DYNAMIC ANALYSIS OF DISCRETE-ELEMENT PLATES ON NONLINEAR FOUNDATIONS

by

Allen E. Kelly
Hudson Matlock

Research Report Number 56-17

Development of Methods for Computer Simulation of Beam-Columns and Grid-Beam and Slab Systems

Research Project 3-5-63-56

conducted for

The Texas Highway Department

in cooperation with the
U. S. Department of Transportation
Federal Highway Administration

by the

CENTER FOR HIGHWAY RESEARCH
THE UNIVERSITY OF TEXAS AT AUSTIN

July 1970

PREFACE

A numerical method for the dynamic analysis of plates on nonlinear foundations was developed during this study. The method offers the highway engineer a rational approach for the solution of many plate and slab vibration problems, including pavement slabs and highway bridges which can be idealized as orthotropic plates.

The method was programmed and coded for use on a digital computer. Although the program was written for the Control Data Corporation (CDC) 6600 computer it can be made compatible with IBM 360 systems. Copies of the program presented in this report may be obtained from the Center for Highway Research at The University of Texas at Austin.

This work was sponsored by the Texas Highway Department in cooperation with the U. S. Department of Transportation Bureau of Public Roads, under Research Project 3-5-63-56. The Computation Center of The University of Texas at Austin contributed the computer time required for this study. The authors are grateful to these organizations and the many individuals who have assisted them during this study.

Allen E. Kelly
Hudson Matlock

July 1970

LIST OF REPORTS

Report No. 56-1, "A Finite-Element Method of Solution for Linearly Elastic Beam-Columns" by Hudson Matlock and T. Allan Haliburton, presents a finite-element solution for beam-columns that is a basic tool in subsequent reports.

Report No. 56-2, "A Computer Program to Analyze Bending of Bent Caps" by Hudson Matlock and Wayne B. Ingram, describes the application of the beam-column solution to the particular problem of bent caps.

Report No. 56-3, "A Finite-Element Method of Solution for Structural Frames" by Hudson Matlock and Berry Ray Grubbs, describes a solution for frames with no sway.

Report No. 56-4, "A Computer Program to Analyze Beam-Columns under Movable Loads" by Hudson Matlock and Thomas P. Taylor, describes the application of the beam-column solution to problems with any configuration of movable non-dynamic loaas.

Report No. 56-5, "A Finite-Element Method for Bending Analysis of Layered Structural Systems" by Wayne B. Ingram and Hudson Matlock, describes an alternating-direction iteration method for solving two-dimensional systems of layered grids-over-beams and plates-over-beams.

Report No. 56-6, "Discontinuous Orthotropic Plates and Pavement Slabs" by W. Ronald Hudson and Hudson Matlock, describes an alternating-direction iteration method for solving complex two-dimensional plate and slab problems with emphasis on pavement slabs.

Report No. 56-7, "A Finite-Element Analysis of Structural Frames" by T. Allan Haliburton and Hudson Matlock, describes a method of analysis for rectangular plane frames with three degrees of freedom at each joint.

Report No. 56-8, "A Finite-Element Method for Transverse Vibrations of Beams and Plates" by Harold Salani and Hudson Matlock, describes an implicit procedure for determining the transient and steady-state vibrations of beams and plates, including pavement slabs.

Report No. 56-9, "A Direct Computer Solution for Plates and Pavement Slabs" by C. Fred Stelzer, Jr., and W. Ronald Hudson, describes a direct method for solving complex two-dimensional plate and slab problems.

Report No. 56-10, "A Finite-Element Method of Analysis for Composite Beams" by Thomas P. Taylor and Hudson Matlock, describes a method of analysis for composite beams with any degree of horizontal shear interaction.

Report No. 56-11, "A Discrete-Element Solution of Plates and Pavement Slabs Using a Variable-Increment-Length Model" by Charles M. Pearre, III, and W. Ronald Hudson, presents a method of solving for the deflected shape of freely discontinuous plates and pavement slabs subjected to a variety of loads.

Report No. 56-12, "A Discrete-Element Method of Analysis for Combined Bending and Shear Deformations of a Beam" by David F. Tankersley and William P. Dawkins, presents a method of analysis for the combined effects of bending and shear deformations.

Report No. 56-13, "A Discrete-Element Method of Multiple-Loading Analysis for Two-Way Bridge Floor Slabs" by John J. Panak and Hudson Matlock, includes a procedure for analysis of two-way bridge floor slabs continuous over many supports.

Report No. 56-14, "A Direct Computer Solution for Plane Frames" by William P. Dawkins and John R. Ruser, Jr., presents a direct method of solution for the computer analysis of plane frame structures.

Report No. 56-15, "Experimental Verification of Discrete-Element Solutions for Plates and Slabs" by Sohan L. Agarwal and W. Ronald Hudson, presents a comparison of discrete-element solutions with the small-dimension test results for plates and slabs, along with some cyclic data on the slab.

Report No. 56-16, "Experimental Evaluation of Subgrade Modulus and Its Application in Model Slab Studies" by Qaiser S. Siddiqi and W. Ronald Hudson, describes an experimental program developed in the laboratory for the evaluation of the coefficient of subgrade reaction for use in the solution of small dimension slabs on layered foundations based on the discrete-element method.

