders, the reading of records, and the processing of data. The experimental results will be as individual as the number of teams conducting tests. Since it is the intention to compare two random records, a well proven stochastic tool to apply is the correlation function—or particularly—the correlation coefficient. Thus, the criterion for deciding whether two modes are the same can be mediated by how close to unity the correlation coefficient for them is. Statistical methods will be employed to compare test mode shapes with analytical mode shapes. Analytical and test measurements of mode shapes will be taken at the same set of structural locations, so that behavior will be compared at like points. At present, the comparisons in this study will be limited to undamped real modes, as opposed to damped complex modes. A test mode shape and an analytical mode shape will each be considered to be a statistical set of data. The number of test modes to be compared need not be the same as the number of analytical modes, so that all available modes from both sources within a specified frequency range can be considered. The primary tool for making comparisons will be the correlation coefficient.

According to James & James 'Mathematical Dictionary', the definitions of the statistical quantities that will be used in this analysis are as follows:

$$\text{Variance} = \sum_{i=1}^{n} (x_i - x')^2 \, p(x_i) = \sigma^2 \qquad (1)$$

where n is the number of sample points in the statistical file,

$$x' \text{ is the mean} = \frac{1}{n} \sum_{i=1}^{n} x_i \qquad (2)$$

and $p(x_i)$ is the probability of finding the sample x_i in any statistical record. Other statistical quantities will be introduced after discussing those introduced thus far.

The following interpretations will be given to these quantities for purposes of this analysis. Data from each mode is a distinct statistical record. The quantity x_i is the measurement of the departure from the equilibrium position at a given point "i" of a single vibration mode from an eigenvalue analysis. If x_i were measured in time from an undamped oscillation, it is known from the definition of a mode shape that there will be an equal number of positive excursions as negative excursions away from the equilibrium position of a mode. The mean of a time varying sample would be the equilibrium position, so x' would be zero. The x_i's are not taken as time varying samples, but as the normalized envelopes of a finite number of extreme model positions for which the average is definitely not zero. The probability $p(x_i)$ is a quantity which varies from mode to mode. The function $p(x_i)$ for a mode is not known a priori, so it will be estimated by using the scheme of empirical probability density as described in Y. W. Lee's "Statistical Theory of Communications", page 125. The range of x_i depends on how a mode is normalized. For example, if a mode is normalized to its maximum component, the range is $-1 < x_i < +1$. For normalization according to a particular component or to mass, the ranges can be greater or less than unity. The range is divided into equal increments, Δx. Modal data is scanned for how many of the modal displacements fall within each increment; i.e, the number N_i in each increment Δx_i. The frequency ratio is constructed for each increment, by dividing the count in each increment N_i by the total number of modal data points n: N_i/n. According to Jacob Bernoulli's theorem on probability, the quantity $\frac{N_i/n}{\Delta x}$ stays within an ϵ neighborhood of $p(x_i)$ as n gets large and Δx grows small. It will be assumed that the number of samples of a mode shape n is sufficiently large to justify the use of the quantity $\frac{N_i/n}{\Delta x}$ as a reasonable approximation to the probability density function $p(x)$ for use in this algorithm. The probability that a model sample point "h" will fall within the range of

$$x < h < x + \Delta x \text{ is } \int_x^{x+\Delta x} p(x)dx$$

Because of the discretized nature of the available data the integral will be approximated in incremental form by

$$P(x_i < h < x_i + \Delta x) \approx \frac{N_i/n}{\Delta x}(x_i + \Delta x - x_i) \approx N_i/n \qquad (3)$$

On occasion, a fault in a sensor or a recorder will arise which voids data for some point. This renders the probability null for finding data at that point. Instead of admitting a zero probability, this analysis will reduce the sample space "n" to the corresponding number of data points available in both analysis and test.

With the above provisions for this analysis, the statistical quantities reduce to:

Variance

$$\sigma^2 = \sum_{i=1}^{n} (x_i - x')^2 \; N_i/n \qquad (4)$$

Standard Deviation

$$\sigma = +\sqrt{\sigma^2} = + [\sum_{i=1}^{n} (x_i - x')^2 \; N_i/n]^{1/2} \qquad (5)$$

R.M.S (Root Mean Square)

$$\rho = [1/n\sum x_i^2 \; N_i/n]^{1/2}. \text{ If } x' = 0 \text{ then } \rho = \sigma.$$

Correlation Coefficient between two statisitical files $x_i(a)$ and $x_i(e)$ is:

$$r_{ab} = \frac{\sum_{i=1}^{n} [x_i(a) - x'_a] \; \rho_a(x_i)][x_i(e) - x'_e] \; \rho_e(x_i) \cdot \rho_{ae}(x_a, x_e)_i}{\sqrt{\sum_{i=1}^{n} [x_i(a) - x'_a]^2 \rho_a(x_i) \; \sum_{i=1}^{n} [x_i(e) - x'_e]^2 \rho_e(x_i)}}$$

using $x'_a = x'_e = 0$, and using the frequency ratios for approximating probability causes "r" to simplify to:

$$r_{ab} = \frac{\sum_{i=1}^{n} x_i(a) \, x_i(e) \frac{N_a(x_i)}{n} \frac{N_e(x_i)}{n}}{\sqrt{\sum_{i=1}^{n} x_i(a)^2 \frac{N_a(x)}{n} \sum_{i=1}^{n} x_i(e)^2 \frac{N_e(x)}{n}}} = \frac{\sum_{i=1}^{n} x_i(a) \, x_i(e) \frac{1}{n^2} N_a(x_i) \, N_e(x_i)}{\sigma_a \; \sigma_e}$$

$$r_{ab} = \sum_{i=1}^{n} \frac{x_i(a)}{\sigma_a} \frac{x_i(e)}{\sigma_e} \frac{N_a(x_i) \, N_e(x_i)}{n^2} \qquad (6)$$

Equation (6) can be interpreted as the total effect over the whole mode of multiplying the modal amplitude in the "a" set of data when scaled by its standard deviation, by the modal amplitude in the "e" set of data when scaled by the "e" standard deviation, for every pair of corresponding points, and summed for all n points. This method gives complete freedom to both the analyst and to the experimenter for organizing data in the manner of his choice (so long as it is uniform) and of normalizing to any convenient scheme independent of each other. When each set of data is scaled to its own standard deviation, both are placed on equal footings regardless of how the modal data was originally obtained.

All modes in both sets will be compared with each other, and the correlation coefficient will be computed for every combination.

Relative Deviation

It is instructive to determine where and by how much test and analytical mode shapes differ from one another. The approach here is to scale the modes so as to be highly sensitive to differences. Scaling by the RMS value for each mode will put both modes on such a footing that their differences will show a greater spread than they would if scaled by the variance. Using subscript "a" to indicate quantities belonging to analytical data and "e" to indicate quantities belonging to experimental data, the expression for relative deviation in the i^{th} sample of the j^{th} mode is written as:

$$(\text{Rel. Dev.})_{ij} = \left| \frac{x_i(a)_j}{\rho^a} - \frac{x_i(e)_j}{\rho^e} \right| \tag{7}$$

Relative deviation in a mode is the absolute value of the difference between the analytical modal displacement scaled by its modal RMS, and the experimental modal displacement scaled by its own modal RMS. An option will be given to specify what percentage of spread is to be displayed with the grid point identity. The default value of this spread is 5%.

Three other quantities will be computed which have been found to be useful by practicing test engineers. These quantities are defined in an internal GSFC memorandum written by D. J. Hershfeld dated January 8, 1971, entitled "Fitting Mode Shapes to Experimental Data." According to equation (2) of the memo (when transcribed into notation adapted in this report) the normalizing factor of the j^{th} mode is:

Normalizing Factor

$$C_j = \frac{\sum_{i=1}^{n} x_i(a)_j \, x_i(e)_j}{\sum_{i=1}^{n} x_i(a)_j^2} \tag{8}$$

The standard error, according to equation (5) of the memo in transcribed notation, is:

Standard Error

$$S = \left| \frac{1}{n} \left[\sum_{i=1}^{n} x_i(e)^2 \, N_e(x_i) - \sum_{i=1}^{n} x_i(a)^2 \, N_a(x_i) \right] \right|^{1/2} = \left| \sigma_e^2 - \sigma_a^2 \right|^{1/2} \tag{9}$$

Differences in mode shapes are computed by a common scaling of their absolute values by the standard error:

$$\text{Scaled Difference} = \left| [x_i(a) - x_i(e)]/S \right| \tag{10}$$

The tabulations of input modal arrays will be processed by this program. Listings will be made of modal masses, modal stiffnesses, and modal damping arrays versus frequency.

IMPLEMENTATION

In order to put this statistical correlation method of comparing test and analysis into operation, attention must be given to three distinct operations: a. the generation of the analytical vibration data and the transfer of its results, b. the sensing of the test vibration data and the transfer of its results, and c. the computation of the comparison between test and analysis and the acceptance of data from the two sources.

Analytical vibration data are expected to be implemented with NASTRAN. Output of the results is needed in non-standard form, so an ALTER packet was written to tailor the data and to translate the data to BCD for universal readability. These analytical data need further processing before computation can start and are passed to intermediate processors.

Two assemblies of test data are expected. They are: the list of points where test data were taken and the tabulation of mode shapes and frequencies in simple BCD format. They need to be transferred to files in the primary computer for computation.

The computation according to the theory outlined above is accomplished in a program called STATCORR. It expects input data in particular formats and allows the user to choose from a variety of output options.

Consideration has been given to analytical and test data coming from various sources. The STATCORR program will operate on data independent of its source, so long as it is entered according to a specified format. During this development, the analysis was done with NASTRAN and the tests were performed by the Environmental Test & Integration Branch at GSFC. References to specific analytical operations will pertain to capabilities in NASTRAN.

Provision was made for data coming from several runs, structures with symmetry, different normalizations, and non-uniform samples.

Liaison

At the outset, it is mandatory that comparisons be based on data gathered from identical locations. Liaison between test and analysis personnel prior to either the generation of the NASTRAN model or the instrumentation of the test article must establish the points and the component directions at which data is to be sampled. This set of locations will be referred to as the TESET vector. TESET will govern the location and orientation of test sensors and will govern the entries in the analytical partitioning vector. It is used in tabulating mode shapes from test and analysis, and it is the primary reference within STATCORR for sorting and reporting differences.

Equally important as matching data points is matching boundary conditions. The most difficult experimental boundary to enforce is clamped. The easiest to enact is free-free. Liaison between the test engineer and the analyst will help to uncover any disparity between the test set-up and the analytical boundary constraints.

Provision has been made for using symmetry. If the structure has symmetry that can be taken advantage of, liaison should define whether both test and analysis will collect data on only the basic segment. If so, agreement is needed on the naming of each plane of symmetry. Then, each test mode and each analytical mode must be tagged with SYM # PLAN parameters for the type of symmetry across each plane of symmetry. If only analytical data is based on symmetry, while test data is gathered over the whole structure, liaison is needed for three additional items. Octants of symmetry need to be identified. Each test point must be checked for the existence of an image point of reflection in the basic segment of analysis. Multiply-reflected image points must be provided with clones. A list of points with their individual component numbers belonging to each octant is organized by analysis, but should be double checked by the test engineer before proceeding.

Analysis

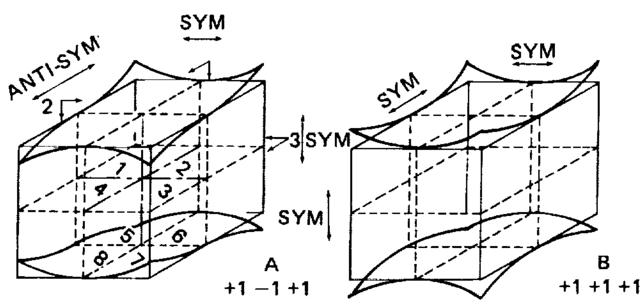
In NASTRAN, there are 5 options for eigenvalue analysis: Determinant, Inverse Power, Givens, Feer, and Hessenberg. All methods are manageable for use in conjunction with STATCORR. There is a problem, however, in transferring NASTRAN eigenvalue results into STATCORR. A DMAP ALTER has been written that manages this transfer. Modal frequencies, mass, and stiffness are delivered as part of the LAMA table in DTI format, using TABPCH. Partitioned mode shapes are delivered as a single matrix in DMI format using OUTPUT3. An explanation of the ALTER packet will follow subsequently. There are some precautions that should be observed with some of the 5 methods. With the Feer method one should check-point the eigenvalue run and exit after the READ module. Results should be examined for the acceptability of the last half of the roots extracted. If negative roots or vectors that fail to meet the orthogonality criterion are found, the PHIA matrix and LAMA table should be partitioned down to only the valid roots immediately after restarting; then one can proceed with the prepared DMAP ALTER route. Sometimes with the Determinant and the Inverse Power methods, several runs may be required before all the roots in a given frequency range are found. Data from any number of separate runs may be combined, but some roots may be repeated in different runs. A post-processor program (MERGE) has been provided to combine eigenvector data from the several runs, detect and eliminate duplicates, and sequence them in ascending order of frequencies. Modes may be normalized according to MASS, MAX, or POINT, but once a decision is made for a method of normalizing, it must be used exclusively for all analytical modes henceforth.

Partitioning is done using the DMAP module PARTN which requires a user supplied partitioning vector. The vector has been arbitrarily named TESET (standing for test set), but this name is imbedded in the DMAP statements of the ALTER packet, so it becomes necessary to use this name. DMI bulk data cards define the vector as a rectangular matrix of a single column, and the number of rows equal to order G of the structural model. This partitioning vector will be designed to serve a double duty. In addition to its duty of partitioning the PHIG matrix, TESET will serve the STATCORR program as a guide for correlating the sequence of test data with analytical data. The device employed to enable this secondary duty is that of constructing the DMI vector in ordered pairs. The first member of this pair is a NASTRAN internal sequence number which corresponds to a component of a test point. The second member of the pair is the Grid Point ID number belonging to the test point. The internal sequence number is actually the row number of the equation of equilibrium for a given degree of freedom as they are organized in NASTRAN's system of matrices. The matrix of eigenvectors, PHIG, is the matrix to be partitioned, so the vector used to do the partitioning has to be the same order as the subject matrix. The final letter of standard NASTRAN matrix names gives its size; so in this case, the partitioning vector must be of order G; or briefly, the vector is G-sized. The format of the DMI input data requires that the first member of the pair is to be an integer, and second member of the pair is to be a real number. Consequently, in implementing this DMAP ALTER the user is required to first list the grid point identification number (ID) as a real number. Just placing a decimal point after the ID number makes it a real number. The internal sequencing used by NASTRAN for matrix operations could be different than that which the analyst originated with his grid point sequencing. The partitioning vector must be organized according to internal sequencing, so it is advisable to include DIAG 21 and 22 and CHKPNT YES in the executive control packet, and then insert an ALTER to EXIT somewhere after the GP4 module. The detailed correspondence between internal and external sequencing for the various sets will be printed when DIAG 21 and 22 are activated. The internal sequence number of every corresponding entry on the DOF list will become the row number of the partitioning vector. The row numbers are required to appear in ascending sequence on the DMI cards.

Symmetry

As a special analytical case, an analyst likes to take advantage of symmetry of a particular structure for the economy and improved definition of modes. Cyclic symmetry is automated within NASTRAN, but reflective symmetry is not. Mode shapes analyzed via cyclic symmetry can automatically be described either for the whole structure, or for just a segment. The only additional processing needed for such modes, before they would be ready for STATCORR, is partitioning. On the other hand, modes analyzed via reflective symmetry require additional processing in order to prepare the modes for STATCORR. Eigenvalue analyses employing reflective symmetry may cause the analytical data to be delivered in the form of several runs, depending on the available combinations of symmetry and anti-symmetry. The post processor program for analytical data, mentioned above in the discussion of DET and INV methods, will operate, as explained before, by combining all modes, eliminating duplicates, and sequencing both frequency and GP list. In addition, the analyst will supply 3 parameters that will act like flags to the STATCORR program for indicating the symmetry status with respect to each of the bounding planes of symmetry. Each of the three parameters can carry the values −1, 0, or +1 for the symmetry status. "0" denotes "absence of symmetry" such as for the face plane of a plate. Positive 1 denotes symmetric behavior about the plane of symmetry. Negative 1 denotes anti-symmetric behavior about the plane of symmetry. Names given to these three parameters are SYM1PLAN, SYM2PLAN, and SYM3PLAN. Values for these symmetry parameters must always be supplied, if modal data is based upon information gathered on the basis of only a basic segment of a structure. If data from both test and analysis are flagged for OK having symmetry, STATCORR tests for the existence of symmetry for one file and anti-symmetry for the other file occurring simultaneously with respect to any one of the three planes. When such an unlikeness occurs, the algorithm immediately declares the correlation coefficient to have the value zero, and suspends calculations for both variance and RMS. Likeness amongst all three parameters between the two sets simultaneously indicates to STATCORR that only the correlation coefficient for the basic segment need be calculated, because it is equal to that for the complete structure. RMS is also the same for the whole structure as it is for the basic segment. However, variance of the whole structure is equal to the number of octants of symmetry multiplied by the variance of the basic segment. Standard deviation of the whole structure is equal to the square root of the number of octants of symmetry multiplied by the standard deviation of the basic segment.

To illustrate how the correlation coefficient has values alternating between zero and that for the basic segment, a pair of free parallelepipeds with 3 planes of symmetry will be used. Modes



involving translations in only one coordinate will provide enough detail to be realistic without obscuring the concepts with complications.

Symmetry parameters SYM1PLAN, SYM2PLAN, and SYM3PLAN for the two solids being compared have the values +1, −1, +1 for solid A and +1, +1, +1 for B. A set of modes conforming to these parameters is diagrammed to create a mental image.

The correlation coefficient between the two structures is

$$r = \frac{\sum_{i=1}^{n} x_i y_i}{[\sum_{i=1}^{n} x_i^2]^{1/2} [\sum_{i=1}^{n} y_i^2]^{1/2}} = \frac{(xy)_1 + (xy)_2 + (xy)_3 + (xy)_4 \ldots\ldots\ldots\ldots (xy)_8}{[x_1^2 + x_2^2 + x_3^2 + \ldots . x_8^2]^{1/2} [y_1^2 + y_2^2 + y_3^2 + \ldots . y_8^2]^{1/2}}$$

where $\sum_{i=1}^{n_s} x_i y_i = (XY)_s$, $\sum_{i=1}^{n_s} x_i^2 = X_s$, and $\sum_{i=1}^{n_s} y_i^2 = Y_s$.

Substitute the equivalent, signed basic segment portion for octants 2 and above.

$$r = \frac{(xy)_1 + (xy)_1 + [(-x)y]_1 + [(-x)y]_1 + (xy)_1 + (xy)_1 + [(-x)y]_1 + [(-x)y]_1}{[8x_1]^{1/2} [8y_1]^{1/2}}$$

$$= \frac{0}{8\sigma_x \sigma_y} = 0.$$

If, however, both A and B were symmetric with respect to all 3 planes of symmetry, the correlation coefficient can be represented in terms of basic segment data as follows:

$$r = \frac{8(xy)_1}{8^{1/2}\,\sigma_x\,8^{1/2}\,\sigma_y} = \frac{8(xy)_1}{8\,\sigma_x\,\sigma_y} = \frac{(xy)_1}{\sigma_x\,\sigma_y} \qquad \text{q.e.d.} \qquad (11)$$

This concept is built into the correlation algorithm of STATCORR. A table of correlation coefficients for various symmetry permutations clearly shows the oscillation between two values.

SYMMETRY PARAMETERS	CORRELATION COEFFICIENT	SYMMETRY PARAMETERS	CORRELATION COEFFICIENT
S O O : x A O O : y	0	S A O : x A A O : x	0
A O O : x S O O : x	0	A A O : x A A O : y	BASIC
A O O : x A O O : y	BASIC	S A S : x S S S : y	0
S O O : x S O O : y	BASIC	S S S : x S S S : y	BASIC
S A O : x S S O : y	0	A S S : x S A A : y	0
S S O : x S S O : y	BASIC	A S A : x A S A : y	BASIC
A S O : x A A O : y	0	S A S : x S A A : y	0
A S O : x A S O : y	BASIC	A A A : x A A A : y	BASIC

Variance. The variance is given by

$$\sigma^2 = \sum_{i=1}^{n} x_i^2$$

$$\sigma^2 = X_1 + X_2 + X_3 + X_4 + X_5 + X_6 + X_7 + X_8$$

which, in terms of equivalent basic segment for the parallepiped, becomes:

$$\sigma^2 = 8\ X_1 = 8\ \sigma_1{}^2, \text{ or } \sigma^2_{whole} = 2^p\ \sigma^2_{basic} \tag{12}$$

where p = the number of planes of reflective symmetry.

Standard Deviation. Similarly,

$$\sigma = (\sigma^2)^{1/2} = (8\ \sigma_1{}^2)^{1/2} = 2\sqrt{2}\,\sigma_1 \text{ or } \sigma_{whole} = 2^{p/2}\,\sigma_{basic}$$

Root Mean Square

$\rho = \sigma/\sqrt{n}$, where m equals the sample points in the basic segment and n= 8 m.

