Technical Memorandum No. 33-220

Summary of the Functions and Capabilities of the Structural Analysis System Computer Program

Theodore E. Lang

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

June 15, 1965

Technical Memorandum No. 33-220

Summary of the Functions and Capabilities of the Structural Analysis System Computer Program

Theodore E. Lang

M. E. Alper

M. E. Alper, *Manager*
Applied Mechanics Section

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

June 15, 1965

[redacted]
Jet Propulsion Laboratory
California Institute of Technology

Prepared Under Contract No. NAS 7-100
National Aeronautics & Space Administration

CONTENTS

FIGURES

FIGURES (Cont'd)

FOREWORD

The Structural Analysis System (SAS) Computer Program described in this report was developed by Philco Corporation, Western Development Laboratories (WDL), Palo Alto, California, under contract to the Jet Propulsion Laboratory, Pasadena, California (Contract No. 950321).

The program was developed to solve complex structural problems by the analytic techniques of the direct stiffness method. The development work by WDL was supervised by Dr. P. R. Cobb with Dr. R. J. Melosh as project engineer. The support effort by JPL was under the supervision of R. R. McDonald and Dr. M. E. Alper with T. E. Lang as project engineer.

ABSTRACT

33141

The functions and operations of a large capacity Structural Analysis System Computer Program developed to analyze frame and shell-type structures are described. Included is a summary of the capabilities of the program and a discussion of certain of the problems encountered in development of the program. The participation of engineering personnel in the setup and running of a typical shell problem is outlined.

Author

I. INTRODUCTION

The purpose of this report is to describe the functions and capabilities of the Structural Analysis System (SAS) Computer Program that was developed by Philco Corporation, Western Development Laboratories (WDL), under contract to the Jet Propulsion Laboratory. The original task outlined in the contract was to develop a structural analysis program that would have the capability of predicting the static and dynamic behavior of complex shell-type structures. By complex, here, is meant shell structures that may be material-anisotropic, laminated as in sandwich construction, locally stiffened, and/or geometrically irregular. Hence, included are shell characteristics that are not amenable to closed-form solutions or finite difference or numerical integration techniques. Therefore, an approximate method of analysis was selected which has built-in solution convergence to exact answers as the structural idealization is refined. This method of analysis uses the "finite-element" concept and an energy approach. It entails representing a structure by an array of substructures (finite elements) that are selected to be consistent with the geometric character of the original structure. For example, for singly and doubly curved shells, a flat triangular plate element is a logical element to use in the idealization.

Using the finite element concept there are two approaches to setting up a problem, namely, the force or flexibility method and the displacement or stiffness method (Ref. 1 and 2). Based on its simplicity, efficiency, and generality, the stiffness method was selected and a computer program was developed to solve shell problems using the IBM 7094 computer. Because of the requirement for having the capability to analyze stiffened shells and layered structures, several elements other than the triangular shell element were added to the program element library. Thus the final program is applicable in analyzing a wide spectrum of structural types including truss and frame structures, shells of revolution, and combinations of these.

In establishing a program development plan compatible with contractual requirements, it was decided to divide

the effort into sequential tasks (contractual phases) so that the initially developed programs could be assessed before extending the work scope. This approach was practical because the finite element representation had not been developed so it was assured that an adequate shell representation could be formulated. In this report, the program developed under Phases I, Ia, and II of the contract is described. The intention here is to present an overall picture of the program and its capabilities without delving into details on mathematical algorithms, computer operations, input-output format, etc. For information of this type, two documents have been prepared by WDL (Ref. 3 and 4) which are updated as changes and extensions to the program are made. One additional document (Ref. 5) that complements the WDL reports and this report summarizes the test problems and associated input-output data that were used in final checkout of the SAS program.

II. DESCRIPTION OF THE STRUCTURAL ANALYSIS SYSTEM PROGRAM

The computer-oriented Structural Analysis System (SAS) is based upon the analysis techniques of the direct stiffness method. In applying this method a given structure is idealized by an array of subelements that approximate the local structure they represent. For a shell structure, the flat triangular shell element having membrane and bending stiffness is well suited for the idealization. Hence, in applying this method, any shell is approximated by a polyhedron of triangles in which displacements and forces are specified or determined at the apexes or nodes of the triangles (see Fig. 1).

If a triangle is arbitrarily oriented with respect to a set of overall coordinate axes, then six independent variables are required to describe the deflection of a node (three displacements and three rotations). These deflections or their complement in forces and moments are either unknown or prescribed at each node (see Fig. 2), and if unknown are calculated in solving the problem. The number of nodes is dependent on the triangle grid array selected to represent the structure. For a fine grid array,

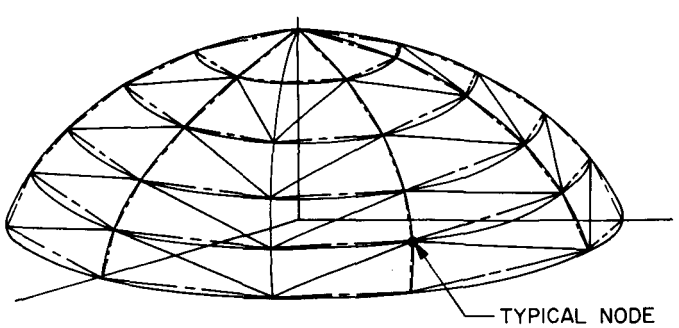


Fig. 1. Triangular grid array of a spherical shell

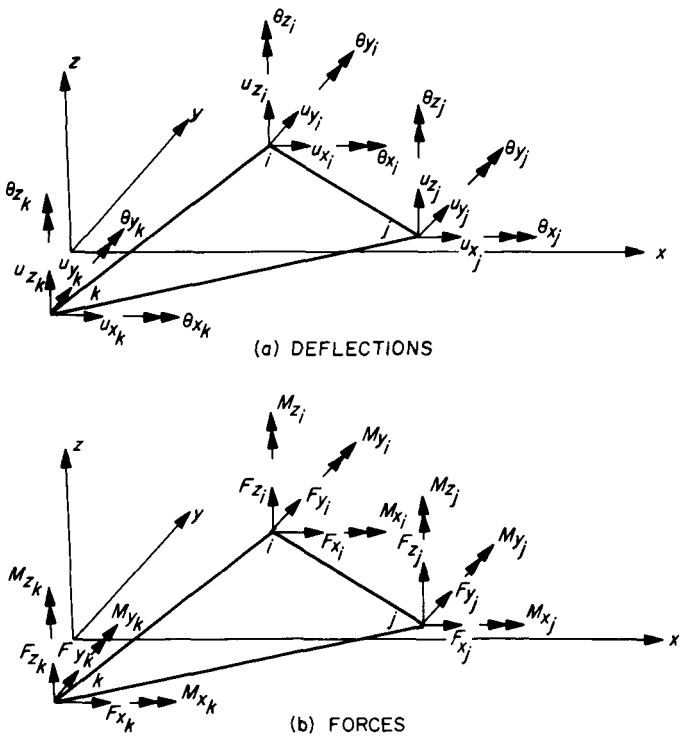


Fig. 2. Node deflections and forces

which is generally required in shell representations, a large number of nodes results. An upper limit on the number of unknowns of the problem is six times the number of nodes. Hence, a typical shell problem inherently involves solving a large number of simultaneous equations which easily can exceed core capacity of even the largest modern digital computers. Therefore, in the SAS it was necessary to supplement in-core storage with magnetic tape and/or disk storage. Implementing this data-storage technique requires data to be input and output from the core of the computer in convenient block sizes as the calculations are performed, in order not to exceed core capacity. Thus, certain of the SAS subprograms use the tape-core coupling technique in which data is input in small blocks from tape to core. As the calculations are completed, the output is stored on tape in like manner or retained in core if it fits and is part of the next calculation. With this arrangement there is virtually no limit on the amount of data storage capacity; however, the core-tape coupling involves significant data handling and transfer times.

For some manipulative operations, the amount of data normally is small so the operations can be performed in-core. This, of course, restricts the applicability of the algorithm; however, in many instances there are alternate schemes of combating this capacity problem, or the problem is not acute. In the SAS, for example, the subprogram used to compute characteristic roots in an eigenvalue problem is limited to in-core operation. However, even within this constraint, 130 roots and vectors can be computed simultaneously, which for most problems, is adequate.

To completely automate a structural analysis, the computer program must be set up to perform two basic functions; namely, generate the stiffness and stress matrices and loading vectors, and then perform the manipulations necessary to solve for the unknowns.

The first function, the generation phase, is performed by the computer only if geometric and materials data are input to the program. From these input data the SAS program generates the following:

1. Element stiffness and stress matrices,
2. Fixed node forces due to temperature profiles and gradients,
3. Equivalent node forces due to uniform pressure loads,
4. Gravity loading vectors from imposed acceleration states,
5. System mass matrices for uniformly distributed mass, and the
6. Small deflection buckling stiffness matrices.

This generation phase is represented by two subprograms or links in the program library. These links have code names BILD and BUKL and involve several generation functions which are outlined in Fig. 3 and 4.