Report No. 56-17, "Dynamic Analysis of Discrete-Element Plates on Nonlinear Foundations" by Allen E. Kelly and Hudson Matlock, presents a numerical method for the dynamic analysis of plates on nonlinear foundations.

ABSTRACT

This work describes a discrete-element method for the dynamic analysis of plates or slabs on nonlinear foundations. The method has been programmed for a high-speed digital computer and can be used to obtain solutions to a wide variety of plate vibration problems.

A step-by-step numerical integration procedure is employed to numerically integrate the solution in time. The assumption of a linear variation of acceleration during the time-step interval is utilized to develop a recursive solution procedure. Recommendations for the selection of the time-step increment, based on the stability analysis of the algorithm, are presented.

The nonlinear analysis is performed by an iteration procedure which adjusts the load rather than the foundation stiffness. This so-called load iteration method is presented as an alternative to the familiar stiffness adjustment procedures. Although the closure is slower with regard to the number of cycles required to reach equilibrium, a significant reduction in the computer time per cycle is realized by load iteration.

The program has been developed to accept a general variation in the elastic properties of the plate and in the nonlinear foundation characteristics. Furthermore there is considerable latitude in the description of the plan configuration and the dynamic loading.

Several example problems demonstrating the method are included, as is an example of the preparation of data for computer input.

KEY WORDS: mechanics, orthotropic bridges, slabs, vibration.

SUMMARY

The purpose of this work was the development of a numerical method for dynamic analysis of plates or slabs on nonlinear foundations. From the computer program which was developed, several example problems are presented to illustrate the validity of the numerical procedure and the potential application to highway bridge and pavement problems.

The computer program was developed to solve a model of the elastic slab consisting of rigid bars and elastic joints and torsion bars. This idealization, called a discrete-element model, has been successfully used to obtain static solutions to slab problems. The inertia properties of the plate were added to the static model in the form of lumped concentrated masses. Also added to the static model was a method to dissipate energy by viscous dampers or dashpots.

The numerical technique which was developed to propagate the dynamic response was, because of the nonlinear aspects of the problem a step-by-step method. Values of plate deflection, moment, and foundation forces are determined at discrete time intervals.

The response of the plate is first evaluated at time $t = t_o$. Information gained from the response at t_o is then utilized to determine the response at some time Δt from t_o. The numerical technique therefore steps ahead an amount Δt to obtain each new solution. For a bridge or pavement problem, many time steps may be required to determine the response of the structure to a moving load.

The selection of the time step increment Δt is an important factor in obtaining correct and meaningful results. Included in this report is a simplified formula to determine the maximum time step increment. This formula has as its variables the plate and foundation stiffness, the mass of the plate model, and the increment length selected for the model representation of the plate.

To facilitate the use of this program for highway problems, the user has been given a convenient tabular format for the organization of data for computer analysis. For example, only two (2) data cards are required for the

program to position on the slab a load moving with any velocity. As the procedure steps ahead in time, the load is automatically advanced at the correct speed.

The numerical method has been verified by solving several simple example problems. First, the free vibration of a simply supported plate was studied and the results from the program compared with theory. For this study, the difference between numerical and theoretical results was insignificant. Additional studies were run with moving loads, and again the results from the numerical procedure were very satisfactory. An example of a slab connecting the pavement with a bridge deck was studied. The foundation was idealized as a bilinear curve, resisting downward deflection but permitting lift-off. The results of this study showed the slab to lift free of the foundation and to oscillate about the static deflection curve. Peak deflections, however, were significantly greater than the static deflection.

IMPLEMENTATION STATEMENT

In this study, another tool has been developed for computer simulation and analysis of slab systems. The computer program described in this work may be used to study some of the dynamic effects of both moving loads and nonlinear foundation support for pavement slabs.

The problems associated with dynamic analysis of highway structures have long been untenable for the highway engineer. Although the use of impact factor to amplify the static load coupled with a static analysis has for years furnished the engineer a convenient design approximation, the dynamic reponse characteristics of the structure have remained submerged due to the extreme complication associated with the required dynamic analysis.

The potential application of this work ranges from sensitivity studies of rigid pavement dynamics to the review of impact factors for certain types of bridge structures. Furthermore, the coupling of research results of the pavement dynamics project with this program will make available to the highway engineer a procedure which will permit the dynamic study of the vehicle, slab, and foundation system.

Recommendations are made for further research in the area of pavement dynamics, especially in the area of the foundation characteristics. Either model tests or carefully controlled full scale tests should be performed to develop data for a correlation study of the numerical method. As more information becomes available about foundation properties, it will be possible to modify and extend the computer method presented in this work.

TABLE OF CONTENTS

NOMENCLATURE

Symbol	Typical Units	Definition
$a_{i,j}$	lb/in	Stiffness matrix coefficient
$\left[a_j\right]$	lb/in	Partitioned matrix of $a_{i,j}$ coefficients
A	-	Constant
$\left\{A_j\right\}$	-	Recursion coefficient vector
$\left[b_j\right]$	lb/in	Partitioned matrix of $b_{i,j}$ coefficients
$b^1_{i,j}$, $b^2_{i,j}$, $b^3_{i,j}$	lb/in	Stiffness matrix coefficients
$\left[B_j\right]$	-	Recursion coefficient matrix
c_d	lb-sec/in^3	Distributed viscous damping
$\left[c_j\right]$	lb/in	Partitioned matrix of $c_{i,j}$ coefficients
$c^1_{i,j}$, $c^2_{i,j}$, ... $c^5_{i,j}$	lb/in	Stiffness matrix coefficients
$\left[C_j\right]$	-	Recursion coefficient matrix
$\left[d_j\right]$	lb/in	Partitioned matrix of $d_{i,j}$ coefficients
$d^1_{i,j}$, $d^2_{i,j}$, $d^3_{i,j}$	lb/in	Stiffness matrix coefficients
D	lb-in^2/in	Isotropic plate bending stiffness