$\rho_{basic} = \sigma_{basic}/\sqrt{m}$, but $\sigma_{whole} = 2\sqrt{2}\ \sigma_{basic}$

$$\rho_{whole} = 2\sqrt{2}\ \sigma_{basic}/\sqrt{8m} = \sigma_{basic}/\sqrt{m} = \rho_{basic} \tag{13}$$

$\rho_{whole} = \rho_{basic}$

Normalization Factor

$$C = \frac{\text{Same numerator as ``r''}}{\sigma_a{}^2}$$

Using the results leading up to equation (11), one can conclude that when there is unlikeness in the symmetry parameters, the numerator is zero, causing C to be null. When there is likeness amongst all of the symmetry parameters,

$$C = \frac{8(XY)_1}{8X_1} = \frac{(XY)_1}{X_1} \tag{14}$$

$$C_{whole} = C_{basic}$$

Standard Error

$$S = [\frac{1}{n}(\sigma_e{}^2 - \sigma_a{}^2)]^{1/2}$$

then $\sigma_e^2 = 8X_1(e)$ and $\sigma_a^2 = 8X_1(a)$. Make these substitutions.

$$S = [\frac{8}{8m}(X_1(e) - X_1(a)]^{1/2} = [\frac{1}{m}(\sigma_e - \sigma_a)_{basic}]^{1/2}$$

$$S_{whole} = S_{basic} \qquad (15)$$

The final alternate of reflective symmetry is the case in which analytical properties are determined by only modeling a basic segment, while test uses the entire structure. The scheme will be to limit the calculations to only those quantities that are necessary. It was discovered during the development of this method that it was not necessary to deploy the basic segment geometry into a mode shape over all 8 octants. By planning ahead, it is possible to arrange that every instrumented point on the test article, when reflected back to the basic segment, will define an image point in the basic segment which is a member of the grid of points used in the analysis. It is possible that several external test points reflect back to a common image point in the basic segment, but in the operation of the processor program, it is necessary to provide a unique point in the basic segment for every instrumented point. A multiply-reflected point violates this requirement. This dilemma is solved by setting up clones of grid points in the NASTRAN bulk data. Thus, the total set of points used by analysis in the basic segment must embrace all of the instrumented points. Those will be of three kinds: direct, image, and cloned points. The direct points correspond to the instrumented points in the basic segment on the test article The image points are those having no corresponding points in the basic segment, and that, when reflected by an appropriate number of times from the basic segment, thus will coincide in ID and location with a unique instrumented point outside of the basic segment on the test article. A cloned point is a grid point in the basic segment with a unique grid point ID number, which has duplicate coordinates of a multiply-reflected direct or image point, but has no structural function. There are as many clones at a point as there are multiplicities of instrumented reflections to a point. Clones are fully constrained points that have no elements connected to them. The point whose coordinates are being cloned is called a parent point. It is usual for the analysis to have more data points than the test data has, so it is necessary to partition the analytical mode shapes down to contain data corresponding to only those points that were instrumented on the test article. Partitioning vector TESET earmarks all direct, image, and cloned points so that when TESET acts on the matrix of analytical mode shapes, PHIG, the modal deflections for instrumented points only will be retained in the matrix PHITE. But PHITE, at this raw stage, is not ready for comparison with test data, because modal data from the basic segment has to be reflected out to the instrumented points with correct signs. The signs of the deflections of instrumented points to be assigned to image and cloned points, depend on their location and the type of symmetry being simulated by an analytical run. A way has been devised to communicate these particulars to the processor program which will enable it to convert PHITE to make it ready for STATCORR. The same parameters that described the type of symmetry for the case of both sources using symmetry will serve equally well for this case. Parameters SYM1PLAN, SYM2PLAN and SYM3PLAN must be supplied so that they will be output to the postprocessor. In order to so communicate a location, a convention must be adapted. The following is the convention used for identifying the octants. Octant # 1 is the basic segment. Octant # 2 is the reflection of the basic segment about the first plane of symmetry. By-pass, momentarily, the definition of octant # 3 and define octant # 4 as the reflection of the basic segment about the second plane of symmetry. Octant #3 lies in between octants 2 and 4. It is a double reflection of the basic segment about the first and second planes of symmetry. Octant # 5 is the reflection of the basic segment about the third plane of symmetry. Rather than define octants 6, 7, and 8 as a series of precise double and triple reflections, it is easier to picture them as lying under octants 2, 3, and 4, just as octant # 5 lies under the basic octant. It remains to set a procedure for reporting the locations of instrumented points. This is done by reorganizing the data, already contained in the TESET vector, into an 8 columned matrix which will be named DOFLIST. Columns one through eight of DOFLIST refer to octant locations one through eight. Direct points are tabulated in ascending sequence of internal DOF number in column one. Image points and parents of cloned points reflected from octant # 2 are tabulated in ascending sequence of internal DOF number in column two. Similarly, tabulations follow for octants 3 through 8. Tabulations are omitted for any octant for which symmetry is absent (i.e., SYM#PLAN is zero). The essential difference between TESET and DOFLIST will be stressed at this juncture. TESET is a vector tabulation of internal degrees of freedom vs. instrumented grid point numbers including clone IDs. DOFLIST is an 8 columned matrix tabulation of internal degrees of freedom vs. all instrumented grid point numbers except that parent IDs are used for clones.

By definition, if SYM2PLAN is equal to −1, the reflection with respect to the second plane is anti-symmetric, and the vertical displacements in octant # 4, as shown in Figure 2, are of the opposite sign from those in the basic segment. Similarly, if SYM3PLAN is equal to +1, the vertical displacements in octant 5 would also be of opposite sign from those of the basic segment. To determine the sign for octants # 3, 6, 7, and 8, the parameters must be used in particular ways. Thus, the sign of displacements in octant # 3 depend on the types of symmetry across both the first and second planes of symmetry. If one were symmetric and the other anti-symmetric, the vertical displacements for octant # 3 would be of opposite sign to the basic. This discussion deals with a given point in the basic segment which is paired with its image point in the reflected octant; therefore all such pairs of displacements are equal in magnitude. Symmetry and anti-symmetry for 3 translational and 3 rotational degrees of freedom will be defined with respect to a plane of symmetry, and then is translated to items of action pertaining to the correlation coefficient under the option of "symmetry for analysis and whole structure for test." Consider, for discussion purposes, that the plane of symmetry is a coordinate plane and, it is oriented vertically. Designate one of the other two coordinate planes to be the horizontal plane, and the remaining one to be transverse.

For the moment, language will be adapted for describing the modal displacements in terms of this convention for the 3 planes. Displacements perpendicular to the plane of symmetry can be described as being towards or away from the plane of symmetry. Displacements perpendicular to the horizontal plane can be described as being up or down. Displacements perpendicular to the transverse plane can be described as back and forth. The goal is to define the behavior of the reflection with respect to the plane of symmetry for each of the 6 displacement components at a Grid Point, for either the symmetric or the anti-symmetric case.

When data is collected on a whole structure with respect to a single fixed coordinate system with coordinate planes imbedded in the frame of the planes of symmetry, it is unaware of the descriptions: twd/away, up/down, back/forth. Therefore, when analytical symmetric data is converted from the basic segment to modal behavior in the seven other octants, it must take into account the way that displacements will appear in the fixed coordinate system of the whole. As an example, according to Table S, the symmetric behavior involving displacements perpendicular to the plane of symmetry will result in the basic and image points both moving synchronously toward or synchronously away from the plane of symmetry. In terms of the coordinate system of the whole, the displacements involve the same coordinate axis. Since they are both moving toward the plane of symmetry, they are moving towards each other; that is, they bear opposite signs.

Table S

IMAGE VS BASIC MOTIONS WITH RESPECT TO PLANES OF SYMMETRY

DIRECTION OF MOTION		RELATIONSHIPS
TRANS ⊥ P/S	Both twd Both away	– Symmetric
	One twd & one away	- AntiSym
ROT'N ⊥ P/S	Δvert/ Δforth: Both Δ up or Both Δdown with Both Δforth	– Symmetric
	One Δup & one Δdown with Both Δforth	– AntiSym
TRANS ⊥ H/P	Both up Both down	–Symmetric
	One up & one down	– AntiSym
ROT'N ⊥ H/P	Δacross/ Δforth: Both Δtwd Both Δaway with Both Δforth	– Symmetric
	One Δtwd & one Δaway with Both Δforth	– AntiSym
TRANS ⊥ T/P	Both forth Both back	– Symmetric
	One forth & one back	– AntiSym
ROT'N ⊥ T/P	Δacross/Δup: Both Δtwd or Both Δaway with Both Δup	– Symmetric
	One Δtwd & one Δaway with Both Δup	– AntiSym

Consequently, adaptation of basic segment data to octant # 2 symmetrical modal behavior, requires that the sign be changed for those components which are perpendicular to the plane of symmetry. Currently, it is recognized that only translational data is measured in test, but Table U will be prepared for sign changes not only for the 3 translational components, but also for the three rotational components. In order to implement this conversion, certain restrictions must be placed on the analytical model. Only the basic coordinate system will be allowed to be used for grid points in the basic segment which correspond to instrumented test points. Its coordinate planes must be parallel to planes of reflective symmetry. To report how the basic coordinate system is oriented to the planes of symmetry, a vector is required to be constructed on DMI cards called IDCORD. The basic coordinate system ID (non-zero) is entered as an integer in the row position, and the translational component which is perpendicular to the 1*st* plane of symmetry is entered as a real number, followed by the translational component, which is perpendicular to the 2*nd* plane of symmetry. The size of the matrix to be entered on the "0" card is: rows = ID # + 1, columns = 1. If a small number is used for the coordinate system ID, it will eliminate the unnecessary processing of zeroes. The card will have the appearance of that shown in Figure 3, page 18.

Table U

ACTION TABLE FOR CONVERTING BASIC SEGMENT DATA
SIGN OF DISPLACEMENT

OCTANT LOCATION OF IMAGE POINT	TYPE & SEQUENCE OF REFLECTION			FOR COMPONENT NO.					
	SYM1	SYM2	SYM3	1 2 3	2 3 1	3 1 2	4 5 6	5 6 4	6 4 5
2	+1			CHNG	NOOP	NOOP	NOOP	CHNG	CHNG
	-1			NOOP	CHNG	CHNG	CHNG	NOOP	NOOP
4		+1		NOOP	CHNG	NOOP	CHNG	NOOP	CHNG
		-1		CHNG	NOOP	CHNG	NOOP	CHNG	NOOP
5			+1	NOOP	NOOP	CHNG	CHNG	CHNG	NOOP
			-1	CHNG	CHNG	NOOP	NOOP	NOOP	CHNG
3	+1	-1		CHNG	NOOP		NOOP	CHNG	
	-1	+1		NOOP	CHNG		CHNG	NOOP	
	+1	+1				NOOP			CHNG
	-1	-1				CHNG			NOOP
6	+1		-1	CHNG		NOOP	NOOP		CHNG
	1		+1	NOOP		CHNG	CHNG		NOOP
	+1		+1		NOOP			CHNG	
	-1		-1		CHNG			NOOP	
8		+1	-1		CHNG	NOOP		NOOP	CHNG
		1	+1		NOOP	CHNG		CHNG	NOOP
		+1	+1	NOOP			CHNG		
		-1	-1	CHNG			NOOP		
7	+1	1	1	CHNG			NOOP		
	-1	+1	+1	NOOP			CHNG		
	+1	-1	+1		NOOP			CHNG	
	1	+1	-1		CHNG			NOOP	
	+1	+1	1			NOOP			CHNG
	-1	-1	+1			CHNG			NOOP

COMP PERP TO TP P/S		
1ST	2ND	3RD
1	2	3
2	3	1
3	1	2

Notice that the octants are not entered in ascending sequence. The scheme used here was to work towards increasing complexity starting with octants involving only a single reflection, then double, and finally triple reflection.

DMI	IDCORD	0	2	1	1	ID + 1	1
DMI	IDCORD	1	ID	PERP 1ST	PERP 2ND		
			(INTEGER)	(REAL)	(REAL)		

Figure 3

What action needs to be taken to convert a component in the basic segment to a modal displacement for an image point in an external octant has been worked out for either type of symmetry, for all six component displacements, and for the three distinct orientations of the basic coordinate system. The action to be taken is, either to change the sign of that component or not. If the sign is to be changed, the notation in the table is CHNG. If the sign is to be left unchanged, the entry in the table is expressed in computer jargon as NOOP, meaning no operation.

Certain components can participate in only certain types of multiple reflections. Null entries are mutually exclusive. The processor program will accomplish the actions tabulated here, so that the analytical mode shapes delivered to the STATCORR program will represent whole structure behavior at those image points corresponding to instrumented points outside the basic segment.

CODES

DMAP ALTER's; (STATDMAP)

In reviewing the design of the DMAP ALTER packet, one sees that its principal function is to transfer important data blocks from the NASTRAN runs to the STATCORR program. It uses the punch file for all output. It manages the partitioning of the eigenvectors to the order of test modes.

A DMAP ALTER packet must be included with each analytical run in order to provide the data sets and parameters necessary to prepare the output properly for operating the STATCORR program. A listing of the ALTER packet is contained in this report on pages 22 through 26. As of this writing, the packet is compatible with NASTRAN Rigid Format Series P. It was intended to be self explanatory by supplementing it with comment statements. It is stored in the user library account ULIB1 in the Design Engineering Branch under the filename DRA0:[NASCAT.UTILITY] STATDMAP.LIS;1. Analysts who wish to implement this ALTER can access this file, then copy and merge it into the executive control of their own jobs. Scanning the left hand margins of STATDMAP, the reader will notice that there are several subdivisions. Each subdivision will be called a packet according to an identifier. Each ALTER packet subdividion is readily distinguished by a title. The first packet is entitled FREQUENCY DATA and uses the margin identifier $F. PARTITIONED MODE SHAPES is the title of the second packet, identified by $P. The third packet is entitled SYMMETRY FOR BOTH, using $S at the margin. Finally, the ANALYTICAL

SYMMETRY ONLY packet uses identifier $W. Packets F and P are always used. When data comes from a basic segment of symmetry for both analysis and test, packets F, P, and S are used. The entire set of ALTER packets is needed when the test data is based on the whole structure, but analysis uses a basic segment only. Packet $F delivers the LAMA table from a NASTRAN eigenvalue execution to the punched output file using the DMAP utility module, TABPCH. Although the entire LAMA table is exported, the preprocessor program selects only three entries for each mode. These are words 5, 6, and 7 in Record 2 of LAMA giving the cyclic frequency, generalized mass, and generalized stiffness, respectively. Packet $P instructs the user how to set up his DMI cards for defining the partitioning vector TESET. Module PARTN employs TESET to partition the matrix of eigenvalues PHIG from the analytical size down to experimental size. Module OUTPUT 3 delivers test-sized eigenvectors PHITE and vector TESET to the punch output file in DMI format.

Packet $S provides a place for the analyst to catalog which of the planes of symmetry he named first, second, and third. There are options for rectangular, cylindrical or spherical coordinate systems, so the user is advised to erase the unused options to avoid the clutter. The primary purpose of packet $S is to define the type of reflective symmetry which was simulated for a given computer run. Each time a run is made with a different set of boundary conditions at the planes of symmetry, the appropriate entries should be made for the SYM # PLAN parameters. These parameters are not used at all in the NASTRAN execution. DMAP is being used as a transfer agent to pass the values of these parameters into the post-processor program in a self-contained package so that none of the pertinent data goes adrift or gets mixed with the wrong set. Module PARAM is set up for the user to input their values, and module PRTPARM transfers them to the output file. All other previous output has been sent to the punch output file, and these parameters should be no exception. This is done by intercepting the print output from PRTPARM just long enough to route it to the FORTRAN logical unit for punch output, and then restoring it. PARAM statement S4 directs the parameters to the punch output, and S7 restores it to print output.

Packet $W instructs the user how to set up his DMI cards for tabulating the image points in the octants for matrix DOFLIST and for the IDCORD card. Module OUTPUT3 delivers the DOFLIST matrix and the IDCORD vector to the punch output file in DMI format. Once again, DMAP is being used strictly as a transfer agent to pass the data from the user to the processor program.

Statement F1 is inserted at position 103 of RF3 after the READ module has output LAMA. It picks up data block LAMA, which is a table, and sends it to the punch output file in DTI format. Parameter 1 is given the BCD value "FN" to use in setting up the continuation cards.

Statement P1 does a row partition on the matrix of eigenvectors, PHIG. It is inserted at R.F. 3 statement # 111 after the matrix PHIG has been checkpointed. The position of the column partitioning vector is left deliberately unspecified, while TESET is entered at the position of the row partitioning vector (RP). TESET has values at internal sequence positions of the G set which correspond to the degrees of freedom of points that were instrumented on the test article. PARTN separates the rows into two sub matrices according to whether there is a value in the partitioning vector for that row or not. The symbolic name of the partition matching the row of RP without values is A_{11}, and that, matching the rows of RP with values is A_{21}. Since our interest is in the A_{21} partition only, all positions of the output data blocks from PARTN are left purged, except the second. The name given to the A_{21} partition is PHITE. Giving a positive value of the parameter SYM in the first position indicates a non-symmetric partition. The positive value given to parameter, TYPE, in the second position indicates that the output partitions are real single precision matrices. PHIG is a real single precision matrix so any of its partitions will be the same. The value 2 for the *3rd* and *4th* positions indicates that their forms will be general rectangular.

Statement P2 picks up data blocks PHITE and TESET and writes them on the punch output file in DMI format. Parameter P1 in the first position is negative to indicate that data being output should be sent to the FORTRAN logical unit. BCD value, MOD, is assigned to the second parameter for use in setting up continuation cards for the PHITE matrix, and BCD value, DOF, is assigned to the third parameter for use in setting up continuation cards for the TESET vector.

The explanation for why ALTER 125 was selected for inserting packet P will be delayed until after the explanation of statement S8. Statements S1, S2, and S3 define parameters SYM#PLAN through the module PARAM using logical operator MPY (multiply). In each instance, variable parameter IN1 is multiplied by the constant 1 to produce the output value. In effect, the value given to the variable parameter IN1 is the value of the output.

Statement S4 uses the module PARAM to change the value of the parameter which controls the printing of NASTRAN output (OUTTAP) that is located in the system common block (SYSTEM) of NASTRAN. The purpose of S4 is to change the OUTTAP assignment from the value of the FORTRAN logical unit for printing to the value of the FORTRAN logical unit for punching. The logical operator SYST uses the value for the *3rd* parameter to catalog the sequence number of of the parameter in the system common block. Referring to Section 2.4.1.8 of the Programmer's Manual, one finds that the sequence number for parameter OUTTAP is 2, so 2 is entered as the value for the third parameter. SYST uses the value of the *4th* parameter to catalog the assignment number of the FORTRAN logical unit. These FORTRAN assignments are arbitrary and depend upon the decision of the systems programmer at the time that NASTRAN is installed on a computer. The punch output was assigned to FORTRAN logical unit 77 at the time that NASTRAN was installed on the computer used for developing this method, so in order to direct the output data blocks of a module under the management of system common block, named OUTTAP, to go to the punch buffer, 77 is entered as the value for the 4th parameter. The BCD value DUM given to the *2nd* parameter is meaningless, because this parameter is not used in the operation of the logical operator SYST.

Statements S5, S6, and S7 involve module PRTPARM. The value 0 assigned to the *1st* parameter enables the module to operate as a parameter printer. The particular BCD values assigned to the *2nd* parameter discriminate amongst all possible parameters and will select only those having the indicated BCD names.

Statement S8 uses the module PARAM in the same sense as in statement S4 to assign the FORTRAN logical unit 6 to the system common block parameter OUTTAP. Making this temporary reassignment of a major regulation item should be treated gingerly so as not to interfere with normal operations. Rapid restorationn is imperative. Care must be exercised when PARAM logical operator SYST is put into operation. Since the printer output block is involved, care must be taken to ensure that anything destined for the print buffer has been already sent there, so that only those things that are governed by parameter OUTTAP during this temporary assignment are routed to the punch buffer. Statement 125 occurs after module OFP has directed OEIGS, LAMA, and DQM1 to the print buffer and before new quantities from SDR2 are being readied for printing. Thus diverting OUTTAP temporarily to punch at statement 125 does not endanger the printing of any important quantities.