As noted in Fig. 3, the structural elements in the library of the generation subprograms include the beam, bar,

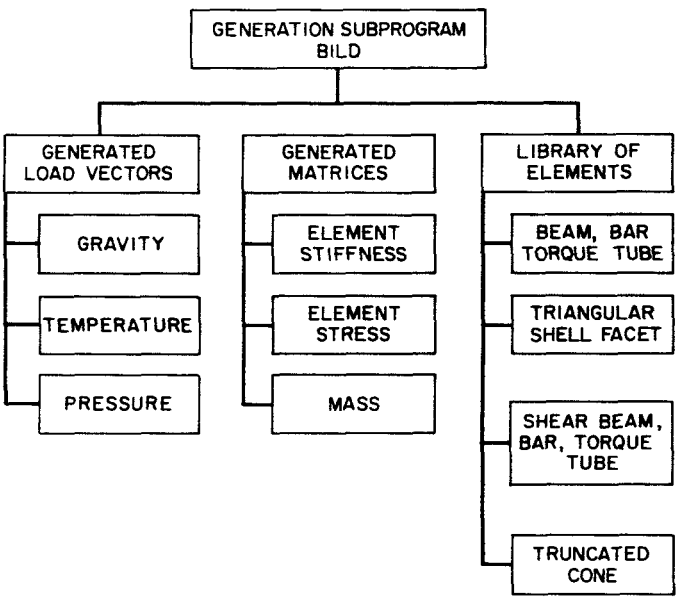


Fig. 3. Generated data and library of elements in the BILD subprogram of the SAS

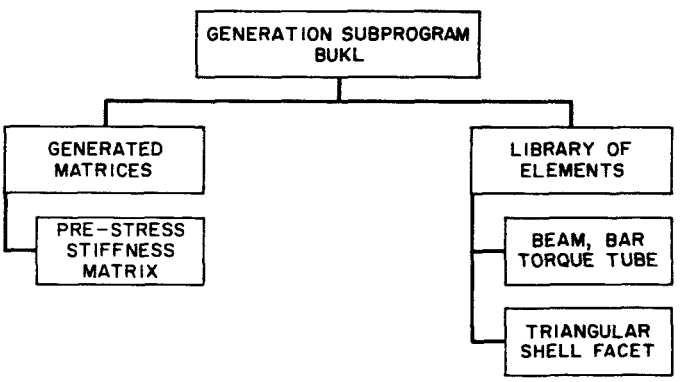
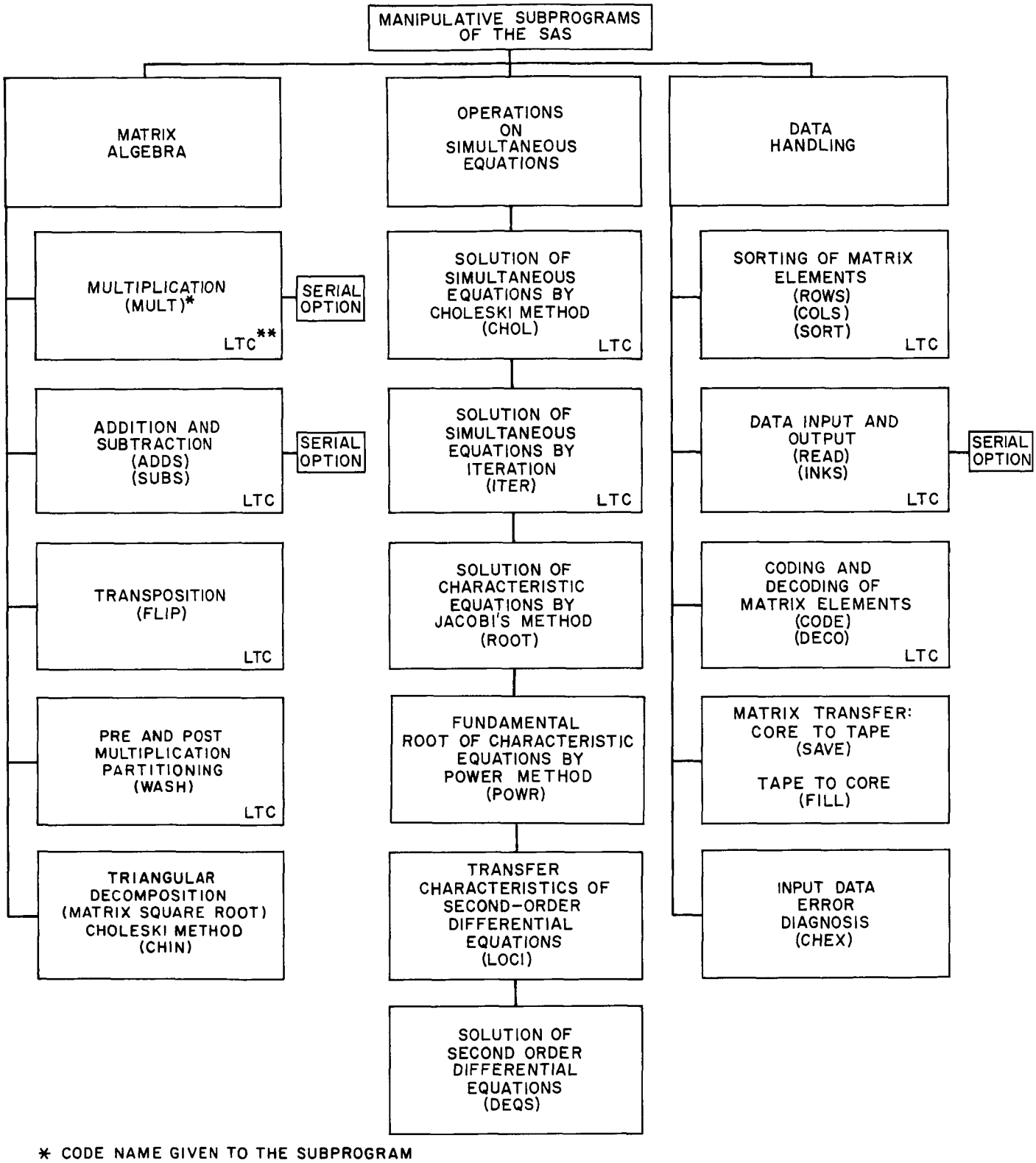


Fig. 4. Generated data and library of elements in the BUKL subprogram of the SAS



* CODE NAME GIVEN TO THE SUBPROGRAM

** LARGER-THAN-CORE CALCULATION CAPABILITY (LTC)

Fig. 5. Manipulative subprograms of the SAS

torque tube, triangular shell facet, shear beam, and truncated cone; these may be used in any combination in a structural idealization.

The second function of the SAS is to manipulate mathematically the generated data to determine the unknowns of the problem. This manipulative phase is currently made up of 17 subprograms. Five of the subprograms perform standard matrix algebra; namely, multiplication, addition and subtraction, transposition, triangular decomposition, and row-column partitioning. Five others perform functions on simultaneous equations. Included here are subprograms for:

1. Solving simultaneous algebraic equations,
2. Finding roots and vectors of the dynamic matrix equation,
3. Calculating root loci for a set of simultaneous equations under a variable change,
4. Solving the second order matrix differential equation with discretized or certain functional forcing functions, and
5. Calculating the fundamental eigenvalue and associated vector of a matrix (used in the buckling calculation).

The seven remaining subprograms of the manipulative set are special-purpose programs for input and output of data, and for carrying out manipulations particular to the data format used in the SAS. The 17 manipulative subprograms are listed in Fig. 5, together with information on subprogram identification and data-size restrictions. The subprograms having "serial option" blocks are set up to perform multiple operations in the function represented by the parent block. For example, serial multiplication is a subprogram option that allows sequential multiplication of matrices without writing separate instructions for each step.

Sequencing the computational steps to carry out a problem solution is accomplished by writing a set of instructions. This set of instructions is called the "pseudo instruction program." Conceptually, pseudo instructions are quite similar to FORTRAN instructions, the only difference being that a pseudo instruction calls for a matrix operation rather than a discrete operation. One pseudo instruction is required for each matrix operation that is to be performed (excepting the "serial" options). For example, to multiply two matrices, one pseudo instruction is needed which contains information on where the two matrices are located (either in core or on prescribed tapes), the operation to be performed (multiplication in this instance) and where the resultant (product) matrix is to be stored (in core or on tape). The actual pseudo instruction for this operation would be of the form of Fig. 6.

Interpretation of this instruction is: read into core matrix KAR001, which is on Tape 9, Location 1, and multiply it by matrix MRR001, which is in core. Designate the product matrix PRR001 and store it on tape 10, location 3.

A Master Intelligence System (MIS) in the core of the computer is the monitor system that controls the execution of the pseudo instructions. For the multiplication operation outlined above the procedure is the following:

1. Upon completion of the pseudo instruction prior to multiplication, control is given to the MIS.
2. MIS reads the next pseudo instruction (multiplication instruction) from the pseudo program tape and determines the tapes that will be needed and the function to be performed.
3. MIS positions all of the tapes involved (tape 9 at location 1, tape 10 at location 3), so that the next location on the tape is either the start of the input data for the operation or space for the output.

			MBC001			ROWS	11002	MBR001	
		11002	MBR001	11003	MBR002	ADDS		MRR001	
		09001	KAR001		MRR001	MULT	10003	DRR001	
		10003	PRR001	09002	FXC001	CHϕL	11001	DIC001	
		11001	DIC001			INKS			
						HALT			

Fig. 6. Example pseudo instruction

4. MIS then locates on the SAS library tape the subprogram that performs the operation (MULT), and brings this subprogram into core.
5. Control is then shifted to this subprogram (MULT), which calls for the input matrices (KAR001, MRR001), performs the calculation (multiplication), and locates the resultant matrix (PRR001) in core or on tape as specified.
6. Control is then returned to MIS and the process is repeated with the next pseudo instruction.

Generally, a pseudo instruction program for structural analysis varies little from problem to problem, so that once an efficient program is written it becomes a standard part of the input data for any structural problem.

Noting the example pseudo instruction given previously, it can be observed that specification of data storage tapes is an integral part of each pseudo instruction. This option requires greater user knowledge in setting up a pseudo instruction program than if tape assignments were specified internal to the program; however, the system has greater applicability through use of this scheme. By user discretion, certain key data can be stored on tapes that will be saved after completion of computations on the computer. These data are then available for subsequent runs, thus avoiding complete regeneration of data. For example, in a structural problem the summed stiffness matrix should be saved if any one of the following conditions is suspected: A nonrecoverable error might occur in calculations that follow the generation phase, and computations are terminated by the computer; a few elements of the structure are likely to be redefined subsequent to current calculations; or additional loading states are likely to be defined subsequent to current calculations. Another advantage in assigning tapes is that the program is made amenable for operation on different computer systems and is operable when one or more tape units are removed from the system for repair.