Symbol	Typical Units	Definition
D_x , D_y	lb-in^2/in	Plate bending stiffness in x and y-directions
D_{xy}	lb-in^2/in	Plate twisting stiffness
$\left[DF \right]$	lb-sec/in	Diagonal matrix of viscous damping coefficients
$\left[\overline{DF} \right]$	lb-sec/in	Uncoupled damping matrix for normal mode analysis
$DF_{i,j}$	lb-sec/in	Viscous damping coefficient
$e_{i,j}$	lb/in	Stiffness matrix coefficient
$\left[e_j \right]$	lb/in	Partitioned matrix of $e_{i,j}$ coefficients
E	lb/in^2	Modulus of elasticity
$\left[E_j \right]$	-	Recursion coefficient multiplier matrix
F_d	lb/in^2	Distributed damping force on middle plane of plate
F_m	lb/in^2	Distributed inertia force on middle plane of plate
F_R	lb	Restoring force for a single mode point displacement
h_t	sec	Magnitude of time step
h_x , h_y	in.	Discrete-element widths in x and y-directions
i	-	Node point identification associated with x-direction
I	-	Iteration index
j	-	Node point identification associated with y-direction
k	-	Time step identification

Symbols	Typical Units	Definition
$\left[K\right]$	lb/in	Stiffness matrix
$\left[K'\right]$	lb/in	Modified stiffness matrix
$\left[\bar{K}\right]$	lb/in	Uncoupled stiffness matrix for normal mode analysis
L	lb	Nonlinear correction load
$\left[M\right]$	$lb\text{-}sec^2/in$	Mass matrix
$\left[\bar{M}\right]$	$lb\text{-}sec^2/in$	Uncoupled mass matrix for normal mode analysis
$M_{i,j}$	$lb\text{-}sec^2/in$	Discrete node point mass
M_x , M_y	lb-in/in	Continuum bending moment
M_{xy} , M_{yx}	lb-in/in	Continuum twisting moment
M^x , M^y	lb-in	Model bending moment
M^{xy} , M^{yx}	lb-in	Model twisting moment
P_x , P_y	lb/in	Distributed axial thrust
P^x , P^y	lb	Concentrated axial thrust
$\left\{q_j\right\}$	lb	Partitioned load vector
$Q_{i,j}$	lb	Lateral load
$\left\{Q'\right\}$	lb	Modified load vector
$\left\{\bar{Q}\right\}$	lb	Load vector for normal mode analysis
s	lb/in^3	Distributed foundation support stiffness

Symbol	Typical Units	Definition
$S_{i,j}$	lb/in	Discrete foundation support stiffness
v	-	Slope of acceleration between time stations
V^x , V^y	lb	Shear force in plate model
w	in.	Deflection
$\dot{w}$	in/sec	Velocity
$\ddot{w}$	in/sec^2	Acceleration
W(i,j)	-	Mode shape
α	-	Product of bending stiffness and Poisson's ratio
η , $\dot{\eta}$, $\ddot{\eta}$	-	Normal coordinates
ν	-	Poisson's ratio (isotropic)
ν_x	-	Effect of x-curvature on that in y-direction
ν_y	-	Effect of y-curvature on that in x-direction
ρ	lb-sec^2/in^3	Distributed plate mass
$[\Phi]$	-	Normal mode matrix
ω	rad/sec	Frequency

CHAPTER 1. INTRODUCTION

This work presents a rational method for step-by-step dynamic analysis of orthotropic plates on nonlinear foundations and uses a method of nonlinear analysis in which the load vector is modified to reflect the nonlinearity, instead of altering the stiffness matrix of the mathematical model between iterations. This load iteration method is coupled with a linear acceleration algorithm for the development of the analysis procedure. Linear accelerations between each two stations in time are prescribed for model node points.

Motivation for development of the numerical procedure came from problems encountered in the field of highway engineering, particularly those related to pavement and bridge structures. The effect of vehicle motion on stresses and deflections of highway structures has long been an unknown factor. To focus on these highway problems, the method is applied to a bridge approach slab, and the type of nonlinearity studied is a special bilinear foundation behavior, to represent the loss of foundation support as the slab rises from the foundation.

A computer program is developed to demonstrate the method of analysis, using a simplified tabular imput form to describe the problem. While the dynamic loading must be specified by the user, either periodic or nonperiodic load, as well as stationary or moving, can be easily described. Furthermore, the foundation characteristics are described by curves composed of straight line segments.

Definition of the Problem

The effect of vehicle motion on highway structures has been a major concern of highway and airfield engineers for some time, and large-scale tests, such as the AASHO Road Test at Ottawa, Illinois (Ref 21), have served to focus attention on the need for a method for evaluating it.