Statement W1 is inserted at position 128 to isolate it from those in packet S, as well as, to make sure that it would not be caught in a bypass caused by a conditional jump. It picks up the data blocks delivered by the analyst in the bulk data in DMI format, and writes them on the punch output file in DMI format once again. Parameter P1 in the *1st* position is negative to indicate that the data output should be sent to the FORTRAN logical unit. The BCD value, LIS, is assigned to the *2nd* parameter, and FRM to the *3rd* parameter for use in setting up the continuation cards.

```
$------------STATCORR PREP----------------------------------------
$------------DMAP ALTER PACKET------------------------------------
$
$
$
$------------FOR USE WITH RIGID FORMAT SERIES P---------------------
$
$
$
$
ALTER   103
$ THIS DMAP ALTER PACKET IS DESIGNED TO PROVIDE OUTPUT TO PROCESSORS
$ WHICH IN TURN PREPARE THE ANALYTICAL INPUT TO THE STATISTICAL COR-
$ RELATION PROGRAM ***STATCORR*** FOR COMPARING ANALYTICALLY DERIVED
$ MODAL DATA WITH EXPERIMENTALLY DERIVED MODAL DATA.
$ SUBPACKETS ***F*** AND ***P*** WILL ALWAYS BE USED.  SUBPACKET ***S***
$ IS DESIGNED FOR USE WHEN BOTH ANALYTICAL AND EXPERIMENTAL DATA ARE
$ REPORTED FOR ONLY A SINGLE SEGMENT OF REFLECTIVE SYMMETRY IN ONE,
$ TWO, OR THREE COORDINATE DIRECTIONS.  SUBPACKET ***W***IS DESIGNED
$ FOR THE CASE WHEN REFLECTIVE SYMMETRY IS USED WITH ANALYSIS BUT NOT
$ WITH TEST.
$
$
$
$*******************FREQUENCY DATA*************************************
$
$
$F1
TABPCH    LAMA,,,,//FN $
$
$
$F WORD 5 IN EACH ENTRY OF RECORD 2 OF THE LAMA TABLE GIVES THE CYCLIC
$F FREQUENCY FOR A MODE.
$F WORD 6 IN EACH ENTRY OF RECORD 2 OF THE LAMA TABLE GIVES THE GENER-
$F ALIZED MASS FOR A MODE.
$F WORD 7 IN EACH ENTRY OF RECORD 2 OF THE LAMA TABLE GIVES THE GENER-
$F ALIZED STIFFNESS FOR A MODE.
$
$
$
$
$*****************PARTITIONED MODE SHAPES****************************
$
$
$
$
$P
ALTER  111
$$P
$P PHIC IS PARTITIONED TO THAT SET OF FREEDOMS WHICH ARE TO BE
$P INSTRUMENTED ON THE TEST ARTICLE.
$P
$P THE PARTTIONING VECTOR TESET IS INPUT USING DMI.  THE PRECEDING
$P EIGENVALUE RUN SHOULD INCLUDE DIAG 21,22 TO GIVE THE CORRELATION
$P BETWEEN THE INTERNAL D.O.F.'S AND THE EXTERNAL NUMBERING.  THE
$P PARTITIONING VECTOR IS ORGANIZED ACCORDING TO INTERNAL SEQUENCING.
$P MODULE PARTN IS ORGANIZED TO DO A ROW PARTITION AMONGST THE DEGREES
$P OF FREEDOM IN ORDER TO PRESENT A DESCRIPTION OF THE MODAL BEHAVIOR
$P ONLY IN TERMS OF THE INSTRUMENTED D.O.F.'S.
$P
$P THE ZERO CARD OF DMI HAS THESE ENTRIES:  FIELD 1, DMI; FIELD 2, TESET;
```

```
$P FIELD 3,ZERO
$P FIELD 4, 2--TO INDICATE THAT THE MATRIX BEING BUILT IS RECTANGULAR;
$P FIELDS 5 & 6, 1--TO INDICATE SINGLE PRECISION IN AND SINGLE PRECISION
$P OUT; FIELD 7, BLANK; FIELD 8, G-SIZE; FIELD 9, 1--TO INDICATE A
$P SINGLE COLUMN OUTPUT.
$P $P P
$P THE NEXT DATA CARD OF DMI HAS THESE ENTRIES:  FIELD 1, DMI; FIELD 2,
$P TESET; FIELD 3, 1--TO INDICATE DATA FOLLOWING PERTAINS TO COLUMN 1
$P   OF THE OUTPUT MATRIX;
$P FIELD 4, THE INTERNAL SEQUENCE NUMBER, IN INTEGER FORMAT, OF
$P THE FIRST INSTRUMENTED D.O.F. ACCORDING TO THE DIAG 21 TABULATION;
$P FIELD 5, THE EXTERNAL GRID POINT ID NUMBER CORRESPONDING TO THE
$P FIRST DOF.  IT IS TO BE WRITTEN AS A REAL NUMBER, I.E. GP ID
$P FOLLOWED BY A PERIOD.
$P FIELD 6, THE SECOND INSTRUMENTED D.O.F. PER DIAG 21; INTEGER.
$P FIELD 7, THE EXTERNAL GRID POINT ID NUMBER CORRESPONDING TO THE
$P SECOND DOF. IT IS TO BE WRITTEN AS A REAL NUMBER.
$P CONTINUE IN A SIMILAR MANNER UNTIL ALL INSTRUMENTED POINTS HAVE
$P BEEN ENTERED. IN THE CASE OF SUBPACKET ***W*** THE GRID POINT
$P NUMBERS INCLUDE CLONES AND IMAGES.
$P1
PARTN      PHIG,,TESET/,PHITE,,/C,N,+1/C,N,1/C,N,2/C,N,2 $
$P
$P MATRIX PHITE HAS BEEN PARTITIONED FROM THE PHIG DATA BLOCK.  PHITE
$P CONTAINS MODAL DATA FOR ONLY THOSE DOF THAT WERE INSTRUMENTED. $
$P
$P2
OUTPUT3    PHITE,TESET //C,N,-1/C,Y,N1=VCT /C,Y,N2=DOF $
$P
$P PHITE AND TESET ARE OUTPUT IN DMI FORMAT  AS PUNCH FILES.
$P IF THE USER WANTS THE
$P DMI CARD IMAGES TO BE WRITTEN TO A DISK FILE INSTEAD OF BEING PUNCHED,
$P THE USER MUST SUPPLY THE APPROPRIATE JOB CONTROL LANGUAGE STATEMENTS.
$
$
$***************************SYMMETRY FOR BOTH****************************
$
$
$S THE ANALYST SHOULD SUBSTITUTE THE PROPER ENTRIES FOR EACH OF THE
$S VARIABLES REPRESENTED IN *X*X*X*X* FORMAT IN THE NEXT PARAGRAPH.
$
$
$
$
$S
$S IF THE ANALYSIS IS TO BE CONFINED TO ONLY ONE SEGMENT OF REFLECTIVE
$S SYMMETRY AND SIMULTANEOUSLY IF TEST RESULTS WILL ALSO BE REPORTED
$S FOR ONLY ONE SEGMENT, THEN THE TYPE OF MODES BEING PRODUCED IN EACH
$S RUN MUST BE INDICATED IN THE FOLLOWING THREE PARAMETER STATEMENTS.
$S FOR RECORD PURPOSES, THE FIRST PLANE OF SYMMETRY IS PERPENDICULAR
$S TO THE *A*A*A*A*  AXIS AND CUTS THAT AXIS AT *A*A*A*A* (RECTANGULAR),
$S OR IS AN R:Z PLANE THRU THETA = *T*T*T*T* , PHI = 0 (CYLINDRICAL),
$S OR IS A PLANE THRU THETA = 0 , PHI = 0 (SPHERICAL).
$S THE SECOND PLANE OF SYMMETRY IS PERPENDICULAR TO THE *B*B*B*B* AXIS
$S AND CUTS THAT AXIS AT *B*B*B*B* (RECTANGULAR), OR IS AN R:Z PLANE
$S THRU THETA = [*T*T*T*T* + PI/2] , PHI = 0(CYLINDRICAL), OR IS A PLANE
$S THRU  THETA = PI/2, PHI = 0 (SPHERICAL).
$S THE THIRD PLANE OF SYMMETRY IS PERPENDICULAR TO THE *C*C*C*C* AXIS
$S AND CUTS THAT AXIS AT *C*C*C*C* (RECTANGULAR), OR IS PERPENDICULAR
$S TO THE Z AXIS AND CUTS THAT AXIS AT Z = *V*V*V*V* (CYLINDRICAL),
$S OR IS A PLANE AT PHI = PI/2 FOR ALL THETA (SPHERICAL).
```

```
$S
$
$S
$S THE DESIGNATION FOR A SYMMETRICAL REFLECTION W.R.T. AN AXIS IS +1;
$S ENTER IN1 = +1 ON THE PARAM CARD IN PLACE OF XXX.
$S THE DESIGNATION FOR ANTI-SYMMETRIC REFLECTION W.R.T. AN AXIS IS -1;
$S ENTER IN1 = -1 ON THE PARAM CARD IN PLACE OF XXX.
$S THE DESIGNATION FOR ABSENCE OF SYMMETRY WITH RESPECT TO AN AXIS IS 0;
$S ENTER IN1 = 0 ON THE PARAM CARD IN PLACE OF XXX.
$
$S
$S$ALTER   125
$S1
$
$
$
$S VALUES OF SYM*PLAN WILL CHANGE EVERY TIME THE CONSTRAINTS
$S OF SYMMETRY CHANGE.
$
$
$
$S1$PARAM      //C,N,MPY/V,Y,SYM1PLAN/V,N,IN1=XXX/V,N,IN2=1  $
$S
$S THIS STATEMENT SAYS WHETHER THE MODE SHAPE IS SYMMETRIC OR ANTI-
$S SYMMETRIC OR HAS NO SYMMETRY WITH RESPECT TO PLANE 1.
$S
$S2
$S2$PARAM      //C,N,MPY/V,Y,SYM2PLAN/V,N,IN1=XXX/V,N,IN2=1 $
$S
$S THIS STATEMENT TELLS WHAT KIND OF SYMMETRY PERTAINS TO PLANE 2.
$S
$S3
$S3$PARAM      //C,N,MPY/V,Y,SYM3PLAN/V,N,IN1=XXX/V,N,IN2=1 $
$S
$S THIS STATEMENT TELLS WHAT KIND OF SYMMETRY PERTAINS TO PLANE 3.
$
$S4
$S4$PARAM      //C,N,SYST/V,N,DUM/C,N,2/C,N,77 $
$S
$S CHANGES THE FORTRAN LOGICAL UNIT FOR THE SYSTEM COMMON DATA BLOCK,
$S CALLED OUTPUT, FROM ITS USUAL VAX VALUE
$S OF 6 TO THE VAX VALUE RESERVED FOR PUNCH WHICH IS 77,SO THAT IT CAN
$S BE WRITTEN TO THE SAME DISK FILE AS THE CHKPNT DICTIONARY AND THE
$S OUTPUT3 FILES.
$S
$S5
$S5$PRTPARM    //C,N,0/C,N,SYM1PLAN $
$S
$S6
$S6$PRTPARM    //C,N,0/C,N,SYM2PLAN $
$S
$S7
$S7$PRTPARM    //C,N,0/C,N,SYM3PLAN $
$S
$S8
$S8$PARAM      //C,N,SYST/V,N,RUM/C,N,2/C,N,6 $
$S
$S RESTORES THE FORTRAN LOGICAL UNIT FOR OUTPUT BACK TO THE VAX VALUE
$S OF 6 NOW THAT THE NEED FOR ASSIGNING IT TO 77 IS OVER.
$
$
```

```
$
$** ***************ANALYTICAL SYMMETRY ONLY**************************
$W
$W USE SUBPACKET **W** IF EXPERIMENTAL DATA COMES FROM THE WHOLE STRUCTURE
$W BUT ANALYTICAL DATA COMES FROM A BASIC SEGMENT OF SYMMETRY.  THE
$W ANALYTICAL BASIC SEGMENT WILL CONTAIN POINTS CORRESPONDING TO THOSE
$W TEST POINTS IN THE BASIC SEGMENT.  IN ADDITION THE ANALYTICAL BASIC
$W SEGMENT WILL CONTAIN POINTS WHICH ARE REFLECTED IMAGES OF ALL TEST  POINTS
$W WHICH ARE LOCATED OUTSIDE OF THE BASIC SEGMENT.  THE PARTITIONING
$W VECTOR ***TESET*** WILL INCLUDE THE DIRECT POINTS PLUS ALL CLONED AND
$W IMAGE POINTS.                        ALL REFLECTED POINTS
$W MUST BE TALLIED FOR ***STATCORR*** TO PROCESS THE DATA
$W CORRECTLY.  THIS TALLY IS COMMUNICATED IN THE FORM OF AN EIGHT
$W COLUMNED MATRIX CALLED ***DOFLIST***
$W
$W
$W
$W
$W WARNING.            WARNING.    BE SURE THAT CLONED GRID POINTS
$W ARE INCLUDED IN THE BULK DATA AND ARE FULLY CONSTRAINED.
$W
$W
$W
$W
$W EACH OF THE 8 COLUMNS IS RESERVED FOR A PARTICULAR OCTANT.  THESE
$W ARE THE DEFINITIONS OF THE OCTANTS.
$W OCTANT 1 IS THE BASIC SEGMENT.  OCTANT TWO IS THE REFLECTION OF THE
$W BASIC SEGMENT ABOUT SYM1PLAN.  OCTANT FOUR IS THE REFLECTION OF THE
$W BASIC SEGMENT ABOUT SYM2PLAN.  OCTANT THREE IS IN BETWEEN OCTANTS
$W TWO AND FOUR.  OCTANT FIVE IS THE REFLECTION OF THE BASIC SEGMENT
$W ABOUT SYM3PLAN.  OCTANTS SIX, SEVEN, AND EIGHT ARE ROTATED AWAY
$W FROM OCTANT FIVE IN THE SAME SENSE THAT OCTANTS TWO, THREE, AND
$W FOUR ARE ROTATED AWAY FROM OCTANT ONE.
$W DATA FOR TALLYING THE IMAGE POINTS REFLECTED FROM EACH OCTANT IS
$W ORGANIZED IN DMI INPUT.  COLUMN ONE IS RESERVED FOR LISTING ALL
$W POINTS PLUS COMPONENT DIRECTIONS WHICH ARE COMPANIONS
$W OF THE INSTRUMENTED POINTS IN THE
$W BASIC SEGMENT.  COLUMN TWO IS RESERVED FOR LISTING ALL UNIQUE
$W POINTS OR PARENTS OF CLONES
$W IN  THE BASIC SEGMENT WHICH ARE REFLECTIONS OF INSTRUMENTED
$W POINTS FROM OCTANT #2.  SIMILARLY, COLUMN THREE IS RESERVED FOR
$W LISTING ALL UNIQUE POINTS OR PARENTS OF CLONES IN THE BASIC SEGMENT
$W WHICH ARE REFLECTIONS OF
$W INSTRUMENTED POINTS FROM OCTANT #3.  COL 4 LIST FOR OCTANT 4 AND
$W SO FORTH THRU COL 8 FOR OCTANT 8.
$W EXAMPLES OF EACH OF THE NINE DMI LOGICAL CARDS FOR THIS MATRIX
$W EXPRESSED IN FREE FIELD FORMAT ARE:
$W DMI,DOFLIST,0,2,1,1,BLANK,G-SIZE,8,BLANK
$W DMI,DOFLIST,1,INT DOF #,GP #,INT DOF #,GP #,INT DOF #,GP #,CONT'N
$W DMI,DOFLIST,2,INT DOF #,GP #,INT DOF #,GP #,INT DOF #,GP #,CONT'N
$W DMI DOFLIST,3,INT DOF #,GP #,INT DOF #,GP #,INT DOF #,GP #,CONT'N
$W DMI     ETC FOR COLUMNS 4 THRU 8.
$W NOTICE THAT THE CONTENT OF CARDS FROM COLUMN ONE ET SEQ. ARE
$W PAIRED QUANTITIES.  THE FIRST OF THE PAIR IS THE ROW NUMBER
$W WHICH IS THE INTERNAL DEGREE OF FREEDOM NUMBER WRITTEN AS AN
$W INTEGER.  THE SECOND OF THE PAIR IS A REAL NUMBER  FORM  OF
$W THE GRID POINT ID CORRESPONDING TO THE INTERNAL DOF .
$W   PAIRS FOR A GIVEN COLUMN
$W OVERFLOW ONTO CONTINUATION CARDS UNTIL ALL POINTS FOR AN OCTANT
$W ARE TALLIED.  DATA FOR THE SUCCEEDING COLUMN CONSTITUTE A NEW
$W LOGICAL CARD STARTING WITH THE COLUMN NUMBER THEN FOLLOWED BY ITS
```

```
$W SUBSEQUUENT PAIRS.
$W
$W
$W
$W
$W A SECOND DMI VECTOR NAMED ***IDCORD*** IS PUT INTO THE BULK DATA
$W TO SPECIFY THE ORIENTATION OF BASIC COORDINATE PLANES WITH THE
$W PLANES OF SYMMETRY.  VALUE #1 IS THE BASIC COMPONENT THAT IS
$W PERPENDICULAR TO THE 1ST PLANE OF SYMMETRY.  VALUE #2 IS THE
$W BASIC COMPONENT THAT IS PERPENDICULAR TO THE 2ND PLANE OF
$W SYMMETRY.  SET VALUE #2 = 0.0 IF THERE IS ONLY ONE PLANE OF
$W SYMMETRY.  NO ENTRY IS MADE FOR THE THIRD COMPONENT REGARDLESS
$W OF THE EXISTENCE OF A THIRD PLANE OF SYMMETRY.
$W
$W
$W
$W W$ALTER 126
$W1
$W1$OUTPUT3     DOFLIST,IDCORD//C,N,-1/C,Y,LIS/C,Y,FRM  $
$W
$W POST PROCESSOR WILL APPLY THE APPROPRIATE SIGN TO THE
$W REFLECTED DISPLACEMENTS
$W ACCORDING TO THE COMBINATION OF IN1 PARAMETERS ENTERED IN STATEMENTS
$W S1, S2, AND S3.
$
ENDALTER
```

MERGE

In case eigenvectors have been obtained from several disjoint NASTRAN runs, it is possible to combine these vectors into a single set for use with STATCORR. Probably some overlap of frequency ranges would occur with consequent duplication of vectors. The pre-processor program MERGE accepts files of eigenvectors from several runs and discards all duplicates. It reassembles the resulting into a sequence according to ascending frequencies and delivers them as a single file. MERGE operates interactively and the user responds to the on-line prompts.

CTGB

MERGE:

FOR:2

CTGB

```
C       PROGRAM MERGE
C
C       THIS PROGRAM WILL MERGE TWO MATRIX FILES WITH THE SAME NUMBER
C       OF COLUMNS INTO A SINGLE MATRIX WITH THE SAME NUMBER OF COL-
C       UMNS AND A TOTAL NUMBER OF ROWS EQUAL TO THE SUM OF THE NUMBER
C       OF ROWS IN THE ORIGINAL TWO MATRICES.
C       A CORRESPONDING COLUMN IN EACH MATRIX (NOT NECESSARILY THE SAME
C       COLUMN NUMBER) MUST BE IN ASCENDING SORT, AND A SORTED MERGE WILL
C       BE DONE BASED ON THE RESPECTIVE CORRESPONDING COLUMNS.
C       IF DESIRED, DUPLICATED ROWS WILL BE DELETED; A MAXIMUM RELATIVE
C       DIFFERENCE (EPSILON) MAY BE SPECIFIED FOR DEFINING "EQUALITY";
C       OR A DEFAULT VALUE MAY BE USED.
C       IF DESIRED, ROWS CONTAINING TERMS BELOW A MAXIMUM THRESHOLD VALUE
C       WILL BE DELETED; THE THRESHOLD VALUE MAY BE SPECIFIED, OR A
C       DEFAULT VALUE MAY BE USED.
C       IF DESIRED, ROWS CONTAINING TERMS ABOVE A MINIMUM CEILING VALUE
C       WILL BE DELETED; THE CEILING VALUE MAY BE SPECIFIED, OR A
C       DEFAULT VALUE MAY BE USED.
C       ADDITIONAL MATRIX PAIRS THAT ARE ASSOCIATED RESPECTIVELY WITH THE
C       ORIGINAL MATRIX PAIR MAY ALSO BE MERGED TOGETHER, BASED ON THE
C       SORTED MERGE OF THE ORIGINAL PAIR.  IN THIS CASE, COLUMNS OF AN
C       ASSOCIATED MATRIX MUST CORRESPOND TO ROWS OF THE ORIGINAL MATRIX
C       WITH WHICH IT IS ASSOCIATED. ANY DELETED DUPLICATE,  BELOW-
C       THRESHOLD, OR ABOVE-CEILING ROWS OF THE ORIGINAL MATRIX WILL
C       RESULT IN CORRESPONDINGLY DELETED COLUMNS OF THE ASSOCIATED
C       MERGED MATRIX.
```

```
            PROGRAM MERGE
      C
            CHARACTER*8  MATRIX1, MATRIX2
            CHARACTER*1  YESNO
            REAL  SORTLIST(2,100)
            INTEGER  MTXKEY(200), ROWKEY(200)
      C
            DO 100 ISET=1,1000000
      C          ASK IF ANOTHER SET IS DESIRED
                    TYPE 1000
                    ACCEPT 2000, YESNO
                    IF (YESNO .NE. 'Y')  GOTO 150
      C          REQUEST NAMES & OPEN MATRIX FILES (LOG. UNIT NOS. 1&2)
                    CALL GETKEYMTX(MATRIX1,MATRIX2,KROW1,KROW2,
           *                                           KCOL1,KCOL2,*100)
      C          REQUEST KEY COLUMNS FOR SORTED MERGE, & READ THEM INTO ARRAY
                    CALL GETSORTCOL(MATRIX1,MATRIX2,KROW1,KROW2,
           *                                           KCOL1,KCOL2,SORTLIST)
      C          GENERATE KEY LISTS FOR MERGING
                    CALL KEYLIST(SORTLIST,KROW1,KROW2,MTXKEY,ROWKEY)
      C          SET TOTAL NUMBER OF ROWS (PRIOR TO DELETIONS)
                    KOUT = KROW1 + KROW2
      C          CULL DUPLICATE VALUES
                    CALL DELDUPES(SORTLIST,MTXKEY,ROWKEY,KROW1,KROW2,KOUT)
      C          CULL BELOW-THRESHOLD VALUES
                    CALL DELTHRESH(SORTLIST,MTXKEY,ROWKEY,
           *                                 KROW1,KROW2,KOUT,*100)
      C          CULL ABOVE-CEILING VALUES
                    CALL DELCEILING(SORTLIST,MTXKEY,ROWKEY,
           *                                 KROW1,KROW2,KOUT,*100)
      C          MERGE KEY MATRICES INTO OUTPUT MATRIX & CLOSE MATRIX FILES
                    CALL MERGEKEY(MTXKEY,KOUT)
      C          MERGE ASSOCIATED MATRIX PAIRS
                    CALL ASSOCIATE(MTXKEY,KROW1,KROW2,KOUT)
      100        CONTINUE
      150   STOP  'END OF MATRIX SET REQUESTS'
      C
      1000  FORMAT(/,' MERGE ANOTHER KEY MATRIX SET? (Y OR N): ',$)
      2000  FORMAT(A1)
            END
```

```
      SUBROUTINE ASSOCIATE(MTXKEY,KROW1,KROW2,KOUT)
C
C     MERGE ASSOCIATED MATRIX PAIRS
C
      INTEGER MTXKEY(200)
      CHARACTER*1 YESNO
C
      DO 100 IASSOC=1,1000000
C        ASK IF ANOTHER ASSOCIATED MATRIX PAIR IS DESIRED
            TYPE 1000
            ACCEPT 2000, YESNO
            IF (YESNO .NE. 'Y')  GOTO 150
C        OPEN ASSOCIATED MATRIX PAIR (LOGICAL UNIT NOS. 1&2) AND
C        CHECK FOR VALID DIMENSIONS.
            CALL GETASCMTX(KROW1,KROW2,MROW,NCOL,*100)
C        MERGE ASOCIATED PAIR INTO OUTPUT MATRIX FILE & CLOSE FILES
            CALL MERGEASC(MTXKEY,KOUT,MROW,NCOL)
100      CONTINUE
C     NO MORE ASSOCIATED MATRIX REQUESTS
150      TYPE *, 'END OF ASSOCIATED MATRIX REQUESTS'
         RETURN
C
1000  FORMAT(/,' MERGE ANOTHER ASSOCIATED PAIR OF MATRICES? (Y OR N): '
     *        ,$)
2000  FORMAT(A1)
      END
```

```
      SUBROUTINE DELDUPES(SORTLIST,MTXKEY,ROWKEY,KROW1,KROW2,KOUT)
C
C     DELETE ROWS WHERE THE MERGING TERM IS A DUPLICATE (WITHIN
C     RELATIVE DIFFERENCE EPSILON) OF THE NEXT TERM.
C
      INTEGER MTXKEY(200), ROWKEY(200)
      REAL SORTLIST(2,100)
      CHARACTER*1 YESNO
C
C     ASK IF DUPLICATE DELETION IS DESIRED
          TYPE 1000
          ACCEPT 2000, YESNO
          IF (YESNO .NE. 'Y')  RETURN
C     SET DEFAULT EPSILON & REQUEST INPUT VALUE TO REPLACE IT
          EPS = 1.0E-5
          TYPE 3000, EPS
          ACCEPT *, EPS
C     CULL OUT DUPLICATE VALUES
          KOUT = KROW1 + KROW2
          DO 100 I=1,KROW1+KROW2-1
              VTHIS = SORTLIST(MTXKEY(I),ROWKEY(I))
              VNEXT = SORTLIST(MTXKEY(I+1),ROWKEY(I+1))
              DIFF = ABS(VNEXT-VTHIS)
C             DETERMINE RELATIVE DIFFERENCE
                  IF (DIFF .EQ. 0.0)  THEN
                      RELDIFF = 0.0
                  ELSE
                      RELDIFF = DIFF/MAX(ABS(VNEXT),ABS(VTHIS))
                  END IF
              IF (RELDIFF .LE. EPS)  THEN
                  MTXKEY(I) = -MTXKEY(I)
                  KOUT = KOUT - 1
                  END IF
100           CONTINUE
      RETURN
C
1000  FORMAT(/,' DELETE DUPLICATE VALUES? (Y OR NO): ',$)
2000  FORMAT(A1)
3000  FORMAT(6X,'ENTER VALUE FOR EPSILON. "," FOR DEFAULT ',
     *       1PE8.1,' :  ',$)
      END
```

```
      SUBROUTINE DELTHRESH(SORTLIST,MTXKEY,ROWKEY,KROW1,KROW2,KOUT,*)
C
C     DELETE ROWS WHERE THE MERGING TERM IS BELOW A GIVEN THRESHOLD
C     VALUE.
C
      INTEGER MTXKEY(200), ROWKEY(200)
      REAL SORTLIST(2,100)
      CHARACTER*1  YESNO
C
C     ASK IF THRESHOLD CULLING IS DESIRED
          TYPE 1000
          ACCEPT 2000, YESNO
          IF (YESNO .NE. 'Y')  RETURN
C     SET DEFAULT THRESHOLD & REQUEST INPUT VALUE TO REPLACE IT
          THRESH = 1.0
          TYPE 3000, THRESH
          ACCEPT *, THRESH
C     CULL OUT BELOW-THRESHOLD ROWS
          DO 100 I=1,KROW1+KROW2
              IF (MTXKEY(I) .LT. 0)  GOTO 100
              IF (SORTLIST(MTXKEY(I),ROWKEY(I)) .GE. THRESH) GOTO 150
              MTXKEY(I) = -MTXKEY(I)
              KOUT = KOUT - 1
100           CONTINUE
C     CHECK TO SEE IF ANY ROWS REMAIN
          IF (KOUT .EQ. 0)  THEN
              TYPE *, 'NO MERGING:'
              TYPE *, 'ALL TERMS ARE LESS THAN THRESHOLD VALUE:',THRESH
              CLOSE(UNIT=1)
              CLOSE(UNIT=2)
              RETURN 1
              END IF
C     ALL REMAINING TERMS ARE GREATER THAN OR EQUAL TO THRESHOLD
150       RETURN
C
1000  FORMAT(/,' DELETE VALUES BELOW A THRESHOLD? (Y OR N): ',$)
2000  FORMAT(A1)
3000  FORMAT(6X,'ENTER VALUE FOR THRESHHOLD. "," FOR DEFAULT ',
     *       1PE8.1,' : ',$)
      END
```

```
        SUBROUTINE DELCEILING(SORTLIST,MTXKEY,ROWKEY,KROW1,KROW2,KOUT,*)
C
C       DELETE ROWS WHERE THE MERGING TERM IS ABOVE A GIVEN CEILING
C       VALUE.
C
        INTEGER MTXKEY(200), ROWKEY(200)
        REAL SORTLIST(2,100)
        CHARACTER*1  YESNO
C
C       ASK IF CEILING CULLING IS DESIRED
            TYPE 1000
            ACCEPT 2000, YESNO
            IF (YESNO .NE. 'Y')  RETURN
C       SET DEFAULT CEILING & REQUEST INPUT VALUE TO REPLACE IT
            CEIL = 1.0E+35
            TYPE 3000, CEIL
            ACCEPT *, CEIL
C       CULL OUT ABOVE-CEILING ROWS
            DO 100 I=1,KROW1+KROW2
                IF (MTXKEY(I) .LT. 0)  GOTO 100
                IF (SORTLIST(MTXKEY(I),ROWKEY(I)) .GT. CEIL)  THEN
                    MTXKEY(I) = -MTXKEY(I)
                    KOUT = KOUT - 1
                    END IF
  100           CONTINUE
C       CHECK TO SEE IF ANY ROWS REMAIN
            IF (KOUT .EQ. 0)  THEN
                TYPE *, 'NO MERGING:'
                TYPE *, 'ALL TERMS ARE GREATER THAN CEILING VALUE:',CEIL
                CLOSE(UNIT=1)
                CLOSE(UNIT=2)
                RETURN 1
                END IF
C       ALL REMAINING TERMS ARE LESS THAN OR EQUAL TO CEILING
  150       RETURN
C
 1000   FORMAT(/,' DELETE VALUES ABOVE A CEILING? (Y OR N): ',$)
 2000   FORMAT(A1)
 3000   FORMAT(6X,'ENTER VALUE FOR CEILING. "," FOR DEFAULT',
       *        1PE8.1,' : ',$)
        END
```

```
      SUBROUTINE GETASCMTX(KROW1,KROW2,MROW,NCOL,*)
C
C     OPEN ASSOCIATED MATRIX PAIR (LOGICAL UNIT NOS. 1&2) AND
C     CHECK FOR VALID DIMENSIONS.
C
      CHARACTER FILNAM1*35, FILNAM2*35, MATRIX1*8, MATRIX2*8
C
C     GET NAMES OF ASSOCIATED MATRIX PAIR
C         FIRST MATRIX:
1             TYPE 1000, '1> ', '1'
              ACCEPT 2000, FILNAM1
              OPEN(UNIT=1,NAME=FILNAM1,TYPE='OLD',READONLY,
     *             CARRIAGECONTROL='LIST',ERR=1)
              READ(1,3000) MATRIX1,MROW1,NCOL1
C         SECOND MATRIX:
2             TYPE 1000, '2> ', '2'
              ACCEPT 2000, FILNAM2
              OPEN(UNIT=2,NAME=FILNAM2,TYPE='OLD',READONLY,
     *             CARRIAGECONTROL='LIST',ERR=2)
              READ(2,3000) MATRIX2,MROW2,NCOL2
C     CHECK FOR VALID DIMENSIONS
          TYPE 4000, '1> ', MATRIX1,MROW1,NCOL1
          TYPE 4000, '2> ', MATRIX2,MROW2,NCOL2
          IF ((NCOL1.NE.KROW1) .OR. (NCOL2.NE.KROW2))   THEN
              TYPE 5000, 'NO MERGING DONE FOR THIS ASSOCIATED PAIR'
              CLOSE(UNIT=1)
              CLOSE(UNIT=2)
              RETURN 1
          ELSE IF (MROW1 .NE. MROW2)  THEN
              TYPE 6000, 'NO MERGING DONE FOR THIS ASSOCIATED PAIR'
              CLOSE(UNIT=1)
              CLOSE(UNIT=2)
              RETURN 1
          ELSE
C             MATRICES ARE OK
              MROW = MROW1
              NCOL = NCOL1 + NCOL2
              RETURN
          END IF
C
1000  FORMAT(1H ,A3,'ENTER NAME OF ASSOCIATED MATRIX FILE ',A1,': ',$)
2000  FORMAT(A35)
3000  FORMAT(A8,2I8)
4000  FORMAT(6X,A3,A8,' IS A ',I8,' ROW BY ',I8,' COLUMN MATRIX')
5000  FORMAT(6X,'# OF COLUMNS DOES NOT MATCH # OF ROWS OF KEY MATRIX',
     *       /,6X,A40)
6000  FORMAT(6X,'MATRICES HAVE DIFFERENT # OF ROWS',/,6X,A40)
      END
```

```
      SUBROUTINE GETKEYMTX(MATRIX1,MATRIX2,KROW1,KROW2,KCOL1,KCOL2,*)
C
C     GET KEY MATRIX FILENAMES, OPEN THEM (FORTRAN LOGICAL UNIT
C     NOS. 1&2), READ HEADER RECORDS, AND CHECK FOR COMPATIBLE SIZES.
C
      CHARACTER*35 FILNAM1, FILNAM2
      CHARACTER*8  MATRIX1,MATRIX2
C
C     FIRST MATRIX:
1         TYPE 2000, '1', '1'
          ACCEPT 6000, FILNAM1
          OPEN(UNIT=1,NAME=FILNAM1,TYPE='OLD',READONLY,
     *          CARRIAGECONTROL='LIST',ERR=1)
          READ(1,3000) MATRIX1,KROW1,KCOL1
C     SECOND MATRIX:
2         TYPE 2000, '2', '2'
          ACCEPT 6000, FILNAM2
          OPEN(UNIT=2,NAME=FILNAM2,TYPE='OLD',READONLY,
     *          CARRIAGECONTROL='LIST',ERR=2)
          READ(2,3000) MATRIX2,KROW2,KCOL2
C     CHECK FOR SAME NUMBER OF COLUMNS
          IF (KCOL1 .NE. KCOL2)  THEN
              CLOSE(UNIT=1)
              CLOSE(UNIT=2)
              TYPE 7000, '1> ', MATRIX1,KROW1,KCOL1
              TYPE 7000, '2> ', MATRIX2,KROW2,KCOL2
              TYPE *, 'NO MERGING DONE:'
              TYPE *, 'MATRICES HAVE DIFFERENT NO. OF COLUMNS'
              RETURN 1
              END IF
      RETURN
C
2000  FORMAT(1X,A1,'> ENTER FILENAME OF KEY MATRIX ',A1,': ',$)
3000  FORMAT(A8,2I8)
6000  FORMAT(A35)
7000  FORMAT(6X,A3,A8,' IS A ',I8,' ROW BY ',I8,' MATRIX')
      END
```

```
      SUBROUTINE GETSORTCOL(MATRIX1,MATRIX2,KROW1,KROW2,
     *                                        KCOL1,KCOL2,SORTLIST)
C
C     DETERMINE KEY COLUMNS FOR SORTED MERGE, AND READ THEM
C     INTO SORTING ARRAY, SORTLIST.
C
      REAL SORTLIST(2,100)
      CHARACTER*8 MATRIX1, MATRIX2
C
C     FIRST MATRIX:
1         TYPE 7000, '1> ', MATRIX1,KROW1,KCOL1
          TYPE 8000
          ACCEPT *, MRGCOL1
          IF ((MRGCOL1.LT.1) .OR. (MRGCOL1.GT.KCOL1))  GOTO 1
C     SECOND MATRIX:
2         TYPE 7000, '2> ', MATRIX2,KROW2,KCOL2
          TYPE 8000
          ACCEPT *,MRGCOL2
          IF ((MRGCOL2.LT.1) .OR. (MRGCOL2.GT.KCOL2))  GOTO 2
C     READ MERGE COLUMNS INTO SORTING ARRAY
              DO 100 J=1,MRGCOL1
              DO 100 I=1,KROW1
                  READ(1,9000) SORTLIST(1,I)
100               CONTINUE
              DO 200 J=1,MRGCOL2
              DO 200 I=1,KROW2
                  READ(2,9000) SORTLIST(2,I)
200               CONTINUE
      RETURN
C
7000  FORMAT(6X,A3,A8,' IS A ',I8,' ROW BY ',I8,' COLUMN MATRIX')
8000  FORMAT(9X,'ENTER COLUMN NO. TO USE FOR SORTED MERGE: ',$)
9000  FORMAT(E16.8)
      END
```

```
      SUBROUTINE KEYLIST(SORTLIST,KROW1,KROW2,MTXKEY,ROWKEY)
C
C     GENERATE KEY LISTS FOR SORTED MERGING
C
      REAL SORTLIST(2,100)
      INTEGER MTXKEY(100), ROWKEY(100)
C
      I1 = 1
      I2 = 1
      DO 100 I=1,KROW1+KROW2
         IF (I1 .GT. KROW1)  THEN
C           END OF COLUMN FOR MATRIX 1 REACHED; SET KEY FOR MATRIX 2
            MTXKEY(I) = 2
            ROWKEY(I) = I2
            I2 = I2 + 1
         ELSE IF (I2 .GT. KROW2)  THEN
C           END OF COLUMN FOR MATRIX 2 REACHED; SET KEY FOR MATRIX 1
            MTXKEY(I) = 1
            ROWKEY(I) = I1
            I1 = I1 + 1
         ELSE IF (SORTLIST(1,I1) .LE. SORTLIST(2,I2))  THEN
C           SET KEY FOR MATRIX 1
            MTXKEY(I) = 1
            ROWKEY(I) = I1
            I1 = I1 + 1
         ELSE
C           SET KEY FOR MATRIX 2
            MTXKEY(I) = 2
            ROWKEY(I) = I2
            I2 = I2 + 1
         END IF
100   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MERGEASC(MTXKEY,KOUT,MROW,NCOL)
C
C     MERGE ASSOCIATED MATRIX PAIR INTO OUTPUT FILE & CLOSE FILES
C
      INTEGER MTXKEY(200)
      CHARACTER FILNAM3*35, MATRIX3*8
C
C     GET NAME OF NEW ASSOCIATED OUTPUT (MERGED) MATRIX
          TYPE 1000
          ACCEPT 2000, FILNAM3
          TYPE 3000
          ACCEPT 4000, MATRIX3
          OPEN(UNIT=3,NAME=FILNAM3,TYPE='NEW',CARRIAGECONTROL='LIST')
C     MERGE MATRIX PAIR INTO OUTPUT MATRIX
          WRITE(3,5000) MATRIX3,MROW,KOUT
          DO 100 J=1,NCOL
          DO 100 I=1,MROW
              READ( ABS(MTXKEY(J)) , 6000 )  TERM
              IF (MTXKEY(J) .GT. 0)  WRITE(3,7000) TERM
100           CONTINUE
C     FINISHED: CLOSE FILES & RETURN
          CLOSE(UNIT=1)
          CLOSE(UNIT=2)
          CLOSE(UNIT=3)
          RETURN
C
1000  FORMAT(/,' ENTER NAME OF NEW (MERGED) MATRIX FILE: ',$)
2000  FORMAT(A35)
3000  FORMAT(6X,'ENTER MATRIX NAME: ',$)
4000  FORMAT(A8)
5000  FORMAT(A8,2I8)
6000  FORMAT(E16.8)
7000  FORMAT(1PE16.8)
      END
```

```
      SUBROUTINE MERGEKEY(MTXKEY,KOUT)
C
C     MERGE KEY MATRICES INTO OUTPUT MATRIX & CLOSE FILES
C
      INTEGER MTXKEY(200)
      CHARACTER*8  MATRIX1,MATRIX2,MATRIX3, FILNAM3*35
C
C     GET NAME OF OUTPUT KEY MATRIX & OPEN NEW FILE
          TYPE 1000
          ACCEPT 2000, FILNAM3
          TYPE 3000
          ACCEPT 4000, MATRIX3
          OPEN(UNIT=3,NAME=FILNAM3,TYPE='NEW',CARRIAGECONTROL='LIST')
C     MERGE KEY MATRICES INTO OUTPUT MATRIX
          REWIND 1
          REWIND 2
          READ(1,5000) MATRIX1,KROW1,KCOL1
          READ(2,5000) MATRIX2,KROW2,KCOL2
          WRITE(3,5000) MATRIX3,KOUT,KCOL1
          DO 100 J=1,KCOL1
          DO 100 I=1,KROW1+KROW2
              READ( ABS(MTXKEY(I)) , 6000 )  TERM
              IF (MTXKEY(I) .GT. 0)  WRITE(3,7000)  TERM
  100         CONTINUE
C     FINISHED: CLOSE FILES & RETURN
          CLOSE(UNIT=1)
          CLOSE(UNIT=2)
          CLOSE(UNIT=3)
          RETURN
C
 1000 FORMAT(/,' ENTER NAME OF NEW (MERGED) MATRIX FILE: ',$)
 2000 FORMAT(A35)
 3000 FORMAT(6X,'ENTER MATRIX NAME: ',$)
 4000 FORMAT(A8)
 5000 FORMAT(A8,2I8)
 6000 FORMAT(E16.8)
 7000 FORMAT(1PE16.8)
      END
```

STATCORR

The statistical correlation program STATCORR was written for implementation on the GSFC Code 750 VAX-11/780 computer system, and the analysis was done using the Code 750 VAX version of Level 17.5 NASTRAN. The empirical testing was performed by the GSFC Environmental Test & Integration Branch and test mode shapes, frequencies, and damping were determined using the standard in-house processing software on a PDP-11/35 computer system.

Three data files are required to input analytical mode shape data to the STATCORR program: 1) the LAMA matrix file, which contains data from the NASTRAN LAMA table (frequency, mass, and stiffness vectors) and mode symmetry information; 2) the PHITE matrix file, which contains the eigenvectors generated by the NASTRAN run; 3) the grid point list file, obtained from the TESET vector. All the data required to build these files is contained in the NASTRAN punch file produced by using the DMAP ALTER package discussed earlier.

Since the NASTRAN analysis and the statistical correlation are performed on the same computer system, the NASTRAN punch file is immediately available for processing (to generate the analytical input files to STATCORR) by the appropriate intermediate processor programs MERGE,LAMA, UNPACKDMI, GRDPTLIST, discussed subsequently. These processor programs read the NASTRAN punch file and produce input files in the format required for input to STATCORR.

One data file for each experimental mode is required to input experimental mode shape data to the STATCORR program. Each experimental data file contains all data (frequency, damping, symmetry, gridpoint IDs, and mode shape data) to describe one experimental mode. The in-house Modal Survey processing program has the capability of generating various ASCII data files containing the necessary grid-point ID, frequency, damping, and mode-shape information, which can be further processed to generate files in the appropriate format for input to the STATCORR program.

Since the experimental mode data is stored on a different computer system, it was necessary to provide a method of data transfer between the Code 750 VAX-11/780 computer and the Environmental Test & Integration Branch PDP-11/35 computer. Because rapid, convenient file transfer was desired and the two computer systems had no similar storage media to permit simple volume transfer between the machines, a remote terminal emulator program with ASCII file transfer capability was adapted for use on the PDP-11/35. This permits the PDP computer to be used as a remote (dial-up) terminal to the VAX-11/780, allowing direct transfer of ASCII text files via phone line between the VAX and PDP computers.