In assigning data storage tapes in a pseudo instruction program, it is efficient to store data on a number of tapes rather than on one or two. The reason is that a search of a tape for particular data is reduced if the number of data groups on the tape is small. In the JPL computer system, seven tapes are available for data storage during program execution (Fig. 7). For the moderately long problems that have been run to check out the SAS program (20 to 50 pseudo instructions), three to five tapes have been used in which one or two of these were saved for recovery purposes.

For a given computer complex, if an adequate number of tapes are available, then the SAS program library (17 subprograms) should be divided onto two or more tapes to reduce search times. For example, the JPL computer system has 18 tape consoles, so that two tapes are available to store the SAS library (Fig. 7, tapes A4 and B5). At program initiation, the Master Intelligence System is read from one of the SAS library tapes into core, at which time it takes control of the computer to perform functions already described. The disk file shown in Fig. 7 is used to store the computer systems library, which, however, is also currently stored on tape A3 as a backup. Eventually the entire SAS program will be adapted to disk storage, since by this mode of data storage, search times are reduced greatly over those for magnetic tape.

Because of the sequential nature of the calculations characterized by the pseudo instruction program, it is possible to restart calculations at any point in the pseudo program provided the data generated to this point have been stored on tape. This feature of the SAS is termed the "recovery feature" and is predicated upon the writer of the pseudo instruction program planning, in advance, recovery points based upon likely locations for errors in the calculations. To support this feature, instructions may be given to computer operators to save certain tapes and then to remount these tapes the next time the program is run.

Related to recovery is the recoverable error option. In the SAS, it is possible to set up a matrix calculation and provide alternate calculations if the matrices are actually larger than core. When too large a matrix is found a recoverable error occurs. The recoverable error option allows for shifting from the subprogram for in-core calculations to the subprogram for larger-than-core calculations without stopping the calculations and resubmitting the problem. This error option could have been automated in the program by action based upon the result of a test on matrix size; however, because of the sacrifice in core space and the infrequent occurrence of matrices of sizes bordering on core capacity, error recovery was made a user option.

The recoverable error option is also important when several problems are stacked for one run on the computer. By use of this option any error in an early calculation will not terminate calculations in subsequent problems.

Having given a descriptive summary of the functions and operation of the SAS program, we now present a summary of the capabilities of the system.

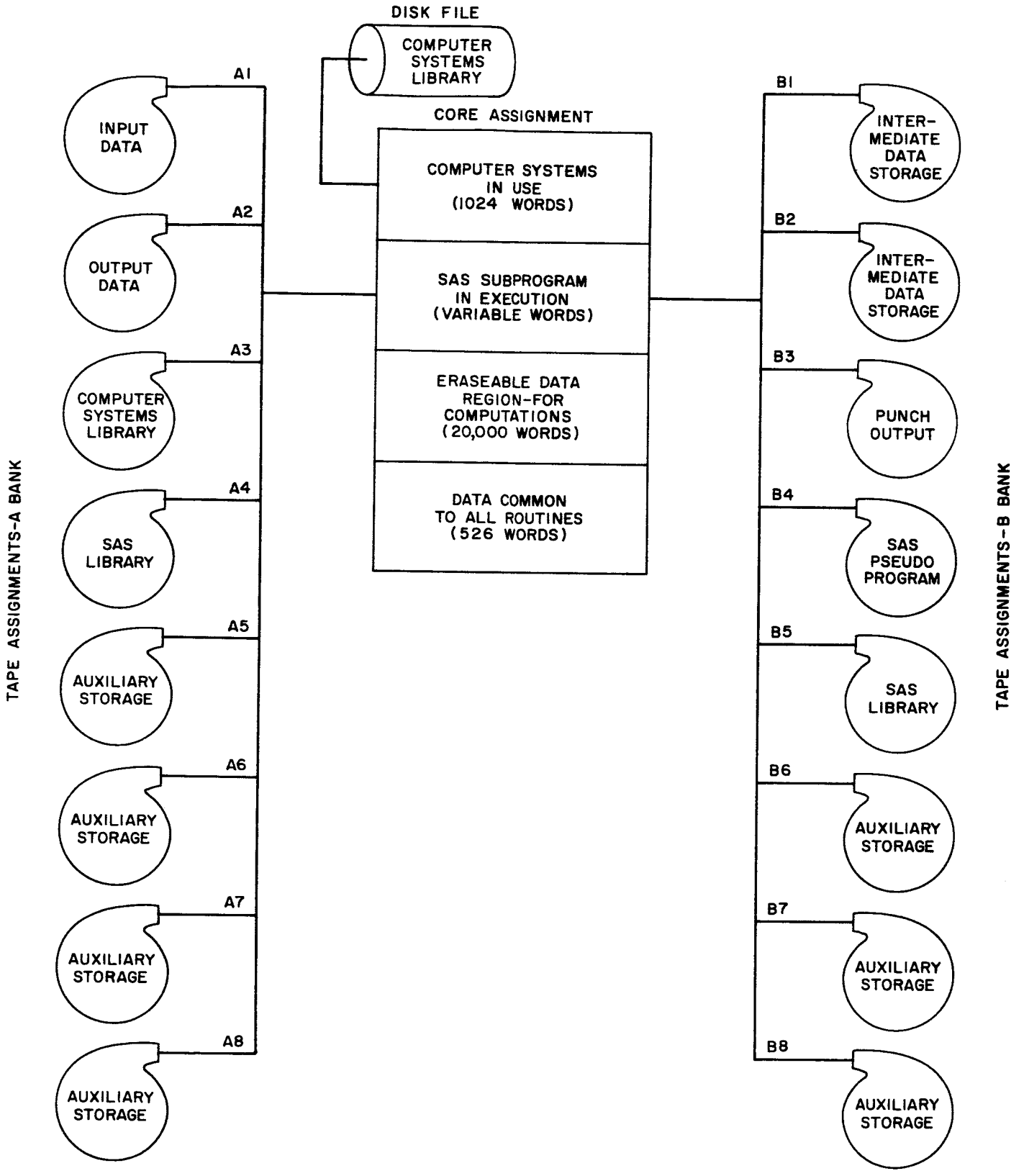


Fig. 7. Core, disk file, and tape assignments of SAS on the JPL computer (IBM 7094)

III. CAPABILITIES OF THE SAS PROGRAM

As indicated in Section II, the SAS program performs two basic operations, namely, generation of structural data and manipulation of data in matrix format. Since these two functions are mutually exclusive they may be considered separately in defining system capabilities. First we will consider the "generation phase" of the SAS.

The program was originally planned to analyze complex shell structures; therefore, emphasis was placed on development of a flat triangular shell element applicable for idealizing shells. However, a structure requiring idealization by triangular elements can also have other elemental members such as stiffeners or flanges, so that several additional elements have been added to the SAS library of elements. Thus, the BILD subprogram can generate structural data in matrix format for several structural types, retaining, in the process, consistent node identification so that the elemental matrices may be superimposed to provide a representation of the composite structure. The characteristics of each of the elements in the SAS library may be summarized as follows (Fig. 3): The beam-bar-torque tube element can have arbitrarily shaped cross sections, provided the shear center and/or twist center are coincident with the principal longitudinal geometric axis. This element representation optionally includes the effects of shear deflection and rotary inertia, and each element can be supported by any of several types of end constraints. The "shear-beam" element differs from the beam element by a loading state that includes a shear stress distribution along the length of the beam at one outer surface. This element representation incorporates a coordinate transformation so that the axis of bending is parallel to, but need not be coincident with, the principal geometric axis of the element. Thus, this element models the behavior of a nonsymmetric stiffening member.

The triangular-plate element can represent a structure having aeolotropic material properties (13 independent constants in the constitutive equations), varying elastic moduli with temperature (interpolation and extrapolation of a material properties table), and varying thickness and/or density properties. A restriction on this element representation when the material is nonisotropic is that the principal material axes must align with prescribed geometric axes unless a transformation of the constitutive equations is made prior to use in the SAS program. The triangular element can be used in representing laminated or layered multidimensional structures.

The conical element which is derived in Ref. 6 is used to analyze shells of revolution when the loading on the shell can be prescribed by a truncated Fourier Series. Orthotropic materials may be analyzed, provided the principal material axes align with the principal geometric axes of the shell. The effects of shear deflection and rotary inertia can be included in analyses with this element.

In generating the structural data by use of BILD, the stiffness and loading matrices are read onto tape in 127-word blocks (120 words of data and 7 words of identification) so that the core of the computer is not filled. Therefore, theoretically any number of individual element matrices can be generated; however, an upper limit has arbitrarily been set at 999 elements in a single pass since this is the limit of assignable matrix numbers.

Equivalent node forces are computed internal to the program for the elements in the SAS library when subjected to temperature gradients normal to the neutral axes, temperature distributions, uniform accelerations, and/or pressure loads. This system capability eliminates lengthy manual calculations to define node forces due to these types of loads. However, individual equivalent node forces may also be input to the system if the analyst desires to augment or change the structural loading.

The generation subprogram BUKL provides capability to generate a prestress matrix for certain of the elements in the library (see Fig. 4). After superpositioning, the resultant matrix is used in the computation of the fundamental buckling mode of the structure. The link POWR facilitates calculation of the first buckling load and mode when the variable band matrix fits in core.