The lack of agreement about the importance of dynamic effects is apparent, especially in the area of highway pavement design. At a recent special conference of the Highway Research Board, Harr suggested a review of the hypothesis that pavement loads are quasi-static and offered the possibility that energy is transmitted in all directions from the point of contact of the wheel with the pavement and may cause cracking and deterioration of the pavement at edges and other points where there is no foundation support (Ref 7). On the other hand, Jones et al have viewed the pavement problem as being essentially one of statics (Ref 12). The results of their investigations suggested that the dynamic effects are not significant, because of the great difference between the speed of a vehicle and the velocity of propagation of elastic waves.

Thus, it appeared that an analysis tool which would permit qualitative and quantitative study of some of the effects of dynamic loading on structural pavements would be useful in determining the significance of the loadings and, subsequently, in designing a wide range of structures. Therefore the development of such a tool was chosen as the problem to be considered in this study.

The Discrete-Element Analysis Procedure

Over a period of years, developments by various investigators have led to the discrete-element analysis procedure, which is the basis of the analysis described in this report. The concept of this use of a discrete-element model for plates can be traced to Ang and Newmark (Ref 3). Tucker extended the concept for beams to grid and plate structures, using an alternating-direction method as the basis for his work on solutions for the grid-beam structure of a plate (Ref 24), and later Ang and Prescott presented model equations for solving complex isotropic plate problems (Ref 2).

An orthotropic plate model was developed by Hudson in a study which extended the work of Tucker and refined the alternating-direction procedure for solving the large number of equilibrium equations generated by the mathematical model (Ref 9). A method for direct solution of these equilibrium equations developed by Stelzer takes advantage of the banded nature of the equations (Ref 20). In this method, the formulation of equilibrium equations results in a partitioned stiffness matrix with a submatrix band width of five, i.e., two submatrices on either side of the main diagonal partition.

A dynamic analysis of elastic plates based on a finite-difference method was developed by Salani (Ref 19). Using an alternating-direction implicit (ADI) iterative procedure, the transient and steady state response of isotropic plates can be determined.

Basis of the Method for Vibration Analysis

The discrete-element model presented in this report is Hudson's model extended to include mass and viscous clamping. The mass of the plate is lumped at stations or node points, and viscous damping is absolute; that is, each node point is connected to a fixed reference plane by a dashpot.

Solutions to the equations of motion are obtained at discrete points in time. An algorithm based on the assumption of linear acceleration between time stations is used to propagate the solution step-by-step. Nonlinear analysis is accomplished by iteration for equilibrium at each time step. A method is presented which does not require the adjustment of the stiffness matrix during the iterative procedure. Instead, the loading is modified to produce convergance to the equilibrium position.

Many techniques for step-by-step analysis of structural vibration have been suggested, ranging from mathematically oriented methods to methods based on assumptions of the nature of the motion between time steps (Refs 16, 19, and 27). The latter technique was selected for use in this study of response of plates on nonlinear foundations.

Application

The tool presented here permits qualitative and quantitative study of some of the effects of dynamic loading on structural pavement and certain types of bridges. With it a bridge can be idealized as a plate, which is more realistic than idealizing it as a beam, and the program is general enough to permit the study of a wide range of structures containing plate-like structures, floors of multi-story buildings, certain types of aircraft structures, and the behavior of such structural grids as those which make up the deck of an offshore drilling platform.

CHAPTER 2. EQUATIONS OF MOTION FOR DISCRETE-ELEMENT MODEL

The model for dynamic analysis is developed by the addition of mass and damping to node points of a discrete-element plate model. The equation of motion is derived by the addition of inertia and damping forces to the model load vector.

The equation of motion and the mathematical model presented in this chapter pertain to linearly elastic thin plates in which lateral deflections are small. Before the discrete-element model is considered, the classical theory for isotropic and orthotropic plates is reviewed. The relationship between the continuum plate equations and the discrete-element model can be demonstrated by application of finite-difference approximations to the continuum expressions.

Classical Equation of Motion

The classical equilibrium equation for a plate on an elastic foundation can be written as

$$\frac{\partial\partial M_x}{\partial x^2} - 2\,\frac{\partial^2 M_{xy}}{\partial x \partial y} + \frac{\partial M_{xy}}{\partial y^2} = q - sw + \frac{\partial}{\partial x}\left(P_x \frac{\partial w}{\partial x} \right)$$

$$+ \frac{\partial}{\partial y}\left(P_y \frac{\partial w}{\partial y} \right) \qquad (2.1)$$

The positive sense for the deflection is upward, which causes the difference in sign for the deflections and in-plane thrusts P from that given by Timoshenko (Ref 22). The equation is valid for either isotropic or orthotropic plates, as material properties do not influence the equilibrium expression.

The development of the equation of motion follows from D'Alembert's principle (Ref 26). An inertia force equal to the negative product of mass per unit area times acceleration is applied on a unit area of the middle plane of the plate:

$$F_m = -\rho \frac{\partial^2 w}{\partial t^2} \tag{2.2}$$

Viscous damping, included in the analysis, develops a force

$$F_d = -c_d \frac{\partial w}{\partial t} \tag{2.3}$$

on the middle plane of the plate.