For a user's guide describing the execution of NASTRAN, STATCORR, and the associated preprocessing programs, please refer to NASA Technical Memorandum 86044. Also included is a working example using the SPARTAN-1 model, and FORTRAN source code listings of the STATCORR program and the preprocessing programs TESETDMI, UNPACKDMI, LAMA, and GRDPTLST.

```
   CCCC  TTTTT  GGGG  BBBB
  C        T   G      B   B
  C        T   G      B   B
  C        T   G      BBBB
  C        T   G GGG  B   B
  C        T   G   G  B   B
   CCCC    T    GGG   BBBB

  SSSSSSSS TTTTTTTTTT   AAAAAA   TTTTTTTTTT  CCCCCCCC   000000   RRRRRRRR  RRRRRRRR
  SSSSSSSS TTTTTTTTTT   AAAAAA   TTTTTTTTTT  CCCCCCCC   000000   RRRRRRRR  RRRRRRRR
 SS            TT     AA      AA     TT     CC        00      00 RR      RR RR      RR
 SS            TT     AA      AA     TT     CC        00      00 RR      RR RR      RR
 SS            TT     AA      AA     TT     CC        00      00 RR      RR RR      RR
 SS            TT     AA      AA     TT     CC        00      00 RR      RR RR      RR
  SSSSSS       TT     AA      AA     TT     CC        00      00 RRRRRRRR  RRRRRRRR
  SSSSSS       TT     AA      AA     TT     CC        00      00 RRRRRRRR  RRRRRRRR
        SS     TT     AAAAAAAAAA     TT     CC        00      00 RR  RR    RR  RR
        SS     TT     AAAAAAAAAA     TT     CC        00      00 RR  RR    RR  RR
        SS     TT     AA      AA     TT     CC        00      00 RR    RR  RR    RR     ....
        SS     TT     AA      AA     TT     CC        00      00 RR    RR  RR    RR     ....
 SSSSSSSS      TT     AA      AA     TT      CCCCCCCC   000000   RR      RR RR      RR  ....
 SSSSSSSS      TT     AA      AA     TT      CCCCCCCC   000000   RR      RR RR      RR  ....

 FFFFFFFFFF   000000   RRRRRRRR      ::::     11
 FFFFFFFFFF   000000   RRRRRRRR      ::::     11
 FF         00      00 RR      RR    ::::   1111
 FF         00      00 RR      RR    ::::   1111
 FF         00      00 RR      RR             11
 FF         00      00 RR      RR             11
 FFFFFFFF   00      00 RRRRRRRR      ::::     11
 FFFFFFFF   00      00 RRRRRRRR      ::::     11
 FF         00      00 RR  RR        ::::     11
 FF         00      00 RR  RR        ::::     11
 FF         00      00 RR    RR        ::     11
 FF         00      00 RR    RR        ::     11
 FF           000000   RR      RR    ::     111111
 FF           000000   RR      RR    ::     111111

   CCCC  TTTTT  GGGG  BBBB
  C        T   G      B   B
  C        T   G      B   B
  C        T   G      BBBB
  C        T   G GGG  B   B
  C        T   G   G  B   B
   CCCC    T    GGG   BBBB
```

```
      PROGRAM STATCORR
C
      PARAMETER (MAXMODE= 200)
      PARAMETER (MAXDOF = 1000)
      COMMON /LIMITS/ MXMD,MXDF
      DATA MXMD/MAXMODE/, MXDF/MAXDOF/

      REAL FREQAN(MAXMODE), FREQEX(MAXMODE)
      REAL DISPAN(MAXDOF,MAXMODE), DISPEX(MAXDOF,MAXMODE)
      REAL MASS(MAXMODE),STIFF(MAXMODE),DAMP(MAXMODE)
      REAL CORREL(MAXMODE,MAXMODE), C(MAXMODE,MAXMODE)
      REAL S(MAXMODE,MAXMODE)
      REAL RMSA(MAXMODE,MAXMODE), RMSE(MAXMODE,MAXMODE)
      REAL AVECT(MAXDOF),EVECT(MAXDOF),DIFF(MAXDOF),DIFABS(MAXDOF)
      INTEGER IDDOFA(MAXDOF,MAXMODE),IDDOFE(MAXDOF,MAXMODE)
      INTEGER ICOMPA(MAXDOF,MAXMODE), ICOMPE(MAXDOF,MAXMODE)
      INTEGER NDOFA(MAXMODE), NDOFE(MAXMODE)
      INTEGER IDDOFI(MAXDOF),ICOMPI(MAXDOF)
      INTEGER NAFIT(MAXMODE),MEFIT(MAXMODE)
      INTEGER ASYMM(3,MAXMODE),ESYMM(3,MAXMODE)
      CHARACTER*10 AHEADER(4),EHEADER(4),HEADER(32)
      CHARACTER*9 CDATE
      CHARACTER*1 YESNO
      LOGICAL PRTAN,PRTEX,LSYMM,LPRINT

      COMMON /WHEN/ CDATE

C
      DATA PRTAN,PRTEX,LSYMM,LPRINT/4*.FALSE./
      DATA AHEADER/2*' NASTRAN',2*'    TEST'/
      DATA EHEADER/2*'    TEST', 2*' NASTRAN'/
      DATA HEADER/'# OF GP''S', '  CORREL', 2*' ', 2*'    RMS',
     *            ' MAX REL', '    GRID', '    MODE', 'FREQUENCY',
     *            '    MODE', 'FREQUENCY', ' COMPARED', '   COEFF',
     *            '      CA', '       S', '(NASTRAN)', '   (TEST)',
     *            'DIFFERENCE', '   POINT', 12*'__________'/
C
C     PRINT SECTION 1 HEADER
         CALL DATE(CDATE)
         PRINT 8000, CDATE
         PRINT 4000
C     CHECK FOR SEPARATE PRINT FILE
         TYPE 5000
         ACCEPT 7000, YESNO
         IF (YESNO .EQ. 'Y')  LPRINT = .TRUE.
         IF (LPRINT)  PRINT 5500, YESNO
C     GET ANALYTICAL FREQ, MASS, STIFFNESS, MODE-SHAPE, & DOF LISTS
         CALL GETAN(FREQAN,MASS,STIFF,ASYMM,DISPAN,NAN,
     *                                IDDOFA,ICOMPA,NDOFA,LPRINT)
C     GET EXPERIMENTAL FREQ, DAMPING, MODE-SHAPE, & DOF LISTS
         CALL GETEXP(FREQEX,DAMP,ESYMM,DISPEX,MEX,IDDOFE,ICOMPE,
     *                                                NDOFE,LPRINT)
C     DETERMINE SYMMETRY, MODE-SHAPE DUMP, & THRESHOLD OPTIONS
         CALL OPTIONS(LSYMM,PRTAN,PRTEX,RTHRESH,LPRINT)
C     PRINT INPUT SUMMARY
         CALL INPSUM(NAN,FREQAN,MASS,STIFF,ASYMM,MEX,FREQEX,DAMP,ESYMMD
C     SORT DOF LISTS & MODE SHAPE MATRICES (BASED ON DOF SORT)
         CALL DOFSORT(IDDOFA,ICOMPA,NDOFA,DISPAN,NAN)
         CALL DOFSORT(IDDOFE,ICOMPE,NDOFE,DISPEX,MEX)
C     COMPUTE CORRELATION COEFFICIENTS & RMS VALUES FOR ALL
C                     POSSIBLE ANALYTICAL/EXPERIMENTAL PAIRS
```

```
          DO 100 N=1,NAN
          DO 100 M=1,MEX
C               GET VECTORS TO BE COMPARED & THEIR DOF INTERSECTION SET
                     CALL GETVEC(DISPAN(1,N),NDOFA(N),AVECT)
                     CALL GETVEC(DISPEX(1,M),NDOFE(M),EVECT)
                     CALL INTERSECT(AVECT,EVECT,IDDOFA(1,N),IDDOFE(1,M),
     *                              ICOMPA(1,N),ICOMPE(1,M),
     *                              NDOFA(N),NDOFE(M),IDDOFI,ICOMPI,NDOFI,
     *                              ASYMM(1,N),ESYMM(1,M),LSYMM)
C               COMPUTE CORR COEFFS & RMS VALUES FOR THIS PAIR
                     CALL CORRMS(AVECT,EVECT,NDOFI,CORREL(N,M),
     *                           C(N,M),S(N,M),RMSA(N,M),RMSE(N,M))
100             CONTINUE
C         PRINT CORRELATION COEFFICIENT TABLE
          CALL CORRTBL(CORREL,NAN,MEX)
C         DETERMINE BEST MATCH FOR EACH EXPERIMENTAL & ANALYTICAL MODE
          CALL MATCH(CORREL,NAN,MEX,NAFIT,MEFIT)
C         FOR EACH MATCHED PAIR, GET X/RMS DIFFERENCES & PRINT SUMMARY
C             PRINT ANALYTICAL MODE SHAPE HEADER
                PRINT 8000, CDATE
                PRINT 1000, AHEADER,HEADER
          DO 200 N=1,NAN
                M = NAFIT(N)
C               GET VECTORS TO BE COMPARED & THEIR DOF INTERSECTION SET
                     CALL GETVEC(DISPAN(1,N),NDOFA(N),AVECT)
                     CALL GETVEC(DISPEX(1,M),NDOFE(M),EVECT)
                     CALL INTERSECT(AVECT,EVECT,IDDOFA(1,N),IDDOFE(1,M),
     *                              ICOMPA(1,N),ICOMPE(1,M),
     *                              NDOFA(N),NDOFE(M),IDDOFI,ICOMPI,NDOFI,
     *                              ASYMM(1,N),ESYMM(1,M),LSYMM)
C               CALCULATE INDIVIDUAL DIFFERENCES
                     CALL RMSDIFF(AVECT,EVECT,NDOFI,RMSA(N,M),RMSE(N,M),
     *                            CORREL(N,M),S(N,M),DIFF,DIFABS)
C               PRINT SUMMARY FOR THIS PAIR, WITH MAX DIFF & > THRESHOLD
                     CALL PRINT(N,FREQAN(N),M,FREQEX(M),NDOFI,CORREL(N,M),
     *                          C(N,M),S(N,M),RMSA(N,M),RMSE(N,M),
     *                          DIFF,DIFABS,IDDOFI,ICOMPI,RTHRESH)
200             CONTINUE
C             PRINT EXPERIMENTAL MODE SHAPE HEADER
                PRINT 8000, CDATE
                PRINT 3000, EHEADER,HEADER
          DO 300 M=1,MEX
                N = MEFIT(M)
C               GET VECTORS TO BE COMPARED & THEIR DOF INTERSECTION SET
                     CALL GETVEC(DISPEX(1,M),NDOFE(M),EVECT)
                     CALL GETVEC(DISPAN(1,N),NDOFA(N),AVECT)
                     CALL INTERSECT(AVECT,EVECT,IDDOFA(1,N),IDDOFE(1,M),
     *                              ICOMPA(1,N),ICOMPE(1,M),
     *                              NDOFA(N),NDOFE(M),IDDOFI,ICOMPI,NDOFI,
     *                              ASYMM(1,N),ESYMM(1,M),LSYMM)
C               CALCULATE COMBINED RMS AND INDIVIDUAL DIFFERENCES
                     CALL RMSDIFF(AVECT,EVECT,NDOFI,RMSA(N,M),RMSE(N,M),
     *                            CORREL(N,M),S(N,M),DIFF,DIFABS)
C               PRINT SUMMARY FOR THIS PAIR, WITH MAX DIFF & > THRESHOLD
                     CALL PRINT(M,FREQEX(M),N,FREQAN(N),NDOFI,CORREL(N,M),
     *                          C(N,M),S(N,M),RMSA(N,M),RMSE(N,M),
     *                          DIFF,DIFABS,IDDOFI,ICOMPI,RTHRESH)
300             CONTINUE
C         PRINT ANALYTICAL &/OR EXPERIMENTAL MODE-SHAPE VECTORS
          CALL PRINTPHI(NAN,NDOFA,IDDOFA,ICOMPA,DISPAN,PRTAN,
     *                  MEX,NDOFE,IDDOFE,ICOMPE,DISPEX,PRTEX)
```

```
      STOP 'FORTRAN STOP -- PROCESSING COMPLETED'
C
1000  FORMAT(1H0,'4. ANALYTICAL MODE SHAPES AND THEIR BEST',
     *        ' EXPERIMENTAL MATCHES:',/'0',12(A10,1X),2(/12(1X,A10)))
3000  FORMAT(1H0,'5. EXPERIMENTAL MODE SHAPES AND THEIR BEST'
     *        ' ANALYTICAL MATCHES:',/'0',12(A10,1X),2(/12(1X,A10)))
4000  FORMAT(1H0,' 1. INTERACTIVE DIALOG:')
5000  FORMAT(/,' IS A SEPARATE OUTPUT LISTING FILE TO BE PRINTED?',
     *         '(Y OR N): ',$)
5500  FORMAT(/,' IS A SEPARATE OUTPUT LISTING FILE TO BE PRINTED?',
     *         '(Y OR N): ',A1)
7000  FORMAT(A1)
8000  FORMAT(1H1,'NASTRAN MODAL ANALYSIS - MODAL SURVEY STATISTICAL',
     *        ' CORRELATION', /1X,A9)
      END
```

```
      SUBROUTINE CORRMS(AVECT,EVECT,NDOFI,CORREL,CA,S,RMSA,RMSE)
C
C     COMPUTE CORRELATION COEFFICIENTS & RMS VALUES
C
      REAL AVECT(*),EVECT(*)
C
C     INITIALIZE RMS'S & CORR COEFFS
          RMSA = 0.0
          RMSE = 0.0
          CORREL = 0.0
          CA = 0.0
          S = 0.0
      IF (NDOFI .EQ. 0)  RETURN
C     COMPUTE VARIANCES OF ANALYTICAL & EXPERIMENTAL VECTORS
C     NOTE: THE FINAL VALUES COMPUTED HERE FOR THE "VARIANCES"
C     ARE NOT TRUE VARIANCES. TO OBTAIN VARIANCES, THE VALUES
C     VARA, VARE, & VARAE SHOULD BE EACH MULTIPLIED BY  1/NDOFI .
C     THESE VALUES ARE NOT REPORTED, BUT ARE USED ONLY TO
C     CALCULATE CORREL, CA, S, RMSA, & RMSE.  THE 1/NDOFI FACTOR
C     IS NOT INCLUDED, SINCE IT APPEARS IN BOTH THE NUMERATOR AND
C     DENOMINATOR OF CORREL & CA; THE EXTRA ARITHMETIC OPERATIONS
C     WOULD SIMPLY INTRODUCE UNNECESSARY ERROR IN THE RESULTS.
C     THE 1/NDOFI FACTOR IS INCLUDED IN THE CALCULATIONS OF
C     S, RMSA, & RMSE.
          VARA = 0.0
          VARE = 0.0
          VARAE = 0.0
          DO 100 N=1,NDOFI
              VARA = VARA + AVECT(N)*AVECT(N)
              VARE = VARE + EVECT(N)*EVECT(N)
              VARAE = VARAE + AVECT(N)*EVECT(N)
100           CONTINUE
C     COMPUTE CORRELATION COEFFICIENT FOR THIS PAIR OF VECTORS
          IF (VARAE .NE. 0.0)  CORREL = VARAE/SQRT(VARA*VARE)
C     COMPUTE CORREL COEFF (REFERENCED TO ANALYTICAL VECTOR)
          IF (VARA .NE. 0.0)  CA = VARAE/VARA
C     COMPUTE ROOT OF MEAN SQUARE DIFFERENCE
          S = SQRT( ABS(VARE-CA*CA*VARA)/NDOFI )
C     COMPUTE RMS VALUES
C     NOTE:  FOR THIS APPLICATION, RMS = STD.DEV. = SQRT(VARIANCE) .
          RMSA = SQRT(VARA/NDOFI)
          RMSE = SQRT(VARE/NDOFI)
      RETURN
      END
```

```
      SUBROUTINE CORRTBL(CORREL,NAN,MEX)
C
C     PRINT AN  NAN X MEX  TABLE OF THE ANALYTICAL VS. EXPERIMENTAL
C     CORRELATION COEFFICIENTS.
C
      COMMON /LIMITS/ MAXMODE,MAXDOF
      COMMON /WHEN/ CDATE
      REAL CORREL(MAXMODE,MAXMODE)
      CHARACTER CDATE*9
C
      DO 200 MFIRST=1,MEX,16
      DO 200 NFIRST=1,NAN,50
          MLAST = MIN( MEX, MFIRST+15 )
          NLAST = MIN( NAN, NFIRST+49 )
C         PRINT HEADER FOR ONE PAGE OF THE TABLE
              PRINT 1000, CDATE
              PRINT 2000
              PRINT 3000, (M,M=MFIRST,MLAST)
          DO 100 N=NFIRST,NLAST
              PRINT 4000, N,(CORREL(N,M),M=MFIRST,MLAST)
100       END DO
200   END DO

      RETURN

1000  FORMAT(1H1,'NASTRAN MODAL ANALYSIS - MODAL SURVEY STATISTICAL ',
     *       'CORRELATION', /1X,A9,
     *       /1H0,'3. CORRELATION COEFFICIENTS FOR ANALYTICAL VS. ',
     *       'EXPERIMENTAL COMPARISONS:')
2000  FORMAT('0ANALYTICAL',24X,'EXPERIMENTAL MODES')
3000  FORMAT(3X,'MODES',3X,I5,15I7)
4000  FORMAT(1X,I6,4X,16(1X,F6.3))
      END
```

```
      SUBROUTINE DOFSORT( IDDOF, ICOMP, NDOF, DISP, NMODES)
C
C     PUT DOF LIST FOR EACH MODE INTO ASCENDING SORT,
C     AND SLAVE-SORT THE CORRESPONDING MODE-SHAPE VECTORS.
C
      COMMON /LIMITS/ MAXMODE,MAXDOF
      REAL DISP(MAXDOF,MAXMODE)
      INTEGER IDDOF(MAXDOF,MAXMODE), ICOMP(MAXDOF,MAXMODE), NDOF(MAXMODE)
C
      DO 200 J=1,NMODES
          IF (NDOF(J) .EQ. 1) GOTO 200
          DO 100 I=1,NDOF(J)-1
          DO 100 K=I+1,NDOF(J)
              IF (IDDOF(K,J) .LT. IDDOF(I,J))  THEN
                  CALL SWAP( IDDOF(I,J), IDDOF(K,J) )
                  CALL SWAP( ICOMP(I,J), ICOMP(K,J) )
                  CALL SWAP(  DISP(I,J),  DISP(K,J) )
              ELSE IF ((IDDOF(K,J) .EQ. IDDOF(I,J)) .AND.
     *                 (ICOMP(K,J) .LT. ICOMP(I,J))        )  THEN
                  CALL SWAP( IDDOF(I,J), IDDOF(K,J) )
                  CALL SWAP( ICOMP(I,J), ICOMP(K,J) )
                  CALL SWAP(  DISP(I,J),  DISP(K,J) )
              END IF
100           CONTINUE
200       CONTINUE
      RETURN
      END
```

```
      SUBROUTINE GETAN(FREQS,MASS,STIFF,ASYMM,SHAPES,NMODES,
     *                                   IDDOF,ICOMPA,NDOFS,LPRINT)
C
C     GET FREQUENCY, MASS, STIFFNESS & SYMMETRY VECTORS;
C     MODE-SHAPE MATRIX; AND GRID POINT LIST.
C
      COMMON /LIMITS/ MAXMODE,MAXDOF

      REAL FREQS(MAXMODE),MASS(MAXMODE),STIFF(MAXMODE)
      REAL SHAPES(MAXDOF,MAXMODE)
      INTEGER ASYMM(3,MAXMODE)
      INTEGER IDDOF(MAXDOF,MAXMODE),ICOMPA(MAXDOF,MAXMODE)
      INTEGER NDOFS(MAXMODE)
      CHARACTER*35 FRQFIL,SHPFIL,DOFFIL
      CHARACTER*8 FRQMTX,SHPMTX
      LOGICAL LPRINT
C
C     GET FILE NAMES
10        TYPE 1000
          ACCEPT 2000, FRQFIL
          OPEN(UNIT=1,NAME=FRQFIL,TYPE='OLD',READONLY,ERR=10)
          IF (LPRINT)  PRINT 1500, FRQFIL
20        TYPE 3000
          ACCEPT 2000, SHPFIL
          OPEN(UNIT=2,NAME=SHPFIL,TYPE='OLD',READONLY,ERR=20)
          IF (LPRINT)  PRINT 3500, SHPFIL
30        TYPE 4000
          ACCEPT 2000, DOFFIL
          OPEN(UNIT=3,NAME=DOFFIL,TYPE='OLD',READONLY,ERR=30)
          IF (LPRINT)  PRINT 4500, DOFFIL
C     READ FREQUENCY FILE
          READ(1,7000) FRQMTX,NFREQS,NCOLS
          IF (NCOLS .NE. 6)  THEN
40            TYPE 5000, FRQMTX,NCOLS
              IF (LPRINT)  PRINT 5000, FRQMTX,NCOLS
              CLOSE(UNIT=1)
              CLOSE(UNIT=2)
              CLOSE(UNIT=3)
              STOP
              END IF
          IF (NFREQS .GT. MAXMODE)  PRINT 6000, NFREQS, MAXMODE
          NMODES = MIN(NFREQS,MAXMODE)
          DO 100 I=1,6
          DO 100 J=1,NFREQS
              IF (J .GT. MAXMODE)  THEN
                  READ(1,8000) DUMMY
              ELSE IF (I .EQ. 1) THEN  ! GET FREQUENCY LIST
                  READ(1,8000) FREQS(J)
              ELSE IF (I .EQ. 2) THEN  ! GET MASS LIST
                  READ(1,8000) MASS(J)
              ELSE IF (I .EQ. 3) THEN  ! GET STIFFNESS LIST
                  READ(1,8000) STIFF(J)
              ELSE IF (I .EQ. 4) THEN  ! GET PLANE 1 SYMMETRY LIST
                  READ(1,8000) SYMM
                  ASYMM(1,J) = NINT(SYMM)
              ELSE IF (I .EQ. 5) THEN  ! GET PLANE 2 SYMMETRY LIST
                  READ(1,8000) SYMM
                  ASYMM(2,J) = NINT(SYMM)
              ELSE IF (I .EQ. 6) THEN  ! GET PLANE 3 SYMMETRY LIST
                  READ(1,8000) SYMM
                  ASYMM(3,J) = NINT(SYMM)
```

```
            END IF
100         CONTINUE
C     READ MODE-SHAPE FILE
        READ(2,7000) SHPMTX,NDISPS,NSHAPES
        IF (NSHAPES .NE. NFREQS)  THEN
            NMODES = MIN( NFREQS, NSHAPES, NMODES )
            TYPE 9000, NFREQS,NSHAPES,NMODES
            IF (LPRINT)  PRINT 9000, NFREQS,NSHAPES,NMODES
            END IF
        IF (NDISPS .GT. MAXDOF)  PRINT 10000, NDISPS,MAXDOF
        DO 200 J=1,NMODES
        DO 200 I=1,NDISPS
            IF (I .LE. MAXDOF)  READ(2,8000) SHAPES(I,J)
            IF (I .GT. MAXDOF)  READ(2,8000) DUMMY
200         CONTINUE
C     READ DOF ID FILE
        NDOFS(1) = 0
        DO 300 I=1,1000000
            IF (I .LE. MAXDOF)
     *          READ(3,11000,END=350) IDDOF(I,1),ICOMPA(I,1)
            IF (I .GT. MAXDOF)  READ(3,11000,END=350) IDUMMY,IDUMMC
            NDOFS(1) = NDOFS(1) + 1
300         CONTINUE
350     IF (NDOFS(1) .NE. NDISPS)  THEN
            TYPE 12000, NDISPS,NDOFS(1)
            IF (LPRINT)  PRINT 12000, NDISPS,NDOFS(1)
            CLOSE(UNIT=1)
            CLOSE(UNIT=2)
            CLOSE(UNIT=3)
            STOP
            ENDIF
        NDOFS(1) = MIN(NDOFS(1),MAXDOF)
        IF (NMODES .EQ. 1)  RETURN
        DO 500 J=2,NMODES
            NDOFS(J) = NDOFS(1)
            DO 400 I=1,NDOFS(J)
                IDDOF(I,J) = IDDOF(I,1)