It is important to recognize that program restrictions on *types* of structures that can be analyzed is due only to the types of elements contained in the program library. Provision has been made for adding elements to this library without serious revamping of the subroutines involved. For example, the input and output data formats are sufficiently general to accommodate a wide variation in possible element format requirements. Additional information on the current library elements and on the functions and capabilities of the subprograms BILD and BUKL may be found in Ref. 3.

The capabilities of the manipulative subprograms of the SAS are probably of more general interest than those of the generation phase because the manipulative function is not restricted to structurally oriented problems. Some indication of subprogram manipulative capabilities is given in Fig. 5 by the labeling of the subprograms as to in-core or larger-than-core operation; however, additional information is given below for the basic manipulative subprograms including certain restrictions and applications. First it should be noted that, as indicated in Fig. 7, 20,000 words of storage are available in core for general computational useage with the 32 K core machine. Hence, the order of the largest square matrix having all nonzero element values that can be placed in core is $\sqrt{20{,}000} = 141$. In structural problems a matrix of this size is rather small so a matrix coding technique, described in Section IV, was adapted to increase the capability of in-core computation. Thus, the information given below for each manipulative subprogram assumes the matrices are coded unless otherwise stated.

A. Multiplication Subprogram (MULT)

This subprogram multiplies two matrices together where one of the two must fit in core. The multiplication is performed by code matching so neither matrix need be square or identically coded with the other. If none of the row and column codes matches then the product is a null matrix. A simple example of the multiplication process is shown in Fig. 8.

This subprogram can also multiply together uncoded matrices; however, the matrices must both fit in core simultaneously. Another option of this subprogram is serial multiplication in which up to 999 matrices (arbitrary limit) may be multiplied together by a single pseudo instruction.

B. Addition and Subtraction Subprogram (ADDS, SUBS)

This subprogram provides capability to add or subtract matrices in which neither matrix need fit in core. The matrices need not be square nor must codes of the two matrices necessarily match. Serial addition and subtraction is arbitrarily limited to 999 matrices similar to multiplication.

C. Transposition Subprogram (FLIP)

Virtually any size matrix can be transposed by this subprogram because the operation involves only interchanging the row and column codes of each element then relisting of the array. In a single pseudo instruction only one matrix can be transposed; however, a serial transposition option is not needed because the transposition is generally performed after summing or forming products of matrices.

D. Special Multiplication Subprogram (WASH)

One function of this subprogram is to partition (with attendant element scaling if required) a matrix, say $[M]$, by row and column. This is accomplished by forming a matrix triple product $[T]^T [M] [T]$, where the matrix $[T]$ is diagonal having either zero or nonzero (unity if no scaling is required) valued diagonal elements. The parent matrix $[M]$ must fit in core; however, it need not be symmetric or square.

Additional functions have been incorporated in this subprogram to allow numerical scaling or extracting of elements of $[M]$. This operation is performed by element code matching between $[M]$ and a control matrix. The parent matrix $[M]$ is restricted in this operation as in row-column partitioning.

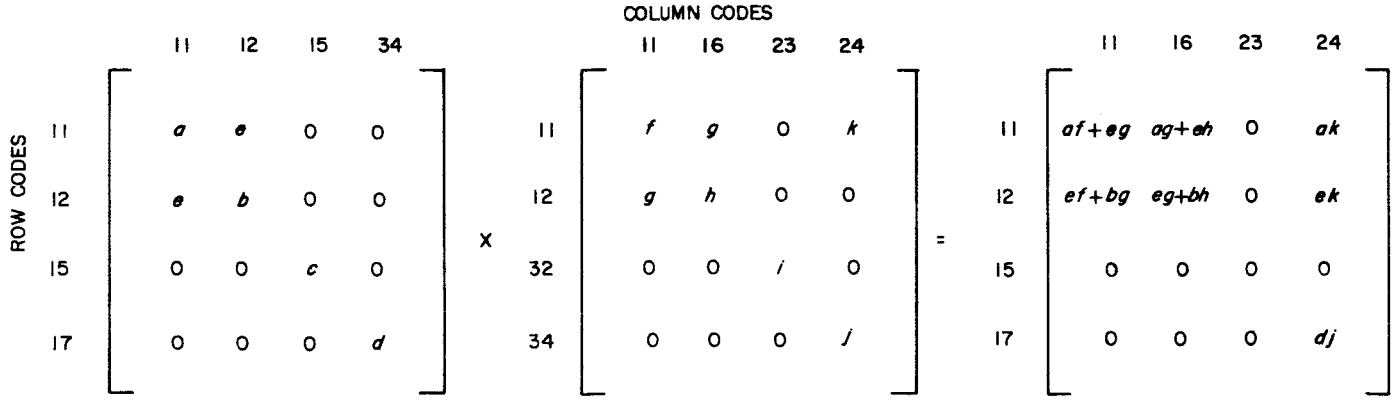


Fig. 8. Illustrative multiplication operation

E. Choleski Decomposition Subprogram (CHIN)

The purpose of this subprogram is to define a matrix which, when multiplied by its transpose, equals the original matrix. The particular form of these matrices is: one matrix has nonzero diagonal and upper off-diagonal elements, and the second is the transpose of the first. That is,

$$\begin{bmatrix} \diagdown & U \\ 0 & \diagdown \end{bmatrix}^T \begin{bmatrix} \diagdown & U \\ 0 & \diagdown \end{bmatrix} = [M]$$

where $[0 \diagdown U]$ is the upper triangular matrix, and $[0 \diagdown U]^T = [U \diagdown 0]$ is the lower triangular matrix. The advantage in finding the matrix $[0 \diagdown U]$ in triangular form is that it can be inverted very simply compared to inverting matrix $[M]$ directly. Once the inverse of $[0 \diagdown U]$ is found, the inverse of $[M]$ is determined by multiplication, that is,

$$\begin{bmatrix} \diagdown & U \\ 0 & \diagdown \end{bmatrix}^{-1} \begin{bmatrix} \diagdown & U \\ 0 & \diagdown \end{bmatrix}^{-1^T} = [M]^{-1}$$

where $[0 \diagdown U]^{-1}$ is the inverse of $[0 \diagdown U]$.

This subprogram is limited to symmetric, positive definite, square matrices. Because of the symmetry condition, only the diagonal and upper off-diagonal elements of $[M]$ must fit in the core of the computer in variable band form.[1] Thus, matrices whose complete array is larger than core can be manipulated; however, there is a definite upper limit on matrix size based upon the number of elements in the upper triangular region and on the diagonal. For example, the matrix of the largest order that can be manipulated by this subprogram is 6,666, in which only the diagonal elements are nonzero. Any matrix having nonzero off-diagonal elements must be of lesser order than 6,666 to fit in core.

A typical application of the decomposition function is in solving the dynamic matrix equation. The fundamental form of the equation is:

$$[M]\{\delta\} = \frac{1}{\omega^2}[K]\{\delta\}$$

where, in general, the matrices $[M]$ and $[K]$ are square and symmetric. By means of the CHIN subprogram we can decompose the stiffness matrix to obtain:

$$[M]\{\delta\} = \frac{1}{\omega^2}\begin{bmatrix} \diagdown & U \\ 0 & \diagdown \end{bmatrix}^T \begin{bmatrix} \diagdown & U \\ 0 & \diagdown \end{bmatrix}\{\delta\}$$

where

$$\begin{bmatrix} \diagdown & U \\ 0 & \diagdown \end{bmatrix}^T \begin{bmatrix} \diagdown & U \\ 0 & \diagdown \end{bmatrix} = [K]$$

We now define a modified deflection vector as:

$$\{\delta^*\} = \begin{bmatrix} \diagdown & U \\ 0 & \diagdown \end{bmatrix}\{\delta\}$$

or

$$\{\delta\} = \begin{bmatrix} \diagdown & U \\ 0 & \diagdown \end{bmatrix}^{-1}\{\delta^*\}$$

which when substituted into the dynamic equation yields

$$[M]\begin{bmatrix} \diagdown & U \\ 0 & \diagdown \end{bmatrix}^{-1}\{\delta^*\} = \frac{1}{\omega^2}\begin{bmatrix} \diagdown & U \\ 0 & \diagdown \end{bmatrix}^T\{\delta^*\}$$

Premultiplying this equation by $[0 \diagdown U]^{-1^T}$, we obtain

$$\begin{bmatrix} \diagdown & U \\ 0 & \diagdown \end{bmatrix}^{-1^T}[M]\begin{bmatrix} \diagdown & U \\ 0 & \diagdown \end{bmatrix}^{-1}\{\delta^*\} = \frac{1}{\omega^2}[I]\{\delta^*\}$$

where $[I]$ is the identity matrix. The advantage in decomposing the stiffness matrix in the above example is that the matrix triple product $[0 \diagdown U]^{-1^T}[M][0 \diagdown U]^{-1}$ yields a matrix that is symmetric, provided $[M]$ is symmetric. Insuring a final matrix that is symmetric allows subsequent use of more efficient mathematical techniques for finding the roots and vectors in solving the problem. The alternative to the above approach is to form the matrix product $[K]^{-1}[M]$ which is not symmetric.

F. Simultaneous Equation Solution Subprogram (CHOL, ITER)

The CHOL subprogram uses the Choelski triangular decomposition technique incorporated in CHIN to solve a set of simultaneous equations. The basic operation is outlined as follows. Given a matrix equation with $\{\delta\}$ the unknowns:

$$[K]\{\delta\} = \{P\}$$

the CHOL subprogram calculates the solution as

$$\{\delta\} = [K]^{-1}\{P\}$$

where the inverse of $[K]$ is found by the decomposition procedure.