Adding Eqs 2.2 and 2.3 to the equilibrium equation yields the equation of motion for a plate vibrating with small deflections:

$$\frac{\partial^2 M_x}{\partial x^2} - 2 \frac{\partial M_{xy}}{\partial x \partial y} + \frac{\partial^2 M_y}{\partial y^2} = -\rho \frac{\partial^2 w}{\partial t^2} - c_d \frac{\partial w}{\partial t} + q - sw$$

$$+ \frac{\partial}{\partial x}\left(P_x \frac{\partial w}{\partial x} \right) + \frac{\partial}{\partial y}\left(P_y \frac{\partial w}{\partial y} \right) \tag{2.4}$$

Bending moments in the isotropic plate are found by the familiar relationships:

$$M_x = D\left(\frac{\partial^2 w}{\partial x^2} + \nu \frac{\partial^2 w}{\partial y^2} \right) \tag{2.5a}$$

$$M_y = D\left(\frac{\partial^2 w}{\partial y^2} + \nu \frac{\partial^2 w}{\partial x^2} \right) \tag{2.5b}$$

$$M_{xy} = -D(1 - \nu) \frac{\partial^2 w}{\partial x \partial y} \tag{2.5c}$$

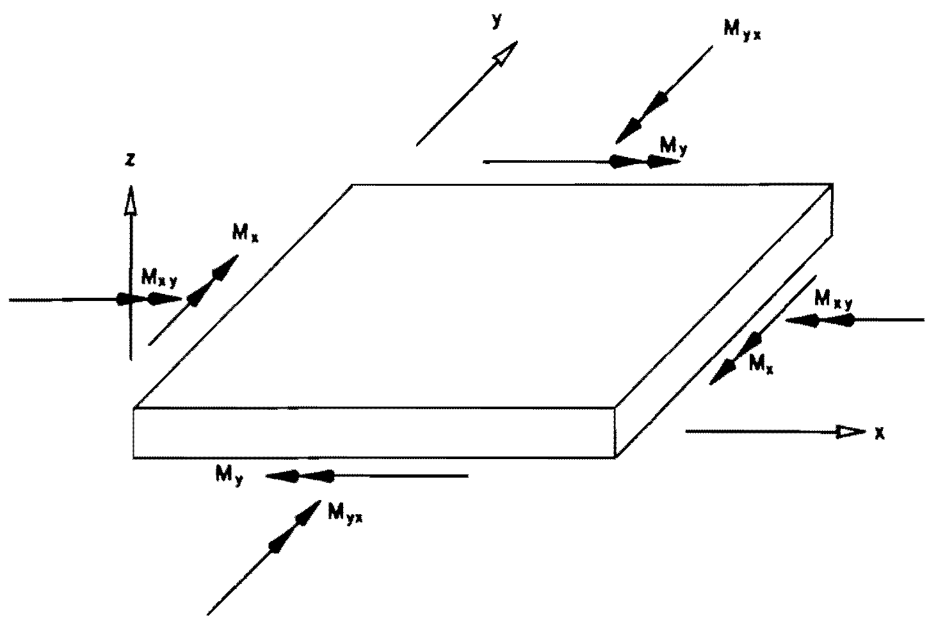


Fig 1. Direction of positive plate moments.

where

$$D = \frac{Et^3}{12(1 - \nu^2)} \tag{2.5d}$$

A sign change is noted when Eqs 2.5 are compared with the moment-curvature relationships given by Timoshenko. Again, this is due to a reversal of the positive w coordinate direction. The assumed positive moment directions for the plate are shown in Fig 1.

The bending moments in a plate of an orthotropic material are of a similar form (Ref 23):

$$M_x = D_x \left(\frac{\partial^2 w}{\partial x^2} + \nu_y \frac{\partial^2 w}{\partial y^2} \right) \tag{2.6a}$$

$$M_y = D_y \left(\frac{\partial^2 w}{\partial y^2} + \nu_x \frac{\partial^2 w}{\partial x^2} \right) \tag{2.6b}$$

$$M_{xy} = -D_{xy} \frac{\partial^2 w}{\partial x \partial y} \tag{2.6c}$$

Substituting the more general Eqs 2.6 into Eq 2.4 gives the equation of motion for orthotropic plates:

$$\frac{\partial^2}{\partial x^2}\left[D_x \left(\frac{\partial^2 w}{\partial x^2} + \nu_y \frac{\partial^2 w}{\partial y^2} \right) \right] + 2 \frac{\partial^2}{\partial x \partial y} \left(D_{xy} \frac{\partial^2 w}{\partial x \partial y} \right)$$

$$+ \frac{\partial^2}{\partial y^2}\left[D_y \left(\frac{\partial^2 w}{\partial y^2} + \nu_x \frac{\partial^2 w}{\partial x^2} \right) \right] = -\rho \frac{\partial^2 w}{\partial t^2} - c_d \frac{\partial w}{\partial t}$$

$$+ q - sw + \frac{\partial}{\partial x}\left(P_x \frac{\partial w}{\partial x} \right) + \frac{\partial}{\partial y}\left(P_y \frac{\partial w}{\partial y} \right) \tag{2.7}$$