                ICOMPA(I,J) = ICOMPA(I,1)
400             CONTINUE
500         CONTINUE
      CLOSE(UNIT=1)
      CLOSE(UNIT=2)
      CLOSE(UNIT=3)
      RETURN
C
1000  FORMAT(/,' ENTER ANALYTICAL LAMA MATRIX FILENAME: ',$)
1500  FORMAT(/,' ENTER ANALYTICAL LAMA MATRIX FILENAME: ',A35)
2000  FORMAT(A35)
3000  FORMAT(' ENTER ANALYTICAL MODE-SHAPE MATRIX FILENAME: ',$)
3500  FORMAT(' ENTER ANALYTICAL MODE-SHAPE MATRIX FILENAME: ',A35)
4000  FORMAT(' ENTER ANALYTICAL GRID POINT LIST FILENAME: ',$)
4500  FORMAT(' ENTER ANALYTICAL GRID POINT LIST FILENAME: ',A35)
5000  FORMAT(' FREQUENCY MATRIX ',A8,' HAS ',I6,' COLUMNS.'
     *        ' SHOULD BE 6 COLUMNS.')
6000  FORMAT(' **** WARNING: FREQUENCY VECTOR HAS ',I3,' ENTRIES.',
     *        /15X,'ONLY THE FIRST',I5,' WILL BE USED.')
7000  FORMAT(A8,2I8)
8000  FORMAT(E16.8)
9000  FORMAT(' **** WARNING: UNEQUAL NUMBER OF FREQUENCIES AND '
     *        'MODE SHAPES. ',
     *        /15X,'NUMBER OF FREQUENCIES:',I6,
```

```
     *        /15X,'NUMBER OF MODE SHAPES:',I6,
     *        /15X,'ONLY THE FIRST',I5,' WILL BE USED.')
10000 FORMAT(' **** WARNING: MODE-SHAPE VECTORS HAVE ',I3,' ENTRIES.',
     *        /15X,'ONLY THE FIRST',I6,' WILL BE USED.')
11000 FORMAT( I8,1X,I1)
12000 FORMAT(' **** ERROR: UNEQUAL NUMBER OF MODE-SHAPE POINTS AND ',
     *        'D.O.F. ID''S. ',I6,' & ',I6)
      END
```

```
      SUBROUTINE GETEXP(FREQEX,DAMP,ESYMM,DISPEX,MEX,
     *                                       IDDOFE,ICOMP,NDOFE,LPRINT)
C
C     GET EXPERIMENTAL FREQUENCY & DAMPING LISTS, MODE-SHAPE MATRIX,
C     AND D.O.F. LIST.
C
      COMMON /LIMITS/ MAXMODE,MAXDOF

      REAL FREQEX(MAXMODE),DAMP(MAXMODE),DISPEX(MAXDOF,MAXMODE)
      REAL DUMMY(5)
      INTEGER ESYMM(3,MAXMODE)
      INTEGER IDDOFE(MAXDOF,MAXMODE),ICOMP(MAXDOF,MAXMODE)
      INTEGER NDOFE(MAXMODE)
      CHARACTER*35 MSFILE
      LOGICAL LPRINT
C
C     PROCESS MODE-SHAPE FILES
          TYPE *, ' '    ! SKIP A LINE
          IF (LPRINT)  PRINT *, ' '
          MEX = 0
          DO 200 J=1,MAXMODE
C             GET NEXT EXPERIMENTAL MODE-SHAPE FILE
10                TYPE 1000
                  ACCEPT 2000, MSFILE
                  IF (MSFILE .EQ. 'NONE')  GOTO 250
                  OPEN(UNIT=10,NAME=MSFILE,TYPE='OLD',READONLY,ERR=10)
                  IF (LPRINT)  PRINT 1500, MSFILE
C             GET FREQUENCY, DAMPING & SYMMETRY FOR THIS MODE-SHAPE
                  READ(10,*) FREQEX(J)
                  READ(10,*) DAMP(J)
                  READ(10,*) (ESYMM(K,J),K=1,3)
C             GET MODE-SHAPE DISPLACEMENTS & D.O.F.'S
                  DO 100 I=1,MAXDOF
                      READ(10,*,END=150)
     *                          IDDOFE(I,J),ICOMP(I,J),DISPEX(I,J)
                      NDOFE(J) = I
100                   CONTINUE
                  PRINT 5000, MAXDOF,MAXDOF
150           MEX = J
              CLOSE(UNIT=10)
200           CONTINUE
          TYPE 6000, MAXMODE
          IF (LPRINT) PRINT 6000, MAXMODE
250   IF (LPRINT .AND. (MSFILE .EQ. 'NONE'))  PRINT 1500, MSFILE
      IF (MEX .EQ. 0)  THEN
          TYPE *,' **** ERROR: NO EXPERIMENTAL MODE-SHAPE REQUESTS'
          IF (LPRINT)  PRINT *,
     *         ' **** ERROR: NO EXPERIMENTAL MODE-SHAPE REQUESTS'
          STOP
          END IF
      RETURN
C
1000  FORMAT(' ENTER NEXT EXPERIMENTAL MODE-SHAPE FILENAME. ',
     *       '("NONE" IF NO MORE) : ',$)
1500  FORMAT(' ENTER NEXT EXPERIMENTAL MODE-SHAPE FILENAME. ',
     *       '("NONE" IF NO MORE) : ',A35)
2000  FORMAT(A35)
5000  FORMAT(' **** WARNING: MODE-SHAPE VECTOR HAS',I6,' OR MORE',
     *       ' ENTRIES.',/13X,'ONLY THE FIRST',I6,' WILL BE USED.')
6000  FORMAT(' **** NO MORE EXPERIMENTAL MODE-SHAPE FILES PERMITTED.'
     *       ' MAXIMUM IS ',I4)

END
```

```
      SUBROUTINE GETVEC(DISP,NDOF,VECT)
C
C     COPY VECTOR OF LENGTH NDOF FROM PERMANENT VECTOR TO
C     WORKING VECTOR.
C
      REAL DISP(*),VECT(*)
C
      IF (NDOF .EQ. 0)  RETURN
      DO 100 I=1,NDOF
         VECT(I) = DISP(I)
100      CONTINUE
      RETURN
      END
```

```
      SUBROUTINE INPSUM(NAN,FREQAN,MASS,STIFF,ASYMM,
     *                  MEX,FREQEX,DAMP,        ESYMM)
C
C     PRINT A SUMMARY OF THE ANALYTICAL & EXPERIMENTAL FREQUENCIES,
C     MASS, STIFFNESS, DAMPING, AND SYMMETRY VECTORS.
C
      REAL FREQAN(*),MASS(*),STIFF(*)
      REAL FREQEX(*),DAMP(*)
      INTEGER ASYMM(3,*),ESYMM(3,*)
      LOGICAL NOSYMA(3),YESYMA(3),NOSYME(3),YESYME(3),FIRST
      CHARACTER CDATE*9
      COMMON /WHEN/ CDATE
      DATA NOSYMA,YESYMA,NOSYME,YESYME/12*.FALSE./, FIRST/.TRUE./
C
C     PRINT HEADER
          PRINT 8000, CDATE
          PRINT 1000
C     PRINT ANALYTICAL SUMMARY
C         PRINT ANALYTICAL HEADER
              PRINT 2000
C         PRINT SUMMARY DATA
              PRINT 3000, (I,FREQAN(I),MASS(I),STIFF(I),
     *                    (ASYMM(J,I),J=1,3),  I=1,NAN)
C         CHECK FOR INCONSISTENT ANALYTICAL "UNDEFINED" SYMMETRY
              DO 200 IAXS=1,3
                  DO 100 MODE=1,NAN
                      IF (ASYMM(IAXS,MODE).EQ.0)   NOSYMA(IAXS)=.TRUE.
                      IF (ASYMM(IAXS,MODE).NE.0)   YESYMA(IAXS)=.TRUE.
100                   CONTINUE
                  IF (NOSYMA(IAXS).AND.YESYMA(IAXS))   PRINT 6000,  IAXS
200               CONTINUE
C     PRINT EXPERIMENTAL SUMMARY
C         PRINT EXPERIMENTAL HEADER
              PRINT 4000
C         PRINT SUMMARY DATA
              PRINT 5000, (I,FREQEX(I),DAMP(I),
     *                    (ESYMM(J,I),J=1,3),  I=1,MEX)
C         CHECK FOR INCONSISTENT EXPERIMENTAL "UNDEFINED" SYMMETRY
              DO 400 IAXS=1,3
                  DO 300 MODE=1,MEX
                      IF (ESYMM(IAXS,MODE).EQ.0)   NOSYME(IAXS)=.TRUE.
                      IF (ESYMM(IAXS,MODE).NE.0)   YESYME(IAXS)=.TRUE.
300                   CONTINUE
                  IF (NOSYME(IAXS).AND.YESYME(IAXS))   PRINT 6000,  IAXS
400               CONTINUE
C     CHECK FOR INCONSISTENT ANA VS. EXP "UNDEFINED" SYMMETRY
          DO 500 IAXS=1,3
              IF ( (NOSYMA(IAXS).AND.YESYME(IAXS))  .OR.
     *             (YESYMA(IAXS).AND.NOSYME(IAXS))       )    THEN
                  IF (FIRST)   PRINT 7000
                  FIRST = .FALSE.
                  PRINT 6000,  IAXS
                  END IF
500           CONTINUE
      RETURN
C
1000  FORMAT(1H0,'2. SUMMARY OF FREQUENCY, MASS, STIFFNESS,',
     *            ' DAMPING, & SYMMETRY:')
2000  FORMAT(//,5X,'ANALYTICAL MODES:'
     *       /1H0,4X,'MODE ','    FREQUENCY      ','      MASS        ',
     *                '     STIFFNESS      ','    SYMMETRY        ',
```

```
     *            /,5X,'____',4(1X,'________________'))
3000  FORMAT(<NAN>(/5X,I3,1X,3(1PE16.8),3X,3I3),/)
4000  FORMAT(//,5X,'EXPERIMENTAL MODES:',
     *        /1H0,4X,'MODE FREQUENCY   DAMPING    SYMMETRY',
     *            /,5X,'____',3(1X,'__________'))
5000  FORMAT(<MEX>(/5X,I3,1X,2(1PE11.3),1X,3I3),/)
6000  FORMAT(10X,'***WARNING: INCONSISTENT UNDEFINED SYMMETRY',
     *             ' FOR PLANE',I2)
7000  FORMAT(//,5X,'ANALYTICAL VS. EXPERIMENTAL SYMMETRY:')
8000  FORMAT(1H1,'NASTRAN MODAL ANALYSIS - MODAL SURVEY STATISTICAL ',
     *        'CORRELATION', /1X,A9)
      END
```

```
      SUBROUTINE  INTERSECT(AVECT,EVECT,IDDOFA,IDDOFE,ICOMPA,ICOMPE,
     *                      NDOFA,NDOFE,IDDOFI,ICOMPI,NDOFI,
     *                      ASYMM,ESYMM,LSYMM)
C
C     COPY INTERSECTION SET OF VECTORS IDDOFA & IDDOFE TO VECTOR
C     IDDOFI.   NDOFI IS THE LENGTH OF THE INTERSECTION SET.
C     REDUCE AVECT & EVECT VECTORS TO CONTAIN ONLY THE CORRESPONDING
C     INTERSECTION SET OF TERMS, I.A.W. THE MASTER SET IDDOFI.
C     IF THE INTERSECTION SET IS NULL (NO EQUIVALENT D.O.F ID'S),
C     THE VALUE RETURNED FOR NDOFI IS 0 .
C     IF SYMMETRY IS TO BE CONSIDERED, AND ANALYTICAL & EXPERIMENTAL
C     SYMMETRY ARE NOT THE SAME, THE VALUE RETURNED FOR NDOFI IS 0 .
C
      REAL AVECT(*),EVECT(*)
      INTEGER IDDOFA(*),IDDOFE(*),IDDOFI(*)
      INTEGER ICOMPA(*),ICOMPE(*),ICOMPI(*)
      INTEGER ASYMM(3),ESYMM(3)
      LOGICAL LSYMM
C
      IA = 1
      IE = 1
      NDOFI = 0
      IF (LSYMM)  THEN
C         SYMMETRY IS TO BE CONSIDERED:
C         IF A & E ARE UNLIKE, LEAVE INTERSECTION SET AS NULL
              IF (ASYMM(1) .NE. ESYMM(1))  RETURN
              IF (ASYMM(2) .NE. ESYMM(2))  RETURN
              IF (ASYMM(3) .NE. ESYMM(3))  RETURN
          END IF
C
      DO 100 I=1,NDOFA+NDOFE
          IF ((IA.GT.NDOFA) .OR. (IE.GT.NDOFE))  GOTO 150
          IF ( (IDDOFA(IA) .EQ. IDDOFE(IE)) .AND.
     *         (ICOMPA(IA) .EQ. ICOMPE(IE))       )  THEN
              NDOFI = NDOFI + 1
              IDDOFI(NDOFI) = IDDOFA(IA)
              ICOMPI(NDOFI) = ICOMPA(IA)
              AVECT(NDOFI) = AVECT(IA)
              EVECT(NDOFI) = EVECT(IE)
              IA = IA + 1
              IE = IE +1
          ELSE IF (IDDOFA(IA) .LT. IDDOFE(IE))  THEN
              IA = IA + 1
          ELSE IF (IDDOFE(IE) .LT. IDDOFA(IA))  THEN
              IE = IE + 1
          END IF
100       CONTINUE
150   RETURN
      END
```

```
      SUBROUTINE MATCH(CORREL,NAN,MEX,NAFIT,MEFIT)
C
      COMMON /LIMITS/ MAXMODE,MAXDOF

      DIMENSION CORREL(MAXMODE,MAXMODE),NAFIT(MAXMODE),MEFIT(MAXMODE)
C
C     DETERMINE BEST EXPERIMENTAL MODE-SHAPE MATCH (HIGHEST
C     CORRELATION COEFFICIENT) FOR EACH ANALYTICAL MODE-SHAPE.
C
          DO 200 N=1,NAN
              BEST = -1.0
              DO 100 M=1,MEX
                  IF (ABS(CORREL(N,M)) .GT. BEST)  THEN
                      BEST = ABS(CORREL(N,M))
                      NAFIT(N) = M
                      END IF
100               CONTINUE
200           CONTINUE
C
C     DETERMINE BEST ANALYTICAL MODE-SHAPE MATCH (HIGHEST
C     CORRELATION COEFFICIENT) FOR EACH EXPERIMENTAL MODE-SHAPE.
C
          DO 400 M=1,MEX
              BEST = -1.0
              DO 300 N=1,NAN
                  IF (ABS(CORREL(N,M)) .GT. BEST)  THEN
                      BEST = ABS(CORREL(N,M))
                      MEFIT(M) = N
                      END IF
300               CONTINUE
400           CONTINUE
      RETURN
      END
```

```
      SUBROUTINE OPTIONS(LSYMM,PRTAN,PRTEX,RTHRESH,LPRINT)
C
C     DETERMINE SYMMETRY, MODE-SHAPE DUMP, & THRESHOLD OPTIONS
C
      LOGICAL LSYMM,PRTAN,PRTEX,LPRINT
      CHARACTER*1 YESNO
C
C     DETERMINE IF SYMMETRY IS TO BE CONSIDERED
          TYPE 1000
          ACCEPT 2000, YESNO
          IF (YESNO .EQ. 'Y')  LSYMM = .TRUE.
          IF (LPRINT)  PRINT 1500, YESNO
C     DETERMINE IF ANALYTICAL AND/OR EXPERIMENTAL MODE-SHAPE
C                                          VECTORS ARE TO BE PRINTED
          TYPE *, ' '      !SKIP A LINE
          IF (LPRINT)  PRINT *, ' '
          TYPE 3000, 'ANALYTICAL'
          ACCEPT 2000, YESNO
          IF (YESNO .EQ. 'Y')  PRTAN = .TRUE.
          IF (LPRINT)  PRINT 3500, 'ANALYTICAL', YESNO
          TYPE 3000, 'EXPERIMENTAL'
          ACCEPT 2000, YESNO
          IF (YESNO .EQ. 'Y')  PRTEX = .TRUE.
          IF (LPRINT)  PRINT 3500, 'EXPERIMENTAL', YESNO
C     GET RELATIVE DEVIATION THRESHOLD.  DEFAULT IS 5.0 %
          RTHRESH = .05
          TYPE 4000, RTHRESH
          ACCEPT *, RTHRESH
          TYPE 5000, RTHRESH
          IF (LPRINT)  THEN
              PRINT 4500, .05, RTHRESH
              PRINT 5000, RTHRESH
              END IF
      RETURN
C
1000  FORMAT(/,' IS ANALYTICAL VS. EXPERIMENTAL SYMMETRY TO BE ',
     *          'CONSIDERED? (Y OR N): ',$)
1500  FORMAT(/,' IS ANALYTICAL VS. EXPERIMENTAL SYMMETRY TO BE ',
     *          'CONSIDERED? (Y OR N): ',A1)
2000  FORMAT(A1)
3000  FORMAT(' PRINT ',A12,' MODE-SHAPE VECTORS? (Y OR N): ',$)
3500  FORMAT(' PRINT ',A12,' MODE-SHAPE VECTORS? (Y OR N): ',A1)
4000  FORMAT(//' RELATIVE DEVIATIONS GREATER THAN A THRESHOLD',
     *        ' VALUE WILL BE PRINTED.',
     *       /' THE DEFAULT THRESHOLD IS ',F6.3,
     *       /' ENTER DESIRED THRESHOLD. "," FOR DEFAULT: ',$)
4500  FORMAT(//' RELATIVE DEVIATIONS GREATER THAN A THRESHOLD',
     *        ' VALUE WILL BE PRINTED.',
     *       /' THE DEFAULT THRESHOLD IS ',F6.3,
     *       /' ENTER DESIRED THRESHOLD. "," FOR DEFAULT: ',F6.3)
5000  FORMAT('0RELATIVE DEVIATIONS GREATER THAN ',2PF7.2,
     *        ' % WILL BE PRINTED.')
      END
```

```
      SUBROUTINE PRINT(N1,FREQ1,N2,FREQ2,NDOF,CORREL,C,S,RMSA,RMSE,
     *                 DIFF,DIFABS,IDDOF,ICOMP,RTHRESH)
C
C     PRINT CORRELATION INFORMATION FOR MODES N1,N2 COMPARISON
C
      DIMENSION DIFF(*),DIFABS(*),IDDOF(*),ICOMP(*)
      DIMENSION ID(30000)
C
C     DETERMINE MAXIMUM DIFFERENCE & DIFFS > THRESHOLD
          DIFMAX = 0.0
          IDMAX = 0
          NDIFF = 0
          IF (NDOF .GT. 0)  THEN
              DIFMAX = ABS(DIFF(1))
              IDMAX = IDDOF(1)
              DO 100 N=1,NDOF
                  IF (ABS(DIFF(N)) .GT. DIFMAX)  THEN
                      DIFMAX = ABS(DIFF(N))
                      IDMAX = IDDOF(N)
                      END IF
                  IF (ABS(DIFF(N)) .GT. RTHRESH)  THEN
                      NDIFF = NDIFF + 1
                      ID(NDIFF) = N
                      END IF
100               CONTINUE
              END IF
C
C     PRINT SUMMARY LINE FOR THIS MATCHED PAIR
          PRINT 1000, N1,FREQ1,N2,FREQ2,NDOF,CORREL,C,S,
     *                RMSA,RMSE,DIFMAX,IDMAX
          IF (NDOF .EQ. 0)  RETURN
C
C     PRINT ALL DIFFS > THRESHOLD , DEFAULT IS  5%
          IF (NDIFF .GT. 0)  THEN
              PRINT 2000, RTHRESH,(IDDOF(ID(N)), ICOMP(ID(N)),
     *                             DIFF(ID(N)), N=1,NDIFF)
              PRINT 3000, (IDDOF(ID(N)), ICOMP(ID(N)), DIFABS(ID(N)),
     *                    N=1,NDIFF)
              END IF
      RETURN
C
1000  FORMAT('0',I6,2(F15.6,I7),4X,F9.3,2X,5(1PE11.3),I9)
2000  FORMAT(21X,'RELATIVE DEVIATIONS (XA/RMSA-XE/RMSE) > ',
     *       2PF7.2,'% : (GRID ID/DEVIATION)',
     *       20(/21X,5(I9,'-',I1,'/',1PE10.3)))
3000  FORMAT(21X,'SCALED DIFFERENCES ( (XA-XE)/S ) > THRESHOLD :',
     *       ' (GRID ID/DIFFERENCE)'
     *       20(/21X,5(I9,'-',I1,'/',1PE10.3)))
      END
```

```
      SUBROUTINE PRINTPHI(NAN,NDOFA,IDDOFA,ICOMPA,DISPAN,PRTAN,
     *                    MEX,NDOFE,IDDOFE,ICOMPE,DISPEX,PRTEX)
C
      COMMON /LIMITS/ MAXMODE,MAXDOF
      COMMON /WHEN/ CDATE

      REAL DISPAN(MAXDOF,MAXMODE),DISPEX(MAXDOF,MAXMODE)
      INTEGER IDDOFA(MAXDOF,MAXMODE), IDDOFE(MAXDOF,MAXMODE)
      INTEGER ICOMPA(MAXDOF,MAXMODE), ICOMPE(MAXDOF,MAXMODE)
      INTEGER NDOFA(MAXMODE), NDOFE(MAXMODE)
      LOGICAL PRTAN,PRTEX
      CHARACTER CDATE*9
C
      IF (PRTAN)  THEN  ! PRINT ANALYTICAL MODE-SHAPE VECTORS
C         PRINT HEADER
              PRINT 5000, CDATE
              PRINT 1000
C         PRINT ALL VECTORS
              DO 100 I=1,NAN
                  PRINT 2000,  I,(IDDOFA(J,I),ICOMPA(J,I),DISPAN(J,I),
     *                            J=1,NDOFA(I))
100               CONTINUE
          END IF
C
      IF (PRTEX)  THEN  ! PRINT EXPERIMENTAL MODE-SHAPE VECTORS
C         PRINT HEADER
              PRINT 5000, CDATE
              PRINT 3000
C         PRINT ALL VECTORS
              DO 200 I=1,MEX
                  PRINT 4000,  I,(IDDOFE(J,I),ICOMPE(J,I),DISPEX(J,I),
     *                            J=1,NDOFE(I))
200               CONTINUE
          END IF
      RETURN
C
1000  FORMAT(1H0,'6. ANALYTICAL MODE-SHAPE VECTORS ',
     *          '(GRID PT/DISPLACEMENT)')
2000  FORMAT(1H0,'ANALYTICAL MODE',I4,
     *       200(/5X,5(I9,'-',I1,'/',1PE10.3)))
3000  FORMAT(1H0,'7. EXPERIMENTAL MODE-SHAPE VECTORS ',
     *          '(GRID PT/DISPLACEMENT)')
4000  FORMAT(1H0,'EXPERIMENTAL MODE',I4,
     *       200(/5X,5(I9,'-',I1,'/',1PE10.3)))
5000  FORMAT(1H1,'NASTRAN MODAL ANALYSIS - MODAL SURVEY STATISTICAL ',
     *       'CORRELATION', /1X,A9)
      END
```

```
      SUBROUTINE RMSDIFF(AVECT,EVECT,NDOFI,RMSA,RMSE,CORREL,
     *                   S,DIFF,DIFABS)
C
C     CALCULATE RELATIVE (TO RMS) DIFFERENCE BETWEEN ANALYTICAL &
C     EXPERIMENTAL DISPLACEMENT FOR EACH GRID POINT.
C     CALCULATE ABSOLUTE DIFFERENCE SCALED TO ANALYTICAL.
C
      REAL AVECT(*),EVECT(*),DIFF(*),DIFABS(*)
C
      IF (NDOFI .EQ. 0)  RETURN
      DO 100 I=1,NDOFI
          IF (CORREL .GE. 0.0)  THEN
              DIFF(I) = 0.0
              IF ( (RMSA .NE. 0.0)  .AND.  (RMSE .NE. 0.0) )
     *            DIFF(I) = AVECT(I)/RMSA - EVECT(I)/RMSE
              DIFABS(I) = 0.0
              IF (S .NE. 0.0)
     *            DIFABS(I) = (AVECT(I) - EVECT(I)) / S
          ELSE IF (CORREL .LT. 0.0)  THEN
C             CORRECT FOR 180-DEGREE PHASE SHIFT
              DIFF(I) = 0.0
              IF ( (RMSA .NE. 0.0)  .AND.  (RMSE .NE. 0.0) )
     *            DIFF(I) = AVECT(I)/RMSA + EVECT(I)/RMSE
              DIFABS(I) = 0.0
              IF (S .NE. 0.0)
     *            DIFABS(I) = (AVECT(I) + EVECT(I)) / S
          END IF
100       CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SWAP(I,J)
C
C     INTERCHANGE THE VALUES IN TWO 4-BYTE VARIABLES
C
      ITEMP = I
      I = J
      J = ITEMP
      RETURN
      END
```