Limitations on the calculation are that the $[K]$ matrix must be square, positive definite, and symmetric. Because

[1]Assuming symmetric matrices the bandwidth of a row is merely twice the number of locations between the diagonal element and the last nonzero off-diagonal element. In general, the row-bandwidth values should be made minimal to reduce calculations and provide for handling of larger order matrices (see Section V).

of the symmetry condition only the diagonal and upper off-diagonal elements are read into core (in variable bandwidth) so the total $[K]$ matrix can be larger than core. A follow-on phase of the contractual work is to develop this subprogram for operation on matrices that do exceed core capacity, by performing the necessary calculations sequentially so that the problem of filling core is not a factor.

The ITER subprogram also solves the equation $[K]\{\delta\} = \{P\}$ but uses the Accelerated Seidel Iteration Method. Limitations on this method are that $[K]$ must be square, positive definite, and have nonzero diagonal elements. However, the matrix $[K]$ need not fit in core nor be symmetric. Therefore, the ITER subprogram is capable of solving a more general class of equations than CHOL, but the computational time for ITER is generally significantly greater than that for CHOL.

Basing the capacity of the programs on the size of $[K]$ is not expressing the true limitation of the system. This can be demonstrated very easily. Assume the storage required for the diagonal and upper off-diagonal elements of $[K]$ exceeds core. A procedure called partitioning can be used effectively to allow use of the CHOL subprogram in preference to ITER. The original matrix equation to be solved is

$$[K]\{\delta\} = \{P\}$$

Assume this equation is partitioned into two equal parts as follows:

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{Bmatrix} \delta_1 \\ \delta_2 \end{Bmatrix} = \begin{Bmatrix} P_1 \\ P_2 \end{Bmatrix}$$

This arrangement is actually an array of two matrix equations, namely:

$$[K_{11}]\{\delta_1\} + [K_{12}]\{\delta_2\} = \{P_1\}$$

$$[K_{21}]\{\delta_1\} + [K_{22}]\{\delta_2\} = \{P_2\}$$

Solving the second equation for $\{\delta_2\}$ yields

$$\{\delta_2\} = [K_{22}]^{-1}\{P_2\} - [K_{22}]^{-1}[K_{21}]\{\delta_1\}$$

Substituting into the first equation gives

$$[[K_{11}] - [K_{12}][K_{22}]^{-1}[K_{21}]]\{\delta_1\} = \{P_1\} - [K_{12}][K_{22}]^{-1}\{P_2\}$$

or

$$[\bar{K}]\{\delta_1\} = \{\bar{P}\}$$

In this equation $[\bar{K}]$ is of the same order as $[K_{11}]$ and $[K_{22}]$ which are one-half the order of the original matrix $[K]$. Thus, CHOL can be used to determine $[K_{22}]^{-1}$, then to solve the final equation for $\{\delta_1\}$ by inverting $[\bar{K}]$.

Thus, it is apparent that by the method of partitioning, matrix equations that exceed core storage capacity can be solved; however, several matrix manipulations must be performed in place of one.

G. Root Subprogram (ROOT)

The function of this subprogram is to determine the characteristic roots (λ) and associated eigenvectors $\{\delta\}$ from input of the matrix $[R]$, where $[R]$ is defined by

$$\left[[R] - \lambda^2[I]\right]\{\delta\} = \{0\}$$

Referring back to the derivation of the dynamic equation outlined in the CHIN description, it may be concluded that

$$[R] = \begin{bmatrix} U \\ \diagdown \\ 0 \end{bmatrix}^{-1^T} [M] \begin{bmatrix} U \\ \diagdown \\ 0 \end{bmatrix}^{-1}$$

and

$$\lambda^2 = \frac{1}{\omega^2}$$

The algorithm used in this subprogram is Jacobi's Method in which requirements on the input matrix $[R]$ are that it be square, symmetric, and of order 130 or less.

H. Second-Order Differential Equation Subprograms (LOCI, DEQS)

Consider a second-order matrix differential equation of the form:

$$[A]\{\ddot{X}\} + [B]\{\dot{X}\} + [C]\{X\} = \{P\}$$

The subprograms LOCI and DEQS operate with this form of equation in two distinct ways. The function of the subprogram LOCI is to determine the change in value of the roots of the homogeneous equation ($\{P\} = \{0\}$) under some parameter variation. Any parameter in any two of the three matrices $[A]$, $[B]$, or $[C]$ can be varied arbitrarily. The information found by the LOCI subprogram can be interpreted as the transfer function of the system described by the second-order equation.

The function of the DEQS subprogram is to solve digitally the second-order equation when the forcing function is nonzero. Several functional forms can be selected for

the forcing function including the step, ramp, or sinusoidal types. In addition, an arbitrary forcing function can be input by representing the time history of the force by a sequence of impulses (Fig. 9). The forcing function can be an actual force, or a specification of defined displacements, velocities, or accelerations acting on the system. Both the LOCI and DEQS subprograms are limited to matrices of 40th order or less. The Muller method is used to isolate the roots in the LOCI subprogram, and the Runge-Kutta method is used to perform the integration in the DEQS subprogram.

I. Root Extractor Subprogram (POWR)

This subprogram uses the power method to determine the fundamental root and associated vector of a square, symmetric matrix. The matrix is limited to in-core sizes in the sense that only the diagonal and upper off-diagonal elements in variable bandwidth must fit in core. The principal use of this subprogram is to determine the fundamental buckling mode characteristics, or the fundamental eigenvalue and eigenvector in a dynamic calculation.

These subprograms comprise the basic manipulative package of the SAS program. All are limited to calculations with linear systems, and, except for LOCI and DEQS, all involve only real algebra. The algorithms used in each subprogram are considered efficient for applications on digital computers and each is documented completely. It is worth noting also that although these subprograms have certain limitations and restrictions, by

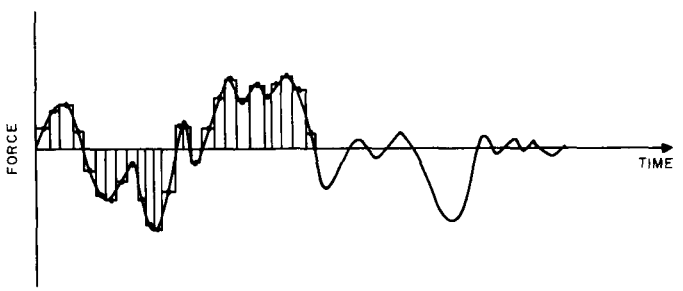


Fig. 9. Discretization of nonharmonic force

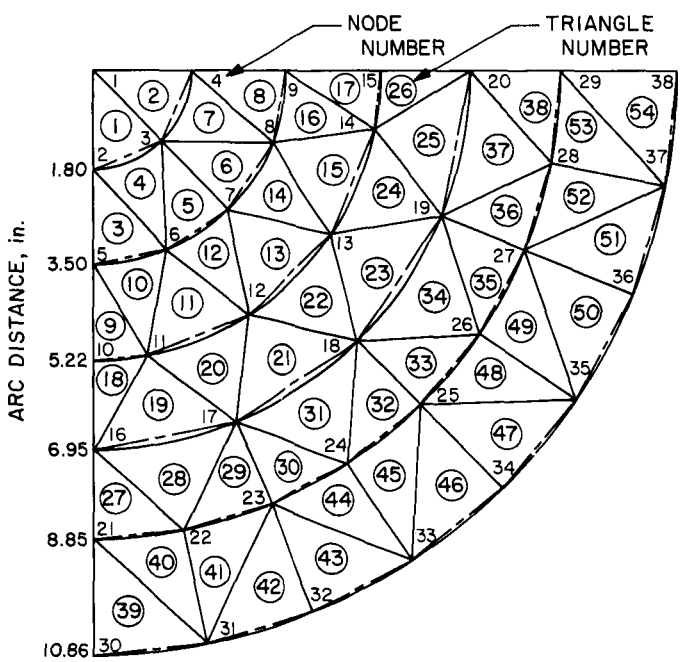


Fig. 10. Triangular grid array for quarter shallow shell

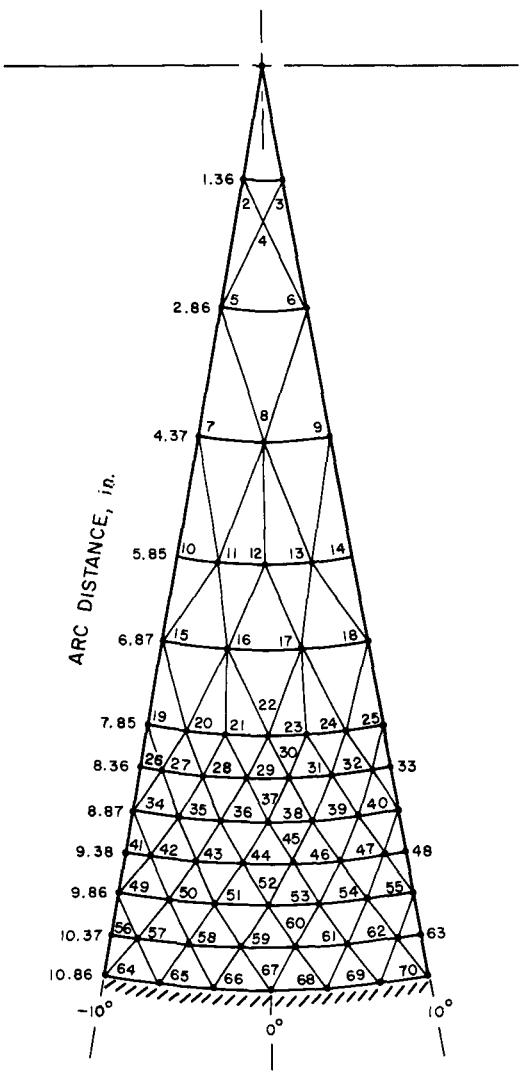


Fig. 11. Triangular array of 20-deg shell sector

judicious use of symmetry conditions, partitioning techniques, structural idealization tradeoffs, etc., these constraints can be circumvented in many problems to obtain accurate solutions.