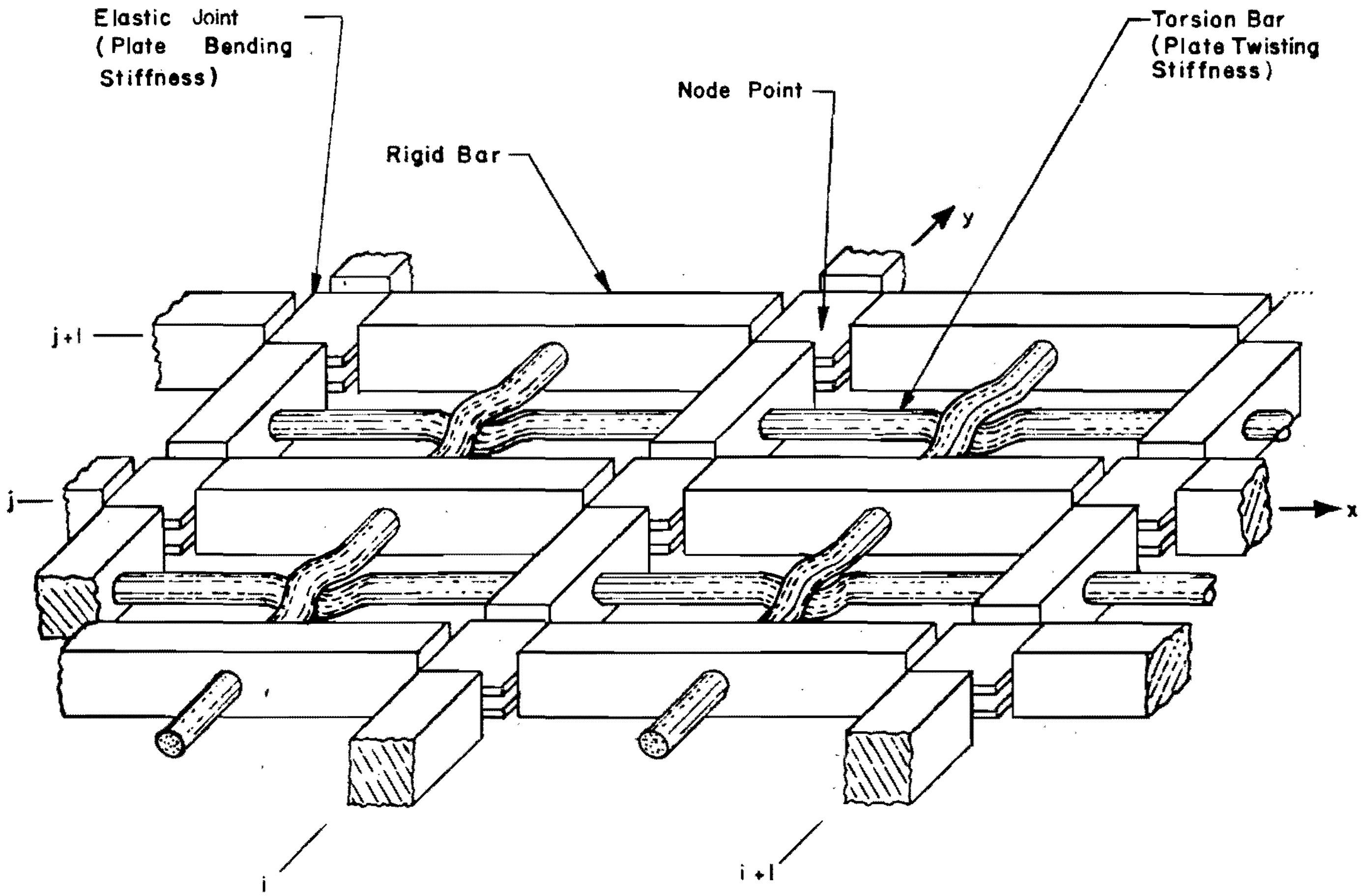


Fig 2. Discrete-element model of a plate or slab.

Volterra and Zachmanoglou have presented numerical solutions of Eq 2.7 for rectangular isotropic plates (Ref 26). However, the solution to Eq 2.7 becomes untenable for plates of variable stiffness and general support conditions.

Discrete-Element Model - Static Analysis

A relatively simple mathematical model of the orthotropic plate can be constructed from rigid bars and elastic elements which simulate bending and twisting properties of the plate. A convenient discrete-element plate model, shown in Fig 2, was developed by Hudson (Ref 9) and Ingram (Ref 11). Torsion bars simulate twisting characteristics of the plate while special elastic joints are used to develop bending properties. Motivation for development of this model stems from work by Matlock and Haliburton on discrete-element beams (Ref 14), in which a similar idealization of rigid bars and elastic joints was used to represent beams.

Derivation of the equilibrium equation for the discrete-element model is presented in Appendix A. In this development, the elastic joint and torsion bar properties are defined by applying a difference approximation to the moment expressions (Eq 2.6). It is important to note that the units of moment in the model are lb-in while the usual units of plate moment are lb-in/in. Furthermore, model moments are identified by superscript and continuum moments by subscripts x and y .

The equilibrium equation of Appendix A could have been derived directly from Eqs 2.1 and 2.6, with the substitution of difference approximations for the partial derivatives resulting in Eqs A.19 through A.31 of Appendix A.

Thus the discrete-element model plays a dual role. It stands by itself as a convenient structural idealization of a plate, and it can be related to the continuum equation by difference approximations and may therefore be viewed as a physical interpretation of the difference equations.

Node equilibrium equations can be combined and written in matrix notation:

$$\left[K\right]\left\{w\right\} = \left\{q\right\} \tag{2.8}$$

The terms of the stiffness matrix $\left[K\right]$ are deflection coefficients given by Eqs A.19 through A.31.