READING CORRELATION RESULTS

The printed output listing generated by interactive execution of the statistical correlation program STATCORR contains seven functionally-defined sections. Sections 6 and 7 are optionally requested by the user.

Section 1 echos the interactive dialog between the user and the program, providing a record of the analytical and experimental data files used in the correlation run, symmetry consideration, requests for mode shape vector listings, and requested threshold for deviation reporting.

Section 2 provides a summary which describes the analytic and experimental modes being compared, including frequency, mass, stiffness, and symmetry characteristics for each analytical mode, and frequency, damping, and symmetry characteristics for each experimental mode. Inconsistencies in undefined symmetry are reported in this section: where undefined symmetry with respect to any plane exists (a "0" symmetry code), symmetry should be undefined with respect to that plane for all modes, both analytical and experimental. If symmetry has been defined for any mode (a "+1" or "-1" symmetry code), with respect to a particular plane, but has not been defined with respect to that plane for another mode, a warning message will be printed for the user's benefit, so that he may take appropriate action, if required.

Section 3 reports the correlation coefficients computed for all mode comparisons. Each analytical mode is compared with each experimental mode. The correlation coefficients are printed in a matrix format, with each row referring to a particular analytical mode, and each column referring to a particular experimental mode. The integers labeling the rows and columns refer to the mode identification numbers reported in Section 2 (i.e., row 1 refers to analytical mode 1 described in Section 2; column 3 refers to experimental mode 3). As an example, the correlation coefficient shown in (row 2, column 3) is calculated (according to eqn(6)) by comparing analytical mode shape 2 with experimental mode shape 3 .
The magnitudes of the correlation coefficients range from 0.0 (no correlation) to 1.0 (perfect correlation). Negative correlation coefficients indicate a phase reversal (180-degree phase shift) between the analytical and experimental modes being compared. In determining which modes compare most favorably (highest correlation coefficient), absolute values of the correlation coefficients are compared.
It is mathematically possible for a 0.0 correlation coefficient to be calculated, but from a practical standpoint this is not a likely occurence. However, exact zero values may be reported in Section 3 when symmetry characteristics are considered (in accordance with the user's affirmative response to the symmetry query in Section 1); in any case where the analytical and experimental modes being compared have unlike symmetry, that correlation coefficient is set equal to 0.0, rather than being calculated.

Section 4 shows a summary of the correlation results for the analytical modes. For each analytical mode, the experimental mode that best matches it (based on the correlation coefficients listed in Section 3) is shown.
This summary is reported in a table format, where each row of the table corresponds to a different analytical mode.
Columns 1 & 2 identify the analytical mode and its characteristic frequency, and columns 3 & 4 give the mode number and frequency of the experimental mode which compares most favorably with this analytical mode. These mode numbers and frequencies correspond to those reported in Section 2.
Column 5 shows the number of sample points (grid points on the structure) whose displacements were used to calculate the statistical data reported in this row.
The Correlation Coefficient, Normalization Factor(CA) and Standard Error(S), calculated according to equations (6), (8), and (9), respectively, for this analytical and experimental mode pair, are given in columns 6, 7, and 8.
The RMS values for the two mode shapes are shown in columns 9 and 10, to indicate the respective scaling used to determine the relative differences between the analytical and experimental mode shape displacements at individual grid points. Column 11 shows the maximum relative difference encountered for this mode shape pair, and column 12 shows the grid point at which this occurs.
If there are any grid points which have (for a particular analytical mode as compared to its best experimental match) relative deviations greater than the percentage specified by the user (in Section 1), those grid points and the fractional values of their relative deviations are listed immediately following the row which gives the statistical report for that anaytical mode. A report is also given of the standard errors at those same grid point locations.
When symmetry characteristics are being compared by STATCORR, it is possible that there will be no experimental modes with the same symmetry parameters as a particular analytical mode. Should this occur, all the statisical data will be reported as zeroes in the row corresponding to that analytical mode.

Section 5 shows a summary of the correlation results for the experimental modes. For each experimental mode, the analytical mode that best matches it (based on the correlation coefficients reported in Section 3) is shown. This summary has exactly the same format as Section 4, except that columns 1 & 2 identify the experimental mode and its characteristic frequency, while columns 3 & 4 give the mode number and frequency of the analytical mode which best compares with that experimental mode.

ORIGINAL PAGE IS
OF POOR QUALITY

Section 6 (optionally requested by the user during the Section 1 dialog) provides a listing of the mode-shape displacements (eigenvectors) for all anaytical modes. This listing shows, for each mode, all of the grid point identification numbers and the displacement associated with each grid point.

Section 7 (optionally requested by the user during the Section 1 dialog) provides a listing of the mode-shape displacements for all experimental modes. This listing is in the same format as Section 6.

LAMA

The table of modal frequencies, masses, and stiffnesses output from NASTRAN are converted by processer program LAMA from table format to matrix format to satisfy the input needs of STATCORR. LAMA operates interactively and prompts the user for the input file name and the output file name.

CTCB

LAMA:

FOR:2

CTCB

```
      PROGRAM LAMATABLE
C
      PARAMETER (MAXEIG = 300)

      CHARACTER LAMA*8
      REAL FREQ(MAXEIG),GMASS(MAXEIG),STIFF(MAXEIG),SYMM(3)
      REAL REC2(7*MAXEIG)
C
      DATA LUN_P, LUN_M  /1, 2/    !! PUNCH & MATRIX FILE LOG UNIT #'S
C
C     POSITION PUNCH FILE TO 1ST CARD OF RECORD 2 OF LAMA TABLE
          CALL FINDREC2(LUN_P,LAMA)
C     READ RECORD 2 OF LAMA TABLE INTO ARRAY REC2
          CALL READREC2(REC2,NVALUES,LUN_P,MAXEIG)
C     EXTRACT FREQUENCY, MASS, & STIFFNESS VECTORS FROM REC2
          NEIGENS = NVALUES/7
          DO 100 I=1,NEIGENS
              FREQ(I)  = REC2(7*I-2)
              GMASS(I) = REC2(7*I-1)
              STIFF(I) = REC2(7*I)
100           CONTINUE
C     DO ASCENDING SORT OF FREQ VECTOR; SLAVE SORT MASS & STIFF VECTORS
          CALL SORT(NEIGENS,FREQ,GMASS,STIFF)
C     CHECK PUNCH FILE FOR SYMMETRY PARAMETERS
          CALL READSYM(SYMM,LUN_P)
C     CLOSE PUNCH FILE
          CLOSE(UNIT=LUN_P)
C     WRITE MATRIX FILE
          CALL WRITEMTX(NEIGENS,FREQ,GMASS,STIFF,SYMM,LUN_M,LAMA)
      STOP
      END
```

```
      SUBROUTINE FINDREC2(LUN_P,LAMA)
C
      CHARACTER PFILE*80,LAMA*8,LINE*80
C
C     OPEN PUNCH FILE
10          TYPE 1000, 'ENTER INPUT PUNCH FILE NAME:'
            ACCEPT 2000, PFILE
            OPEN(UNIT=LUN_P,NAME=PFILE,TYPE='OLD',READONLY,ERR=10)
C     POSITION FILE TO LAMA TABLE
            TYPE 1000, 'ENTER NAME OF LAMA TABLE:'
            ACCEPT 3000, LAMA
            DO 100 I=1,1000000
               READ(LUN_P,2000,END=125) LINE
               IF ((LINE(1:4).EQ.'DTI*') .AND. (LINE(9:16).EQ.LAMA))
     *            GOTO 150    !! LAMA TABLE FOUND
100            CONTINUE
C           TABLE NOT FOUND; START OVER
125            TYPE *, '**** LAMA TABLE NOT FOUND'
               CLOSE(UNIT=LUN_P)
               GOTO 10
C     POSITION PUNCH FILE TO RECORD 2 OF LAMA TABLE
150         DO 200 I=1,1000000
               READ(LUN_P,4000,END=225) LINE(1:24),IREC
               IF ((LINE(1:4).EQ.'DTI*') .AND. (LINE(9:16).EQ.LAMA)
     *            .AND. (IREC.EQ.2))  GOTO 250    !! RECORD 2 FOUND
200            CONTINUE
C           RECORD 2 NOT FOUND; START OVER
225            TYPE *, '**** RECORD 2 OF LAMA TABLE NOT FOUND'
               CLOSE(UNIT=LUN_P)
               GOTO 10
C           POSITION FILE TO REREAD 1ST CARD OF RECORD 2
250            BACKSPACE LUN_P
         RETURN
C
1000     FORMAT(1X,A30,1X,$)
2000     FORMAT(A80)
3000     FORMAT(A8)
4000     FORMAT(A24,I16)
         END
```

```
      SUBROUTINE READREC2(IREC2,NVALUES,LUN_P,MAXEIG)
C
      INTEGER IREC2(7*MAXEIG),ITEMP(4)
      CHARACTER*6 CTEMP(4)
C
C     READ 1ST CARD OF LAMA TABLE RECORD 2
         READ(LUN_P,1000) IREC2(1),IREC2(2)
         NVALUES = 2
C     READ REMAINING CARDS UNTIL 'ENDREC' FOUND
         DO 200 I=1,1000000
            READ(LUN_P,2000,END=225) (CTEMP(J),ITEMP(J),J=1,4)
            DO 100 J=1,4
               IF (CTEMP(J) .EQ. 'ENDREC')  GOTO 250
               NVALUES = NVALUES + 1
               IREC2(NVALUES) = ITEMP(J)
               IF (CTEMP(J)(6:6) .EQ. '-')
     *            IREC2(NVALUES) = -IREC2(NVALUES)
               IF (NVALUES .EQ. 7*MAXEIG)  GOTO 225
100            CONTINUE
200         CONTINUE
C        END OF RECORD 2 OF LAMA TABLE NOT FOUND
225         TYPE 3000, NVALUES
            PAUSE 'TYPE "CONTINUE" OR "STOP"'
C     CHECK TO SEE IF # OF RECORD 2 ENTRIES IS DIVISIBLE BY 7
250      IF (MOD(NVALUES,7) .NE. 0)  TYPE 4000, NVALUES
      RETURN
C
1000  FORMAT(40X,2I16)
2000  FORMAT(8X,4(A6,I10))
3000  FORMAT(1X,'*** WARNING: END OF RECORD 2 NOT REACHED.',
     *       /1X,'              NUMBER OF ENTRIES READ: ',I4)
4000  FORMAT(1X,'*** WARNING: # OF RECORD 2 ENTRIES',I4,
     *       ' NOT DIVISIBLE BY 7')
      END
```

```
      SUBROUTINE READSYM(SYMM,LUN_P)
C
      CHARACTER*80 RECORD
      REAL SYMM(3)
      LOGICAL LSYM1,LSYM2,LSYM3
C
      REWIND LUN_P   !START SEARCH FROM TOP OF FILE
C     SET SYMMETRY DEFAULTS
          SYMM(1) = 0.0
          SYMM(2) = 0.0
          SYMM(3) = 0.0
C     SEARCH PUNCH FILE FOR SYMMETRY DEFINITION RECORDS
          DO 100 I=1,1000000
              READ(LUN_P,1000,END=150) RECORD
              IF (RECORD(21:28) .EQ. 'SYM1PLAN')  THEN
                  LSYM1 = .TRUE.
                  DECODE(2,2000,RECORD(47:48)) SYMM(1)
                  TYPE *, 'PLANE 1 SYMMETRY:', SYMM(1)
              ELSE IF (RECORD(21:28) .EQ. 'SYM2PLAN')  THEN
                  LSYM2 = .TRUE.
                  DECODE(2,2000,RECORD(47:48)) SYMM(2)
                  TYPE *, 'PLANE 2 SYMMETRY:', SYMM(2)
              ELSE IF (RECORD(21:28) .EQ. 'SYM3PLAN')  THEN
                  LSYM3 = .TRUE.
                  DECODE(2,2000,RECORD(47:48)) SYMM(3)
                  TYPE *, 'PLANE 3 SYMMETRY:', SYMM(3)
              END IF
100           CONTINUE
C     REPORT NON-DEFINED SYMMETRY
150       IF (.NOT. LSYM1)  TYPE 3000, '1'
          IF (.NOT. LSYM2)  TYPE 3000, '2'
          IF (.NOT. LSYM3)  TYPE 3000, '3'
      RETURN
C
1000  FORMAT(A80)
2000  FORMAT(F2.0)
3000  FORMAT(1X,'PLANE ',A1,' SYMMETRY NOT SPECIFIED')
      END
```

```
      SUBROUTINE SORT(N,FREQ,GMASS,STIFF)
C
      REAL FREQ(N),GMASS(N),STIFF(N)
C
C     PERFORM ASCENDING SORT OF FREQ VECTOR
C     SLAVE SORT GMASS AND STIFF VECTORS
C
      IF (N .EQ. 1)  RETURN
C
      DO 100 I=1,N-1
      DO 100 K=I+1,N
         IF (FREQ(K) .LT. FREQ(I))  THEN
               TEMP = FREQ(I)
               FREQ(I) = FREQ(K)
               FREQ(K) = TEMP
               TEMP = GMASS(I)
               GMASS(I) = GMASS(K)
               GMASS(K) = TEMP
               TEMP = STIFF(I)
               STIFF(I) = STIFF(K)
               STIFF(K) = TEMP
               END IF
100      CONTINUE
      RETURN
      END
```

```
      SUBROUTINE WRITEMTX(N,FREQ,GMASS,STIFF,SYMM,LUN_M,LAMA)
C
      CHARACTER MFILE*40,LAMA*8
      REAL FREQ(N),GMASS(N),STIFF(N),SYMM(3)
C
C     OPEN MATRIX FILE
          TYPE 1000, 'ENTER OUTPUT MATRIX FILENAME:'
          ACCEPT 2000, MFILE
          OPEN(UNIT=LUN_M,NAME=MFILE,TYPE='NEW',
     *          CARRIAGECONTROL='LIST')
C     WRITE MATRIX FILE HEADER
          IROWS = N
          JCOLS = 6
          WRITE(LUN_M,3000) LAMA,IROWS,JCOLS
C     WRITE VECTORS AS THREE COLUMNS OF MATRIX
          WRITE(LUN_M,4000) FREQ     !! COLUMN 1
          WRITE(LUN_M,4000) GMASS    !! COLUMN 2
          WRITE(LUN_M,4000) STIFF    !! COLUMN 3
          WRITE(LUN_M,4000) (SYMM(1),I=1,N)   !! COLUMN 4
          WRITE(LUN_M,4000) (SYMM(2),I=1,N)   !! COLUMN 5
          WRITE(LUN_M,4000) (SYMM(3),I=1,N)   !! COLUMN 6
      CLOSE(UNIT=LUN_M)
      RETURN
C
 1000 FORMAT(1X,A30,1X,$)
 2000 FORMAT(A40)
 3000 FORMAT(A8,2I8)
 4000 FORMAT(1PE16.8)
      END
```

UNPACKDMI

The data blocks of mode shapes and component degrees of freedom as output from NASTRAN are in a special format called DMI. This format is not acceptable to STATCORR, so these data must be reformatted. The pre-processor program UNPACKDMI is used to take modes and freedoms out of DMI format. The output of the modes from this program will be in final form for STATCORR. The output of freedoms will need further processing by another pre-processor. UNPACKDMI operates interactively and prompts the user for responses.

NEXUS

UNPACKDMI.

FOR:1

NEXUS

```
      PROGRAM UNPACKDMI
      PARAMETER (MAXMTXSIZ = 100000)
      CHARACTER  MATNAM*8, FLAG*1, CONTID*3
      REAL ARRAY(MAXMTXSIZ)
      LOGICAL ALL, EOF
      DATA LUN  / 10 /
C
      DO 500 NFILE=1,1000000
C         GET NEXT FILENAME
              CALL GETFIL(LUN,*499)
          ALL = .FALSE.
          EOF = .FALSE.
          DO 400 NMATRX=1,1000000
C             GET NEXT MATRIX NAME
                  IF (.NOT. ALL) THEN
                      TYPE 1000
                      ACCEPT 2000, MATNAM
                      IF (MATNAM .EQ. '*ALL     ')  ALL = .TRUE.
                      END IF
C             SEARCH FILE FOR MATRIX & GET DIMENSIONS
                  CALL FNDMTX(MATNAM,MROWS,NCOLS,CONTID,LUN,ALL,EOF)
                  IF (EOF)  GOTO 399
                  IF (MROWS*NCOLS .GT. MAXMTXSIZ)  THEN
                      TYPE *,'MATRIX IS TOO LARGE', MATNAM,MROWS,NCOLS
                      TYPE *,'INCREASE PARAMETER MAXMTXSIZ IN UNPACKDMI'
                      GOTO 399
                      ENDIF
C             UNPACK MATRIX INTO ARRAY
                  CALL UNPACK(ARRAY,MROWS,NCOLS,CONTID,LUN,ALL,EOF)
C             REDUCE SQUARE DIAGONAL MATRIX TO VECTOR ?
                  IF (MROWS .EQ. NCOLS)  THEN
                      TYPE 6000, MATNAM
                      ACCEPT 4000, FLAG
                      IF (FLAG .EQ. 'Y')
     *                    CALL REDUCE(MATNAM,ARRAY,MROWS,NCOLS)
                      END IF
C             WRITE UNPACKED MATRIX TO NEW FILE
                  CALL WRTMTX(MATNAM,ARRAY,MROWS,NCOLS)
C             IS ANOTHER MATRIX IN THIS FILE DESIRED ?
  399             IF (ALL .AND. EOF)  GOTO 450
                  IF (.NOT. ALL)  THEN
                      TYPE 3000
                      ACCEPT 4000, FLAG
                      IF (FLAG .NE. 'Y')  GOTO 450
                      END IF
  400         CONTINUE
C         NO MORE MATRICES DESIRED FROM THIS FILE
  450         CLOSE (UNIT=LUN)
C         IS ANOTHER FILE DESIRED ?
  499         TYPE 5000
              ACCEPT 4000, FLAG
              IF (FLAG .NE. 'Y')  GOTO 550
  500     CONTINUE
  550 STOP 'NO MORE FILES REQUESTED.'
C
 1000 FORMAT(' ENTER MATRIX NAME. ("*ALL" FOR ALL MATRICES): ',$)
 2000 FORMAT(A8)
 3000 FORMAT(' DO YOU WANT ANOTHER MATRIX IN THIS FILE ? (Y OR N): ',$)
 4000 FORMAT(A1)
 5000 FORMAT(' DO YOU WANT ANOTHER FILE ? (Y OR N): ',$)
 6000 FORMAT(' REDUCE DIAG MATRIX ',A8,' TO A VECTOR ? (Y OR N): ',$)
      END
```

```
      SUBROUTINE FNDMTX(MATNAM,MROWS,NCOLS,CONTID,LUN,ALL,EOF)
      CHARACTER MATNAM*8, RECORD*80, FLD1TO3*24, CONTID*3
      LOGICAL ALL, EOF
C
      IF (.NOT. ALL)  THEN
          REWIND LUN
          EOF = .FALSE.
          END IF
C
C     SEARCH FOR DESIRED MATRIX & GET DIMENSIONS
          FLD1TO3 = 'DMI     '//MATNAM//'        0'
          DO 100 I=1,1000000
              READ(LUN,1000,END=199) RECORD
              IF (ALL)  FLD1TO3(9:16) = RECORD(9:16)
              IF (RECORD(1:24) .EQ. FLD1TO3)  THEN
                  IF (ALL)  MATNAM = RECORD(9:16)
                  READ(RECORD,2000) MROWS, NCOLS, CONTID
                  CONTID(1:1) = '*'
                  GOTO 150
              ENDIF
  100         CONTINUE
  150     RETURN
C
C     END OF FILE; NEXT MATRIX NOT FOUND
  199     EOF = .TRUE.
          IF (ALL)  TYPE *, 'END OF FILE REACHED'
          IF (.NOT. ALL)  TYPE *, MATNAM, 'MATRIX HEADER CARD NOT FOUND'
          RETURN
C
 1000 FORMAT(A80)
 2000 FORMAT(56X,2I8,A3)
      END
```

```
      SUBROUTINE GETFIL(LUN,*)
      CHARACTER FILNAM*40
C
C     GET NEXT FILENAME
          TYPE 1000
          ACCEPT 2000, FILNAM
C     OPEN FILE
          OPEN(UNIT=LUN,NAME=FILNAM,TYPE='OLD',READONLY,ERR=199)
          RETURN
C
C     NEW DMI FILE NOT FOUND
199       TYPE *, FILNAM, 'FILE NOT FOUND'
          RETURN 1
C
1000  FORMAT(' ENTER NAME OF NEXT DMI FILE: ',$)
2000  FORMAT(A40)
      END
```

```
      SUBROUTINE REDUCE(MATNAM,MATRIX,MROWS,NCOLS)
      CHARACTER*8 MATNAM
      REAL MATRIX(MROWS,NCOLS)
C
C     CHECK FOR SQUARE MATRIX
          IF (MROWS .NE. NCOLS)  THEN
              TYPE *, MATNAM, 'NOT A SQUARE MATRIX. NO REDUCTION DONE.'
              RETURN
              ENDIF
C
C     REDUCE TO  (1 X NCOL)  ROW VECTOR USING DIAGONAL VALUES
          DO 100 N=1,NCOLS
              MATRIX(N,1) = MATRIX(N,N)
  100         CONTINUE
          MROWS = 1
          RETURN
      END
```

```
            SUBROUTINE UNPACK(MATRIX,MROWS,NCOLS,CONTID,LUN,ALL,EOF)
            CHARACTER*72 RECORD
            CHARACTER*16 FLDID(2:5), CARDID*8, CONTID*3
            EQUIVALENCE ( RECORD(1:1), CARDID(1:1) )
            EQUIVALENCE ( RECORD(9:9), FLDID(2)(1:1) )
            CHARACTER*6 FMT(6), REAL, INTGR, BLANK
            REAL MATRIX(MROWS,NCOLS)
            LOGICAL ALL, EOF
            DIMENSION VALUE(4), IVALUE(4)
            EQUIVALENCE (VALUE(1), IVALUE(1))
            DATA FMT(1)/'(8X,'/  ,  FMT(6)/'8X)'/
            DATA REAL/'E16.8,'/  ,  INTGR/'I16,'/  ,  BLANK/'A16,'/
  C
  C         ZERO THE MATRIX
                DO 100 I=1,MROWS
                DO 100 J=1,NCOLS
                    MATRIX(I,J) = 0.0
   100              CONTINUE
  C
  C         PROCESS THE DMI CARDS FOR THIS MATRIX
                DO 400 K=1,1000000
                    READ(LUN,1000,END=425) CARDID,FLDID
                    IF (CARDID .EQ. 'DMI     ') THEN
  C                     THIS IS HEADER CARD OF NEXT MATRIX; THIS MATRIX DONE.
                        IF (ALL)  BACKSPACE LUN
                        GOTO 450
                    ELSE IF (CARDID .EQ. 'DMI*    ') THEN
  C                     THIS IS A COLUMN IDENTIFIER CARD
                        READ(RECORD,2000) JCOL,IROW,VALUE(1)
                        MATRIX(IROW,JCOL) = VALUE(1)
                        IROW = IROW + 1
                    ELSE IF (CARDID(1:3) .EQ. CONTID)  THEN
  C                     THIS IS A ROW/VALUE CARD
  C                     DETERMINE TYPE OF CONTENTS IN FIELDS 2:5
                            DO 200 L=2,5
                                IF (FLDID(L)(13:13) .EQ. 'E')  THEN
                                    FMT(L) = REAL
                                ELSE IF (FLDID(L)(16:16) .EQ. ' ')  THEN
                                    FMT(L) = BLANK
                                ELSE
                                    FMT(L) = INTGR
                                ENDIF
   200                          CONTINUE
  C                     READ CARD IN APPROPRIATE FORMAT
                            READ(RECORD,FMT,ERR=201) VALUE
  C                     PROCESS FIELDS 2:5
   201                      DO 300 L=2,5
                                IF (FMT(L) .EQ. REAL)  THEN
                                    MATRIX(IROW,JCOL) = VALUE(L-1)
                                    IROW = IROW + 1
                                ELSE IF (FMT(L) .EQ. INTGR)  THEN
                                    IROW = IVALUE(L-1)
                                ELSE IF (FMT(L) .EQ. BLANK)  THEN
                                    CONTINUE
                                ENDIF
   300                          CONTINUE
                    ELSE
  C                     THIS IS A NON-DMI CARD; THIS MATRIX DONE
                        READ(RECORD,1000)CARDID,FLDID
                        GOTO 450
                    ENDIF
```

```
400          CONTINUE
C     END OF FILE
425       EOF = .TRUE.
C     END OF MATRIX
450       RETURN
C
1000  FORMAT(A8,4A16)
2000  FORMAT(24X,2I16,E16.8)
      END
```

```
      SUBROUTINE WRTMTX(MATNAM,MATRIX,MROWS,NCOLS)
      CHARACTER MATNAM*8, NAME*12
      REAL MATRIX(MROWS,NCOLS)
C
      NAME = MATNAM//'.MTX'
      OPEN(UNIT=50,NAME=NAME,TYPE='NEW',CARRIAGECONTROL='LIST')
C     WRITE HEADER RECORD TO ASCII FILE
          WRITE(50,1000) MATNAM, MROWS, NCOLS
C     WRITE MATRIX TO ASCII FILE ( 1 VALUE PER RECORD )
          WRITE(50,2000) MATRIX
      CLOSE (UNIT=50)
      TYPE *, 'MATRIX FILE WRITTEN WITH MROWS, NCOLS:'
      TYPE *, NAME, MROWS, NCOLS
      RETURN
C
1000  FORMAT(A8,2I8)
2000  FORMAT(1PE16.8)
      END
```