System capability is reflected also in the time expended in solving a typical problem. Information is in process of being compiled on operation time versus data size for each of the subprograms of the SAS; however, at the time of this writing the compilation is not complete. Some indication of program time-of-operation capability can be given by citing the observed results of the Phase I test problem, which included both a static and dynamic calculation. The dynamic calculation involved finding the flexural natural frequencies of an unconstrained shallow spherical shell. In the setup of the problem it was recognized that the low frequency modes possessed a symmetry property that allowed analysis of only one-quarter of the shell. This sector was idealized by 38 triangular elements having 54 discrete nodes, resulting initially in a stiffness matrix of order 324 (Fig. 10). The time used to generate the stiffness and mass matrices and sum these was approximately 2.5 min. Imposition of boundary conditions, transformation of matrices as depicted in the CHIN description, inversion, determination of eigenvalues and eigenvectors, and inverse transformation of the vectors required 30 min. Of this time, 5 min were used in determining the roots and vectors of an 88th-order matrix, 5 min were used in inverting a matrix of 191st order, and significant time was used in required pre- and post-multiplication operations. In this computation 24 pseudo instructions were executed.

The second calculation of the test problem involved the computation of deflections and stresses for the same shallow shell used in the dynamic calculation except its outer edge was clamped. The loading conditions and shell geometry are defined in Section V. The triangular grid array selected for the shell is shown in Fig. 11. This array has 70 nodes, so the initial stiffness matrix order is 420. The generation of stiffness and loading matrices and subsequent summation of these matrices for the 108 triangles required 3.0 min. Solution of the system of simultaneous equations and computation of stresses required 20 min. Solution of the simultaneous equations, which were of order 307, required 6.0 min to complete.

Subsequent to the running of these problems several modifications have been incorporated in the SAS to improve its time efficiency in certain areas. For example, major changes have been made to the MULT subprogram to reduce its overall operation time. In addition, modifications to ADDS, SUBS, SORT, CHOL, CHEX, ROOT, WASH, and BILD are either in process or are planned to increase their capabilities.

IV. REVIEW OF OBJECTIVES IN PROGRAM DEVELOPMENT

A. Techniques of Achieving a "Balanced" Program

The word balanced is used here to imply development of a computer program that possesses equality in level of performance in the areas of complexity of the problems that can be solved, the time expended in computations and data handling, and the accuracy achieved in the results. In a contractual sense, expressing this condition by explicit constraints is very difficult to do except to set guidelines to dissuade strong effort in one area with obvious disregard for other of the major problem areas. It is felt that this balance of operations, so difficult to assess except by observation of the final product, has been achieved in the SAS program and is a definite credit to the developers of the program.

In initial definition of program development objectives, deciding upon the level of problem complexity was not difficult because of several natural limits. First, considering the triangular element, it has six variables per node (as compared with two for a planar beam element), so that any significant problem will entail use of a large number of variables. Hence, it was established that the program would be designed to solve, efficiently, structural problems in which the stiffness matrices varied from 100th to 2,500th order. For matrices smaller than 100th order the calculations can be performed easily in-core, for which many structural analysis systems are available. For matrices larger than 2,500th order, the capacity of present computer systems is probably exceeded, in that the time required for problem solution becomes excessive and accumulating errors are likely to destroy all accuracy. Hence, the SAS is designed to handle what may be termed intermediate-sized matrix problems.

Within these constraints on size, core capacity may be exceeded in a single calculation; hence subprogram logic must be formulated for in-core as well as larger-than-core calculations. When core is filled the only way to account for remaining data is to use auxiliary storage. It is well known that when auxiliary magnetic tape storage is coupled with in-core calculations the computational time increases significantly because of tape search and read times. However, presently, there is no alternative but to use this storage option so the time for operations must be minimized within this constraint.

In the SAS, time minimizing is achieved by performing tape search and rewinding operations on as near a non-interference basis as possible consistent with the JPL computer system. Every advantage has been taken of current computer capabilities to minimize tape sequencing; however, this remains a major contribution to total computational time. Within the limitations of the present computer system the tape problem could be alleviated by writing an independent computer monitor system, specifically adapted to the SAS. However, this approach is not compatible with job sequencing practices at the JPL computer center.

Fortunately the developers of computer hardware recognize that this tape-core flow problem is a major limitation on computer performance. Consequently, they have introduced several new components and a new monitor system to increase storage capacity and decrease data access time. Disk filing reduces data access time and increases storage capacity, and the FORTRAN IV monitor system allows for tape reading and writing while calculations are performed in-core (buffering). These improvements are not yet used in the SAS program, but when they are incorporated, operation and computation times will be reduced significantly. In planning for eventual conversion of the SAS to new monitor systems, the developers have used the FORTRAN language (rather than machine language) in approximately 95% of the SAS subprograms. This strong emphasis on FORTRAN is justified not only by the increased simplicity of adapting the program to different monitor systems, but also by the ease of interpretation of subprogram functions by a user not completely familiar with the program.

In addition to flow problems from subprogram to subprogram, and between core and tape, consideration was given to minimization of the quantity of data manipulated. Stiffness matrices, coordinate transformation matrices and mass matrices are by nature sparse matrices, having considerably more zero-valued off-diagonal elements than nonzero values. Therefore, it is efficient both in core utilization and in input data preparation to ignore the zero-valued elements. Ignoring zero-valued elements cannot be achieved easily using index incrementing techniques which operate with complete matrix arrays. However, it can be accomplished by assigning to each nonzero element a separate code that identifies the element by row and column. By this technique two words are needed for each matrix element (code and value), but since the matrices are sparse, the result is a net reduction in core storage used per matrix (Fig. 12). Basically, matrix algebra

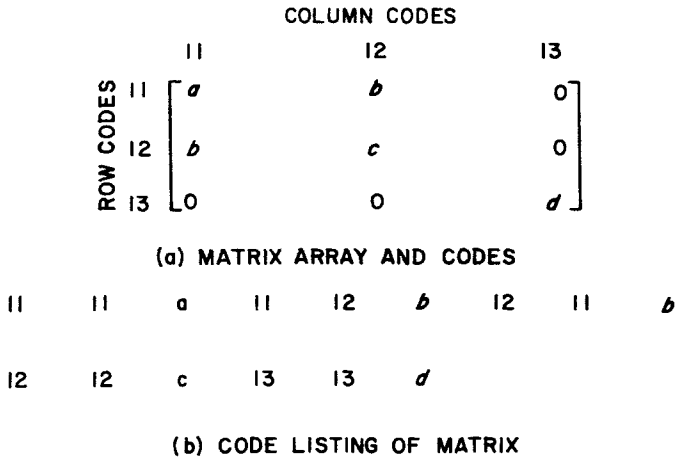


Fig. 12. Illustration of matrix coding technique

is carried out by "code matching," so that overall dimensional compatibility of matrices is not required. For example, in matrix addition, elements of each matrix with matching codes are added while the remainder of elements with nonmatching codes are relisted in the summed matrix (Fig. 13). Although this coding technique may seem to require considerable bookkeeping effort, the codes can be made up of identifying numbers that correspond to deflections or forces at nodes of the structure, so that code interpretation is apparent. In matrix problems not associated with structures, a rational coding system must be defined for easy identification; this, however, is not a difficult task.

The quantity of input data can be reduced by defining several optional coordinate systems that the user may select. For example, in a plate problem, coordinate axes with two axes in the plane of the plate reduce the geometry input data by one-third because only two coordinate values, instead of three, are needed to define the location of a node. This advantage does not exist, however, when analyzing doubly curved shells.

$$\begin{array}{c} \\ 11 \\ 12 \end{array}\!\!\begin{array}{c} \begin{array}{cc} 11 & 12 \end{array} \\ \begin{bmatrix} a & b \\ c & d \end{bmatrix} \end{array} + \begin{array}{c} \\ 12 \\ 15 \end{array}\!\!\begin{array}{c} \begin{array}{cc} 12 & 15 \end{array} \\ \begin{bmatrix} e & 0 \\ 0 & f \end{bmatrix} \end{array} = \begin{array}{c} \\ 11 \\ 12 \\ 15 \end{array}\!\!\begin{array}{c} \begin{array}{ccc} 11 & 12 & 15 \end{array} \\ \begin{bmatrix} a & b & 0 \\ c & d+e & 0 \\ 0 & 0 & f \end{bmatrix} \end{array}$$

Fig. 13. Matrix addition using coded elements

Computer operation time may be reduced also by minimizing the number of times a given amount of data is handled. In attempting to reduce the data handling time it became evident that mathematical partitioning rather than structural partitioning was more efficient. Structural partitioning involves grouping data by row and column, each group representing a segment of the structure. In structural partitioning the operations are reflected in the pseudo instruction program, which becomes very large. When this happens tape search times become a significant factor in reducing program efficiency.

Mathematical partitioning is accomplished by data handling techniques incorporated in each subprogram. This technique evolved from the condition that the subprograms manipulate matrices that either fit in-core or are larger-than-core. Basically, this technique involves partitioning of groups of data by row only, which is considered mathematical because no structural interpretation can be given. Since this operation is internal to the program, the user is unaware of the partitioning, so is not burdened with any data identification problems.

In considering program accuracy, it must be recognized that insuring accuracy over a wide spectrum of problem types and sizes is a difficult task. This fact was recognized at the onset of development of the SAS program, and considerable effort went into determining the possible errors that could occur and rational means of reducing them. A complete description and summary of the work done on error recognition and reduction is given in Ref. 4; hence, only a brief summary is included in this report. Basically five categories of errors are defined in Ref. 4, and each type is described briefly in following paragraphs.