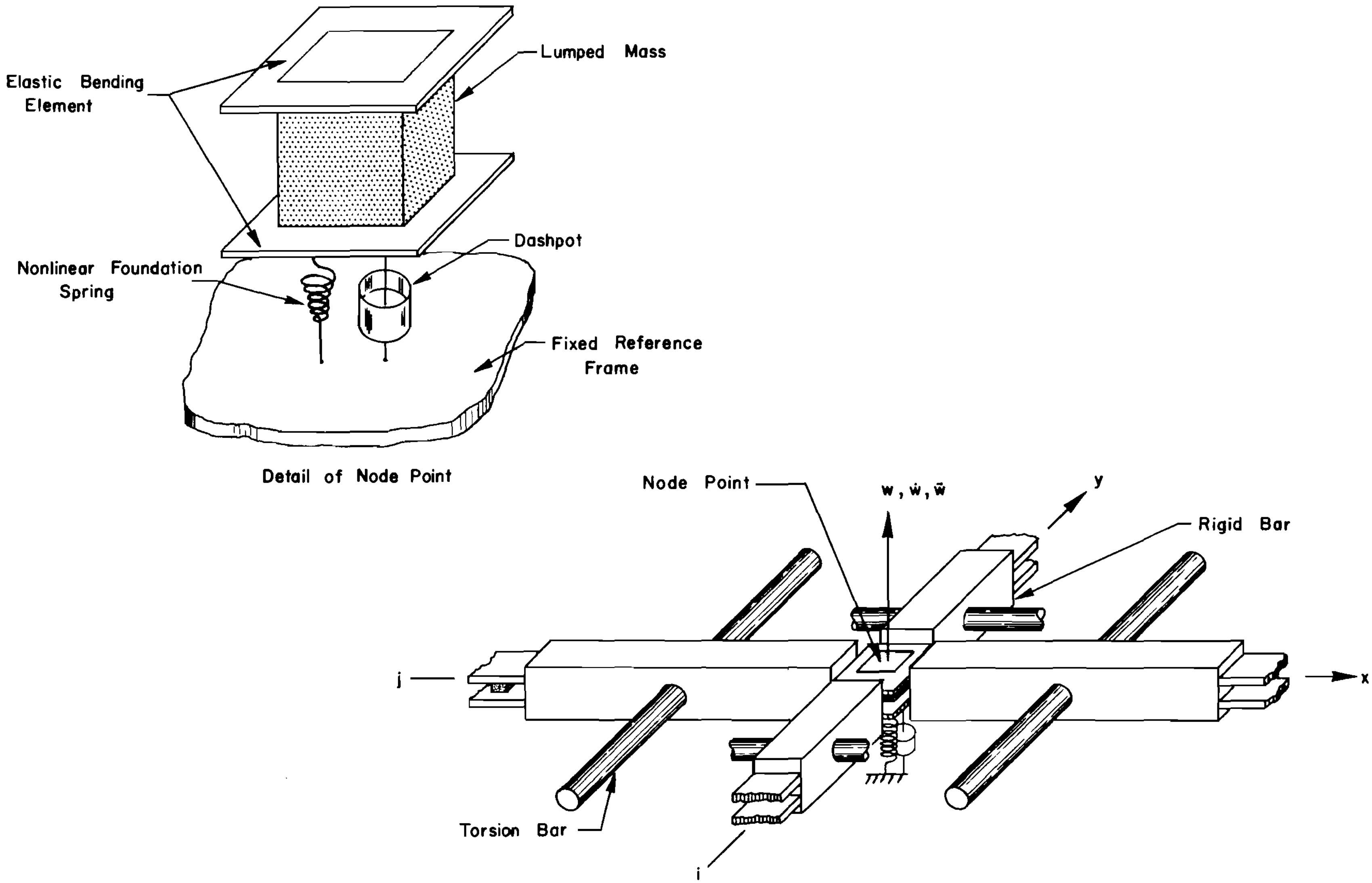


Fig 3. Joint detail of discrete-element model for dynamic analysis.

Discrete-Element Model - Dynamic Analysis

Details of the discrete-element model for dynamic analysis are presented in Fig 3. The rigid bars connecting joints are massless, with the mass of the plate concentrated at joints, or node points. As in the static model, foundation support springs are attached to the model at joints. Viscous dampers, represented as dashpots, are also connected to the joints and to a fixed reference plane.

Adding the inertia and damping forces to the right-hand side or load side of the static equilibrium equation yields the equation of motion for the model (see Appendix B). The equation of motion for each node can be combined and written in matrix form:

$$\left[M\right]\left\{\ddot{w}\right\} + \left[DF\right]\left\{\dot{w}\right\} + \left[K\right]\left\{w\right\} = \left\{q\right\} \qquad (2.9)$$

In Eq 2.9, the stiffness matrix $\left[K\right]$ is that given in Eq 2.8. Due to the idealization of concentrated mass and damping at joints, both the mass matrix $\left[M\right]$ and damping matrix $\left[DF\right]$ are diagonal.

Dynamic response of the discrete-element model is found by integrating Eq 2.9. A numerical method for the integration is presented in the following chapter.

CHAPTER 3. NUMERICAL INTEGRATION

The equations of motion are numerically integrated by an algorithm based on the assumption that the acceleration of each node has a linear variation during the time-step interval. It was necessary to use a step-by-step method because of the nonlinear foundation characteristics.