GRDPTLST

The freedoms output from processor UNPACKDMI are input to processor GRDPTLST to put them into final format for STATCORR. GRDPTLST operates interactively and prompts the user for the input file name and the output file name.

```
  CCCC   TTTTT    GCCC   BBBB
 C         T     C       B   B
 C         T     C       B   B
 C         T     C       BBBB
 C         T     C CCC   B   B
 C         T     C   C   B   B
  CCCC     T      CCC    BBBB

  CCCCCCCC   RRRRRRRR    DDDDDDDD    PPPPPPPP    TTTTTTTTTT  LL            SSSSSSSS  TTTTTTTTTT
  CCCCCCCC   RRRRRRRR    DDDDDDDD    PPPPPPPP    TTTTTTTTTT  LL            SSSSSSSS  TTTTTTTTTT
CC           RR      RR  DD      DD  PP      PP      TT      LL          SS              TT
CC           RR      RR  DD      DD  PP      PP      TT      LL          SS              TT
CC           RR      RR  DD      DD  PP      PP      TT      LL          SS              TT
CC           RR      RR  DD      DD  PP      PP      TT      LL          SS              TT
CC           RRRRRRRR    DD      DD  PPPPPPPP        TT      LL            SSSSSS        TT
CC           RRRRRRRR    DD      DD  PPPPPPPP        TT      LL            SSSSSS        TT
CC  CCCCCC   RR  RR      DD      DD  PP              TT      LL                  SS      TT
CC  CCCCCC   RR  RR      DD      DD  PP              TT      LL                  SS      TT
CC      CC   RR    RR    DD      DD  PP              TT      LL                  SS      TT
CC      CC   RR    RR    DD      DD  PP              TT      LL                  SS      TT      ....
  CCCCCC     RR      RR  DDDDDDDD    PP              TT      LLLLLLLLLL  SSSSSSSS        TT      ....
  CCCCCC     RR      RR  DDDDDDDD    PP              TT      LLLLLLLLLL  SSSSSSSS        TT      ....
                                                                                                 ....

FFFFFFFFFF    000000    RRRRRRRR        ;;;;      11
FFFFFFFFFF    000000    RRRRRRRR        ;;;;      11
FF          00      00  RR      RR      ;;;;    1111
FF          00      00  RR      RR      ;;;;    1111
FF          00      00  RR      RR                11
FF          00      00  RR      RR                11
FFFFFFFF    00      00  RRRRRRRR        ;;;;      11
FFFFFFFF    00      00  RRRRRRRR        ;;;;      11
FF          00      00  RR  RR          ;;;;      11
FF          00      00  RR  RR          ;;;;      11
FF          00      00  RR    RR          ;;      11
FF          00      00  RR    RR          ;;      11
FF            000000    RR      RR      ;;      111111
FF            000000    RR      RR      ;;      111111

  CCCC   TTTTT    GCCC   BBBB
 C         T     C       B   B
 C         T     C       B   B
 C         T     C       BBBB
 C         T     C CCC   B   B
 C         T     C   C   B   B
  CCCC     T      CCC    BBBB
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
         PROGRAM GRDPTLST
C
C        THIS PROGRAM WILL READ THE TESET (DOF ID VS. GRID POINT ID)
C        MATRIX FILE OBTAINED FROM THE NASTRAN PUNCH FILE AND GENERATE
C        A GRID POINT ID LIST FILE, USING THE NON-ZERO ENTRIES OF THE
C        TESET MATRIX.
C
         CHARACTER*35 MTXFIL, GPFIL, MTXNAME*8
C
C        GET TESET MATRIX FILENAME
 10          TYPE 1000, 'ENTER GRID POINT ID MATRIX FILENAME:'
             ACCEPT 2000, MTXFIL
             OPEN(UNIT=1,NAME=MTXFIL,TYPE='OLD',READONLY,ERR=10)
             READ(1,3000) MTXNAME, MROWS, NCOLS
             IF (NCOLS .NE. 1)  STOP 'ERROR: SHOULD BE ONLY ONE COLUMN'
C        GET GRID POINT LIST FILENAME
             TYPE 1000, 'ENTER GRID POINT ID OUTPUT LIST FILENAME:'
             ACCEPT 2000, GPFIL
             OPEN(UNIT=2,NAME=GPFIL,TYPE='NEW',CARRIAGECONTROL='LIST')
C        GENERATE LIST FILE FROM NON-ZERO MATRIX ENTRIES
             NIDS = 0
             DO 100 MDOF=1,MROWS
                 READ(1,5000) GRDPT
                 IF (GRDPT .NE. 0.0)  THEN
                     IGRDPT = INT(GRDPT)
                     COMP = GRDPT - FLOAT(IGRDPT)
                     ICOMP = NINT(10.0*COMP)
                     IF (ICOMP .GT. 6)  THEN
                         IGRDPT = IGRDPT + 1
                         ICOMP = 0
                         END IF
                     WRITE(2,6000) IGRDPT, ICOMP, MDOF
                     NIDS = NIDS + 1
                     END IF
 100             CONTINUE
         CLOSE(UNIT=1)
         CLOSE(UNIT=2)
         TYPE *, NIDS, 'ENTRIES WRITTEN TO GRID POINT LIST'
         STOP
C
 1000    FORMAT(1X,A44,1X,$)
 2000    FORMAT(A35)
 3000    FORMAT(A8,2I8)
 4000    FORMAT(1X,A8,' IS A ',I8,' ROW BY ',I8,' COLUMN MATRIX')
 5000    FORMAT(E16.8)
 6000    FORMAT(I8,'-',I1,I12)
         END
```

FUTURE REQUIREMENTS

1. If the practice of comparing analysis with test, as outlined here, for certifying a design should become commonplace, one could imagine that the demand for additional information would increase. Instead of depending on a comparative analysis to indicate that differences do exist, and roughly where, engineers increasingly might require comparative analysis to predict more precisely where differences arise and their causes. An extension of the existing method in this direction would be to apply a correlation function using a correlation displacement variable. Consider the expression

$$\emptyset(P,r,s) = \int_{V_p} T(x_p,y_p,z) \times A[(x_p - r, y_p - s)z]\ dV.$$

$\emptyset$ measures the best correlation of a test function T at a particular point $P(x_p,y_p,z_p)$ with an analytical function A in the neighborhood of P:$[V_p]$ as the analytical function is evaluated at displacement distances (r and s) away from P in x and y. The motivation for considering such a quantity is to explore a point at which earlier results of a correlation coefficient showed consistent differences over a number of modes at one location of the structure. Since tests are limited as to the intensity of data obtainable in a small region, the test coordinate can be held constant in one or two dimensions while the analysis mesh is intensified in the region to provide enough definition to step through values of r and s to find what location in the vicinity of P shows greater correlation of A with T at P. Such a spatial correlation function might prove helpful in exploring a design defect or an analytical modeling error of a structure.

2. Another extension is to equip the correlation to make comparisons on the basis of mode shapes defined in the complex domain. Correlation with a damping displacement variable could prove useful in refining the damping away from average modal considerations to spatial distributions of damping.

3. Currently, the analysis presumes that any particular point on the test article, instrumentation can detect only one component direction of behavior. If multi-axis behavior at a test point can be provided, the component at an analytical point can be implemented by adding one more digit after the decimal point of the real number representing a grid point. The processor program can discriminate to the nearest integer by multiplying by 10, and then rounding off. Thus, the real value in TESET and DOFLIST can reflect correspondence in keeping with increased experimental detail.

4. Processing of analytical data in preparation for STATCORR is handled in discrete operations. If a number of different analytical runs are being processed at the same time for later assembly as STATCORR input, their separate results might get uncoordinated, so that a symmetric mode shape, adjusted for sign at certain image points might be identified with a wrong frequency or modal stiffness. With further effort, the several operations could be combined with automated bookkeeping to eliminate the risk of getting results scrambled.

5. Human error can be further reduced if the entire analysis were made interactive with computer prompts. Then when STATDMAP is accessed from USERLIB, the whole sequence of orderly information train could be set in motion to keep all relations coordinated.

6. STATCORR can be readily extended to comparisons of static loadings. First of all, the mean, x′, has entirely different meaning in statics than in modal analysis. x′ for statics cannot be prescribed as the equilibrium position. The non-zero mean must be reintroduced into the stochastic quantities. Scaling was meaningful for eigenvalues because the method of normalizing could be arbitrary. The relative deviation of equation (7) should use actual unscaled response data for identifying magnitudes of difference.

7. The implementation of the option for comparing symmetrically based analysis against samples over the entire test article was restricted to the use of basic coordinates only. A logical extension is to allow any number of rectangular coordinates for defining direct and image points. Enough extra work would be involved to implement more than one coordinate system, that it may be tolerable to limit the first extension of rectangular coordinates whose planes are parallel to the planes of symmetry. In a second extension, all types of coordinates can be logically admitted.

USER MANUAL SUMMARY

This document is intended to serve as a guide for comparing analytical modes with test when using NASTRAN, STATCORR and its associated processors, in conjunction with the test data.

First, select one of three bases of comparison: (1) whole vs. whole in which both the analytical modeling and the test instrumentation involves the whole structure; (2) symmetric vs. symmetric in which the analytical modeling and the test instrumentation involves only a basic segment of reflective symmetry; and (3) symmetric vs. whole in which the analysis is based upon modeling of only a basic segment of reflective symmetry and the test is based upon instrumenting the whole structure.

ORIGINAL PAGE IS
OF POOR QUALITY

USER MANUAL SUMMARY

STEP/ROUTE	WHOLE/WHOLE	SYM/SYM	SYM/WHOLE
LIASION	Establish (a) coordinate system (b) model locations of instrumented points (c) component directions at each location, (d) identification number of each point.		
	Analytical model should contain a grid point corresponding to each instrumented point.	Analytical model should contain a grid point corresponding to each instrumented point within the basic segment. Test should have additional instrumentation outside the basic segment to sense phase relationship with respect reflected segment to ascertain the type of modal symmetry across planes of symmetry.	Analytical model should contain a grid point corresponding to each direct point, image point, and cloned point which is instrumented on the test article.
	Determine the precise boundary condition at every test support point and include that boundary behavior at corresponding points in the analytical model. Special care should be taken to ensure that boundary elasticity be measured at an apparent firm boundary point.		
		Agree on which plane will be called the 1st plane of symmetry; which the second plane of symmetry; and which the 3rd plane of symmetry. Agree on how each plane is to be defined in terms of the basic coordinate system.	Agree on which plane will be called the 1st plane of symmetry; which the second plane of symmetry; and which the 3rd plane of symmetry. Agree on how each plane is to be defined in terms of the basic coordinate system.
			Check to see tht instrumented points when reflected back to the basic segment will coincide with a point of the analytical model. When instrumented points in different octants all reflect back to the same point in the model, agreement on its multiplicity will establish the number of clones needed. Each clone bears the number of a particular instrumented point.
STRUCTURAL ANALYSIS	Precede Eigenvalue Run with CHKPNT YES and DIAG 21,22 commands, then take an exit after GP4 has executed using the ALTER packet: ALTER 53, EXIT $, ENDALTER. Use listings of DIAG 21 and 22 to construct partitioning vector TESET.		
			Use DIAG 21 and 22 data to construct the matrix DOFLIST.
	Merge CHKPNT dictionary after ID card for a RSTRT run. Remove ALTER for exiting after GP4. Remove DIAG 21 and 22. Add DIAG 14. Reconsider use of CHKPNT; generally advised if model is large. If mass of model checks O.K., disable GPWG with: ALTER 40, PARAM //*ADD*/V,Y,GRDPNT/-1/0 $. Add DMI cards to BULK DATA for TESET vector. Follow instructions in packet ***P*** for the preparation of TESET data. For an explanation of these instructions see page ???? of the STATCORR report. Fetch DMAP ALTER file DRA0:[NASCAT.UTILITY.STATCORR.DEMO]STATDMAP.LIS. Edit file to suit the route before merging.		
	Delete packets ***S*** and ***W***	Delete packet ***W***.	If coord system is rectangular, enter data at locations: *A*A*A*A, *B*B*B*B, & *C*C*C*C. Remove phrases pertaining to cylindrical and spherical coordinate systems. If coord system is cylindrical, enter data at locations *T*T*T*T. Remove phrase pertaining to rectangular and spherical coord systems. Enter

ORIGINAL PAGE IS
OF POOR QUALITY

values of IN1 for parameters SYMϕPLAN that indicate the kind of symmetry being imposed by the SPC case in Case Control. Remove $S$$ from in front of DMAP statements S1 thru S8. Remove SS from ahead of ALTER 125. If coord system is spherical, remove phrases pertaining to rectangular and cylindrical coord systems.

Merge ALTER into EXECC. Check to see that ENDALTER is only at the very end of ALTER packet.

Add DMI cards to Bulk Data for DOFLIST matrix. Follow instructions in packet **W** for the preparation of DOFLIST data. For an explanation of these instructions see page ???? of the STATCORR report.
Add DMI cards to Bulk Data for IDCORD. Follow instructions in packet **W** for preparation of IDCORD data.
Remove $W1$ from ahead of DMAP statement W1.
Remove WW from ahead of ALTER 128.
Merge ALTER ito EXECC. Check to see that ENDALTER is only at the very end of ALTER packet.

If a printout of only new Bulk Data is wanted during a RSTRT run, set ECHO = UNSORT in Case Control. Put a NASTRAN card ahead of the EXECC with FILES = (NPTP,OPTP) if checkpoint and restart files are to be directed to disk instead of tape. NASTRAN data is ready for eigenvalue run. Prepare system job control language commands for CHKPNT, RSTRT, & PUNCH. $ASSIGN(old problem tape file) FOR008, $ASSIGN(new problem tape file) FOR009, $ASSIGN(punch file) FOR077. Make as many eigenvalue runs as is necessary to span the frequency range.

Change IN1 values for parameters SYMϕPLAN to correspond to symmetry according to new SPC set. Make as many eigenvalue runs as is necessary to model every type of symmetry over the frequency range.

If the DMAP ALTER executed successfully, the following messages will be found in the job listing:
USER INFORMATION MESSAGE 4015 Table named LAMA punched onto ______ cards.
USER INFORMATION MESSAGE 4103 OUTPUT3 has punched matrix data block PHITE onto ______ DMI cards.
USER INFORMATION MESSAGE 4103 OUTPUT3 has punched matrix data block TESET onto ______ DMI cards.
USER INFORMATION MESSAGE 4103 OUTPUT3 has punched matrix data block DOFLIST onto ______ DMI cards.
USER INFORMATION MESSAGE 4103 OUTPUT3 has punched matrix data block IDCORD onto ______ DMI cards.

PROCESSOR MERGE Fetch the MERGE processor from the directory DRAO:[NASCAT.UTILITY.STATCORR.OLD]MERGE.EXE.

Input data LAMA, TESET, and PHITE for all cases.
Sorted on frequency.
Merged.
Duplicates removed.
Output.

Input SYMϕPLAN vs.PHITE for every symmetric case.
SYMϕPLAN is attached to each PHITE.
Sorted on frequency.
Merged.
Duplicates removed.
Output.

Input DOFLIST, IDCORD.
Output for whole.

Send kappa and mu.

STRUCTURAL TEST Check that sensors match in location and orientation. Method is the same regardless of existence of symmetry.
Excite the structure for steady state modal response.
Record frequencies, phase angles, modal displacement data.
Stop excitation and measure decay data for damping.
Process.. Scale to forcing input. Check for best response at a frequency over various forcings. Cull best and assemble.
Check phase angle for symmetry.
Assign SYMϕPLAN and attach to MS data. Record zeta.
Transcribe modal test data into the required format for STATCORR using in-house Modal Survey Program. Set up one data file per mode containing frequency, damping, symmetry planes, GP id's, and modal vector.

CORRELATION ANALYSIS STATCORR	Fetch the Statistical Correlation Program from the utility directory [NASCAT.UTILITY.STATCORR]STATCORR.EXE. Input--Analytical files are: LAMA.TBL, PHITE.MTX, GRDPTF.LIS. Experimental files are: EXPMOD#.INP. One for each mode. Enter NONE after filename of last experimental mode. Parameters: 1. Is analytical vs. experimental symmetry to be considered? (Y or N): 2. Is a print wanted for analytical mode-shape vectors? (Y or N): 3. Is a print wanted for experimental mode=-shape vectors? (Y or N): 4. Relative deviations greater than a threshold value will be printed. The default is 0.050. Enter desired threshold: ",". For default press RETURN key.
INTERPRET STATCORR OUTPUT	Per program READCORR.

1. Report No.
NASA TM-86081

2. Government Accession No.

3. Recipient's Catalog No.

4. Title and Subtitle
Statistical Correlation Analysis for Comparing Vibration Data From Test and Analysis

5. Report Date
FEBRUARY 1986

6. Performing Organization Code
753.2

7. Author(s) T. G. Butler, R. F. Strang, L. R. Purves, and D. J. Hershfeld

8. Performing Organization Report No.
84B0561

9. Performing Organization Name and Address
Goddard Space Flight Center
Greenbelt, MD 20771

10. Work Unit No.

11. Contract or Grant No.

12. Sponsoring Agency Name and Address
National Aeronautics and Space Administration
Washington, DC 20546

13. Type of Report and Period Covered
Technical Memorandum

14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

A theory is developed for comparing vibration modes obtained by NASTRAN analysis with those obtained experimentally. Both are treated as random data sources. Because many more analytical modes can be obtained than experimental modes, the analytical set are treated as expansion functions for putting both sources in comparative form. Three dimensional symmetry was developed for three general cases–non-symmetric whole model compared with a non-symmetric whole structural test, symmetric analytical portion compared with a symmetric experimental portion, an analytical symmetric portion with a whole experimental test. Theory was coded and a statistical correlation program was installed as a utility. Theory was established with small classical structures.

17. Key Words (Selected by Author(s))
Vibration, Comparison, Correlation, Frequency, Eigenvectors, Vibration Analysis, Eigenvalues, Vibration Test, Modes

18. Distribution Statement
Unclassified - Unlimited

Subject Category 39

19. Security Classif. (of this report)
UNCLASSIFIED

20. Security Classif. (of this page)
UNCLASSIFIED

21. No. of Pages
88

22. Price
A05

For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA-Langley, 1986