One major source of error is human mistakes in preparation of input data, which may be designated *input error*. In preparing data for complex problems, there is generally considerable input of a repetitious nature, which is a situation conducive for errors. Fortunately, this type of error can be virtually eliminated by use of an input data diagnostic subprogram. The function of this subprogram is to check element data format, input matrix size and format, pseudo instruction format and tape assignments, and overall compatibility between these packages. A subprogram to perform this function has been developed for the SAS, and is designated the CHEX subprogram. This subprogram has also been coded to be operative as a separate diagnostic package on the IBM 1620 computer with disk filing. This provision allows the checkout of input data decks at reduced cost on the IBM 1620 before submitting the problem for solution on

the IBM 7094. The CHEX subprogram is a single or, in some cases, a two-pass system that detects most major errors in an input data deck and prints out appropriate comments that indicate each error made.

A quite different effort was devoted to reduction of error in structural representation. The stiffness method when applied to analysis of shell-type structures is an approximate method, in that the triangular element that represents the local structure is not an exact representation. For example, for the flat-plate-triangular-element used in the SAS, establishing continuity of deformations between elements requires the use of linear displacement functions which, however, will not allow exact representation of the bending mechanism. Also, internal forces in the structure are determined (lumped) only at the nodes of each triangle, so the exact distribution of forces within the element is not known, and an averaging technique must be used to define stresses. Other factors that influence this approximation are force equilibrium, stress continuity, and field compatibility. This type of error, namely, that the triangular element is not exact in representing local structure, is termed *discretization error.*

Means of reducing this type of error in the SAS centered on deriving an adequate triangular element representation. The scope of this task was not fully recognized at the start of development, and three distinct element representations were generated before an acceptable model was determined. A further complication was the condition that any element representation could not be evaluated until the generation package, as well as most of the data manipulative subprograms were functional and checked out.

The only means of reducing discretization error is to derive a better element representation than the one used. Thus, reduction of this error is contingent upon theoretical developments in the field of the finite element method. In anticipation of likely theoretical developments in finite element representations, the format of the generation subprogram has been set up so that element changes and modifications can be incorporated easily. The triangular element representation now used in the SAS is derived in the Program Technical Document (Ref. 4).

A third category of error, termed *idealization error,* is related to the manner in which the actual structure is idealized by the finite element array. In general, to minimize this error, the finite element breakdown should be coarser in regions where variables (deflections, curvatures, forces) change slowly and finer in regions where variables change rapidly. Orientation of triangles at boundaries, with respect to material and stress axes, and in regions of concentrated loads are other factors that should be considered in defining the triangular array. Hence, it is apparent that much of the control of this error is a function of the initial problem setup, which is predicated upon the knowledge and experience of the user. A potential user might argue that as the grid size is reduced the representation approaches an exact one, and accurate answers are assured (Fig. 14). However, there are two conditions on this argument. First, if the grid size is reduced, the amount of input data increases and the sizes of matrices that must be manipulated increase possibly to an extent that the calculations exceed current computer capability. Second, when the matrices become large, the number of manipulations increases, and an error develops that is inherent in any repetitious calculation; namely, round-off error. This error is called the *manipulative error* and is the fourth type reported in Ref. 4.

The manipulative error is basically a function of the conditioning of the matrices being manipulated, the mathematical procedures used in each subprogram, and the accuracy limits set on the computer. In the SAS program, due to operating time constraints, the accuracy limits have been established at eight places or what is termed "single precision" accuracy of the computer. The mathematical procedures used in the various subprograms of the SAS are based upon mathematical algorithms that are known to be efficient in computer applications. With regard to matrix conditioning, in structures problems, conditioning is normally good because of high attenuation in structural coupling. This effect results in matrices that are "near diagonal," that is, only the off-diagonal elements near the main diagonal are nonzero, and are numerically less than their respective diagonal element values. Since the question of matrix conditioning depends upon the particular characteristics of individual matrices, it is

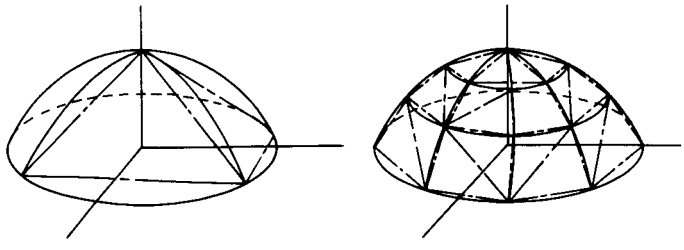


Fig. 14. Structural idealization

difficult to establish methods of correction when ill-conditioning is encountered. In many cases, ill-conditioning is due to the element breakdown selected to represent the structure; hence a corrective measure is to reidealize the structure rather than consider changing any of the manipulative operations.

The fifth error classification is called *interpretation error* and is associated with the problem of interpreting the output or results of computations. Visualize, for a moment, preparing a large amount of input data for an extremely complicated problem, then feeding this data into the computer and half an hour later obtaining some answers. The question is now—how good are these answers? If approximate or estimated answers are not known from another source, then this question is not easy to answer. However, there are at least four controls that may be applied to interpretation of the results. First, the past experience of the program user may be helpful in establishing an "intuitive feel" for the range of answers expected. Second, equilibrium or orthogonality checks may be run to establish the validity of the results. This can be done readily by simply altering the pseudo instruction program. Third, the problem can be rerun with a finer or coarser element (triangular) grid to check convergence of the solution with the possibility of extrapolating the limiting answer if several solutions are obtained. Fourth, if an energy approach is used to define the mathematical model of the finite element, then in some cases potential and complementary energy approaches can be taken in which the answers obtained by the two approaches bracket the correct answer for the idealized structure. This procedure is called "the bounding technique" in finite element applications, and its use is predicated upon a rational energy approach being used to derive the mathematical model of a structural element. The bounding technique has been applied successfully to simple problems (Ref. 7), but remains a basic developmental task for complex elements such as the triangular element. At present, in the SAS, the first, second, and third control measures can be used to interpret results. The fourth control measure, the bounding technique, is being considered for a follow on effort after more information and experience is gained in using the program developed in the earlier phases of work.

B. Methods of Achieving Program Versatility

Reference has already been made to the modular or chain approach being coupled with the pseudo instruction program to form the basis for a flexible computer program. The ability to solve a wide spectrum of problems is reflected in this form of flexibility. One extreme in program applicability is that a "black box" approach can be taken if the user is always solving a particular type of problem. The setup for a program of this type involves establishing first an efficient sequence of pseudo instructions. Some error instructions as well as planned recovery points need to be included to allow for size variations in input data and recovery from computer terminating errors. Once the program is set up and checked out the ultimate user is merely instructed on input-output writeup and interpretation, and recovery procedures. With this approach the user is not required to know matrix algebra even though matrices may be required input. In the SAS, because the matrix coding system is related to the structural idealization, the input matrices take on the appearance of tabulated data so that interpretation is possible without explicit use of the word "matrix." The output data is also printed in tabular form so that an understanding of the meaning of the codes that accompany each element value is sufficient for interpretation.

The other extreme in program usage is in the solution of the general matrix problem. Applicability of the SAS to solve general problems is contingent upon the functions performed by the manipulative subprograms. Hence, the user must be familiar with matrix algebra and the options and capabilities of the SAS subprograms. This information is provided by complete documentation of the manipulative functions in the Program Usage Document (Ref. 3) and of the mathematical algorithms in the Program Technical Document (Ref. 4). By means of this option, problems in any discipline may be solved using the SAS, provided the solutions can be effected using matrix algebra.

Options in required input data are another aspect of program flexibility and center on the capability of selecting coordinate systems that are compatible with the program being solved. In addition to latitude in selecting the coordinates to define structural geometry, other coordinate system options are:

1. Selection of elastic axes when working with monisotropic materials.
2. Specification of local coordinates for elements along which stresses are referenced.
3. Selection of coordinates along which deflections are referenced.

These coordinate options in the SAS are important in different problems and each may be used in accord with user preference or problem constraints.

The ease of incorporating extensions and modifications into the SAS program is also an important form of system flexibility. Basically there are two areas where this form of flexibility is needed. First, the modular format of the program library allows for addition of new subprograms by merely identifying the program in the master intelligence system for pseudo instruction interpretation, and assigning a chain number to the subprogram for locating it on the library tape. The other area where expansion is likely is in the generation subprogram library of stiffness elements. Expansion problems in this area center on changes in the input data. However, the input data format of the SAS has been planned and set up so that any type of element, including the three-dimensional solid, can be put in the library without any format changes.

C. Methods of Insuring Program Reliability

The type of reliability of concern here is that associated with the functioning of the entire program. Since a computer program is merely a set of instructions that the computer is slave to follow, once all of the logic and flow options in the instructions have been tested and proven to be correct, reliability is assured in the sense that a submitted problem will be properly executed. The only variants are then the validity of the input data and the functioning of the computer itself, which occasionally errors, for which the recovery feature is designed to accommodate.

For large programs with numerous flow patterns and options, such as the SAS, checkout is an extensive effort. In the checkout of the SAS, three levels of problems were generated and run. The function of the first level of problems was to establish the rationale of the logic within each subprogram. Next, with the subprograms grouped in a chain library, the second level of problems checked the compatibility between subprograms (in various sequences). Finally, because the program was designed to manipulate large blocks of data that could fill and exceed core, checkout of program capacity dictated the third level of test problems. The test problem for this third level of checkout was controlled by contract agreement. The single problem that was to be run was formulated and set up by JPL personnel with WDL approval and supervision. The problem was run on the JPL computer complex, and the results were subject to JPL approval.

It was recognized that one comprehensive test problem could not be formulated that would check all options of the SAS. Therefore, a support effort was planned and subsequently integrated with overall program development. This activity by JPL personnel was intended to aid WDL in certain critical areas and orient JPL personnel on details of the program functions and operations. In addition to the checkout effort already described, this support function involved the following participation by JPL technical representatives: weekly technical meetings with WDL personnel to discuss any matters relating to program development, review of technical and programming documents as they were generated during development, and participation in writeup of some of the key subprograms of the SAS program. During this checkout phase, JPL assigned from one to three engineers and one programmer as needed to perform the various functions. In review, it is felt that this support effort not only effectively oriented the JPL technical personnel in understanding the SAS program operation, but also it served as a check and balance to increase the reliability and to speed up the checkout of the program.[2]

Checkout of system manipulative subprograms proceeded with the normal types of errors causing difficulty. Some of the program features were very useful in checkout, particularly the recovery feature. Cross checking of different subprogram outputs was used advantageously. For example, the output from the subprograms used to solve simultaneous equations (CHOL, ITER) was compared to the matrix decomposition subprogram (CHIN) by proper selection of the set of equations. Vector orthogonality and matrix symmetry checks, core dump and data tape analyses and other standard techniques were used to check the logic and flow of the program.

The other phase of the program, the generation package, was the source of two fundamental problems that impeded system checkout. These problems were the derivation of an adequate triangular element representation and the rational distributing of nodal forces over each triangle to obtain accurate stress values. The problems associated with the triangular element representation were presented in Section IV. Although displacement

[2]Several activities have resulted from JPL's participation that complement the primary work on program development. For example, JPL documentation of the test problem input data and solutions supplements the technical and usage reports prepared by WDL. During program checkout, additional small test problems were submitted by JPL personnel to verify the operation and flow of certain program options not tested by the comprehensive test problems. In having the opportunity to review the technical aspects of the methods used in the SAS, JPL personnel have been able to initiate supporting advanced development and research projects and to better plan the direction that should be taken in extending and improving the SAS program to meet future analysis needs.

values can be determined very accurately (Ref. 5), the determination of accurate stresses is a well-recognized problem in finite element methods, and for many element types no exact procedure has been developed. In the SAS, the stresses are average values determined by techniques reported in the Program Technical Document (Ref. 4).

V. ROLE OF THE ENGINEER IN STRUCTURAL ANALYSIS APPLICATIONS OF THE SAS PROGRAM

As is true in any computer program a certain amount of basic data must be supplied to the system to initiate the generation and solution routines. Understandably, since the elements used in shell-structure idealizations are more complicated than, say, the beam elements used in frame-structure analyses, the amount of input data for the SAS exceeds that of corresponding frame analysis programs. In fact, the beam, bar, and torque tube elements, which are types of elements needed in the analysis of frame structures, are a part of the element library in the SAS, and so constitute a fractional part of the total input.

In general, the procedures involved in establishing input for frame-type structures are well known, so in the following discussion emphasis will be placed on the problems of input preparation for shell-type structures. In setting up a shell-type problem for the computer, one of the first tasks of the analyst is establishing a triangular array that will adequately represent the characteristics of the structure. This must be done within the constraints on matrix size, and, consequently, may involve the use of: structural planes of symmetry, local refinements in triangular grid sizes, modal convergence techniques, mathematical partitioning, and other methods to avoid bulky manipulative operations. Once the triangular grid is established, the next step is the numbering of nodes. This is extremely important since the matrix row-bandwidths are established by the nature of this numbering sequence. In general, the optimal arrangement is that in which adjacent nodes have node numbers that are close in value. Procedures for testing the quality of particular node numbering arrangements are given in Ref. 4. These procedures have been defined thoroughly so that this calculation can be performed by engineering-aide personnel. After the triangular grid and node numbering sequence have been defined, the element input data can be prepared, also by engineering-aide personnel.

The remaining tasks of the engineer are to select the appropriate set of pseudo instructions and to write the variable transformation matrices (in tabular form.) The variable transformation matrices are used to impose the boundary and symmetry conditions of the problem and to eliminate any rigid-body modes. Several examples of variable transformation matrices that were used to solve the program test problems are given in Ref. 5.

The structural test problem of Phase I may be cited to point out some of the considerations that should go into setting up a problem for the SAS. The test problem was to determine the deflections and stresses of a shallow spherical shell that was clamped at its edge and subjected to two loading conditions: a uniform pressure loading and a thermal loading (Fig. 15).

Observing that the loadings are axisymmetric, hence, the deflections of the shell will be axisymmetric, it is important to make use of this symmetry to reduce the amount of input data. To subdivide a small sector of the shell rather than the entire shell amounts to a significant

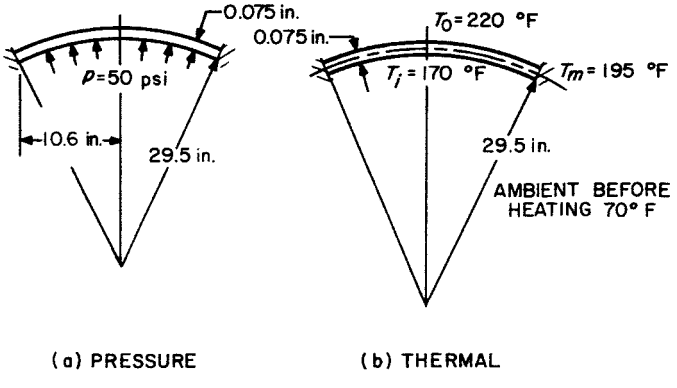


Fig. 15. Pressure and thermal loading

reduction in the number of triangles used in the idealization. However, if only a sector of the shell is idealized, then symmetry boundary conditions (along its meridional edges) must be imposed so that the influence of the part of the shell that is removed is reflected in the behavior of the sector.

In establishing the array of triangular elements, it was recognized that deflections and stresses would vary considerably more in the vicinity of the clamped edge than in the central region of the shell. For the test problem, a 20-deg sector of the shell was selected and was subdivided into the triangular array shown in Fig. 11. In general, values of stresses are referenced to the center-of-gravity of each triangle so that within the two-inch region next to the clamped edge eight distinct sets of stresses along the shell radius were determined.

Basically, the work involved in preparing the input data for this problem consisted of:

1. Establishing the triangular array for which the coordinate distances to each node were computed based upon a given reference coordinate system.
2. Numbering the nodes (Fig. 11).
3. Describing the material of the shell including elastic moduli, coefficient of thermal expansion, and mass density.
4. Selecting appropriate boundary and symmetry conditions and expressing these constraints in the form of a variable transformation matrix (Ref. 5).

This work, along with adapting the appropriate set of pseudo instructions can be completed in two or three days by an engineer and engineering-aide familiar with the data formats.

The SAS program has been developed purposely to possess the generality needed to incorporate many diverse structural conditions; however, with this advantage comes the disadvantage of a more complex input data format. That is, versatility in computer applications implies use of additional input specifications to accommodate the various program options. Consequently, the analyst may be required to supply more input data using the SAS than what he may consider a minimal amount. A good example of this is that a minimal input of elastic constants might be Young's Modulus and Poisson's Ratio when the material is isotropic. However, the stress-strain law used in the SAS is general enough to represent an aeolotropic material (13 elastic constant), so that the 13 constants must be computed for an aeolotropic material as well as for simpler material representations. However, for any given material this calculation need only be performed once, since these data can be retained in a material table which may be used in all subsequent problems as required.

Finally, it is well to mention that the work involved in preparing input data for a computer program such as the SAS should be weighed against any alternatives of finding solutions to the same accuracy by other methods. For structural problems involving laminated or stiffened structures, cutouts, concentrated or asymmetric loads, local support conditions, and other non-obliging conditions for closed form or numerical integration solutions, the SAS or any comparable program is the only means of obtaining approximate answers. Therefore, one additional task of the analyst is to decide if the problem to be solved is appropriate for the SAS program or should be solved using other methods or other computational tools of a combination of all.

Convergence on minimal analyst effort is the goal of most program developers, and it is anticipated that as the SAS program is updated by conditions found through usage, changes will be made that will refine and condense the input-output format. However, within the framework of a given computer system, input-output format can be condensed only so far, since a certain amount of data must invariably be supplied to the computer, which rates as a computational tool and not as a knowledgeable entity.

REFERENCES

1. Hurty, W. C., and Rubinstein, M. F., *Dynamics of Structures*, Prentice-Hall, Inc. Englewood Cliffs, N. J., 1964.

2. Gallagher, R. H., et al., *A Correlation Study of Methods of Matrix Structural Analysis*, Pergamon Press, New York, N. Y., 1964.

3. Melosh, R. J., et al., *Structural Analysis System Usage Report*, Document No. WDL-EM0763, Philco Corporation, Palo Alto, Calif., July, 1963.

4. Melosh, R. T., and Christiansen, H. N., *Structural Analysis System Technical Report*, Document No. WDL-EM0264, Philco Corporation, Palo Alto, Calif., February, 1964.

5. Lang, T. E., *Description of Test Problems for the Structural Analysis System Computer Program*, Jet Propulsion Laboratory, Pasadena, Calif. (to be published).

6. Percy, J. N., Pian, T. H. H., et al., *Application of Matrix Displacement Method to Linear Elastic Analysis of Shells of Revolution*, AIAA, 2nd Aerospace Sciences Meeting, New York, N. Y., January 25–27, 1965 (Preprint No. 65-142).

7. Melosh, R. J., *Development of the Stiffness Method to Define Bounds on Elastic Behavior of Structures*, University of Washington, Ph.D. Thesis, August, 1962, Seattle, Wash.

ACKNOWLEDGEMENTS

The author wishes to extend thanks to several individuals who contributed extensively to the JPL support function in checkout of the SAS program. These individuals are: Dr. C. Virgil Smith and Robert E. Reed for their valuable evaluations in certain technical areas, and Larry Schmele for programming contributions in subprogram development.