Numerical Analysis of Initial-Value Problems

An alternative to the step-by-step methods for vibration analysis is the normal mode method (Ref 10). This approach is attractive because the simultaneous equations describing the dynamic equilibrium of the structure are transformed into N independent, second-order differential equations, where N is the number of degrees of freedom of the structure. The analysis requires first the solution of the eigenvalue problem

$$\left[-\omega^2 \left[M \right] + \left[K \right] \right] \left\{ w \right\} = 0 \qquad (3.1)$$

for both the natural frequencies ω (eigenvalues) and the corresponding normal mode shapes (eigenvectors). The normal modes are related to the structural displacements w by multipliers termed normal coordinates.

$$\left\{ w \right\} = \left[\Phi \right] \left\{ \eta \right\} \qquad (3.2)$$

In Eq 3.2, each column of $\left[\Phi \right]$ is a normal mode and the normal coordinates η determine the contribution of each mode to the total response of the structure. Although the normal coordinates are time dependent variables, the normal mode matrix $\left[\Phi \right]$ is not.

The equations of motion (Eq 2.9) are uncoupled if Eq 3.2 is substituted for $\left\{ w \right\}$ and both sides of Eq 2.9 are post multiplied by the transpose of the normal mode matrix:

$$\left[\Phi\right]^t \left[M\right] \left[\Phi\right] \left\{\ddot{\eta}\right\} + \left[\Phi\right]^t \left[DF\right] \left[\Phi\right] \left\{\ddot{\eta}\right\}$$

$$+ \left[\Phi\right]^t \left[K\right] \left[\Phi\right] \left\{\ddot{\eta}\right\} = \left[\Phi\right]^t \left\{Q\right\} \tag{3.3}$$

or

$$\left[\overline{M}\right] \left\{\ddot{\eta}\right\} + \left[\overline{DF}\right] \left\{\ddot{\eta}\right\} + \left[\overline{K}\right] \left\{\eta\right\} = \left\{\overline{Q}\right\} \tag{3.4}$$

In Eq 3.4 $\left[\overline{M}\right]$, $\left[\overline{DF}\right]$, and $\left[\overline{K}\right]$ are diagonal matrices. However, for the matrix $\left[\overline{DF}\right]$ to be diagonal, $\left[DF\right]$ must be a function of either $\left[M\right]$ or $\left[K\right]$.

The single degree of freedom systems represented by Eq 3.4, are easily solved and then superposed, by means of Eq 3.2, to determine the total response of the structure.

Several features of this approach are appealing. First is the ease of solution of the uncoupled equations. Also, for many problems the excitation of the higher modes of vibration and their contribution to the dynamic response of a structure are insignificant. The investigator therefore needs only to compute the response of the fundamental and a few of the next higher modes to define adequately the structural response.

On the other hand, when the higher modes are important to the response, and the structure has many degrees of freedom, the operations required for vibration analysis will be very time consuming. Furthermore, if the damping matrix $\left[DF\right]$ is not a function of $\left[M\right]$ or $\left[K\right]$, the equations cannot be uncoupled. Finally, normal mode analysis must be limited to linear problems. A step-by-step integration method is therefore required for the analysis procedure presented in this study.

Development of the step-by-step methods can be traced to the use of finite-difference approximations. The problem, involving either derivatives or partial derivatives, is transformed from one with continuous variables to one in which the variables are defined at discrete points in time or space. In Chapter 2 it was shown that the discretization of the space coordinate can be modeled. The finite-difference approximation of the continuum plate equations was shown to represent a discrete-element structure. On the other hand, discretizations of the time coordinate are often more difficult to interpret. An example would be the substitution of central difference expressions for acceleration and velocity into the plate equations of motion given by Eq 2.9:

$$\frac{1}{h_t^2}\left[M\right]\left\{w_{k-2} - 2w_{k-1} + w_k\right\} + \frac{1}{2h_t}\left[DF\right]\left\{-w_{k-2} + w_k\right\}$$

$$+\left[K\right]\left\{w_k\right\} = \left\{Q_k\right\} \qquad (3.5)$$

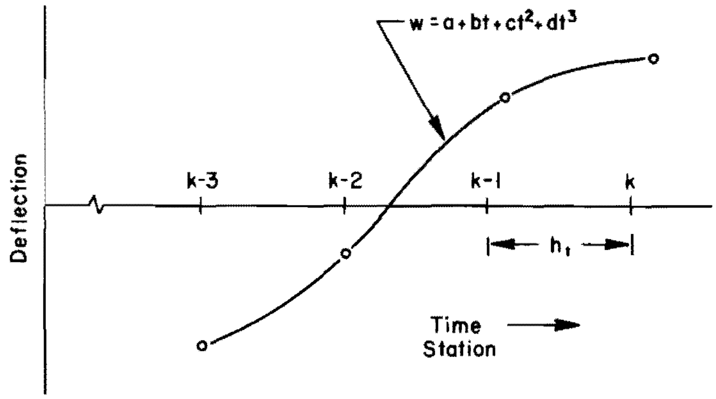
With small time steps (Ref 5), Eq 3.5 can be solved explicitly for w_k. However, the variation of the time-dependent deflection during the time interval h_t is not clear.

Other methods for step-by-step analysis yield direct physical interpretation of the nature of the displacement during the time interval h_t. In 1951 Houbolt published a numerical method for vibration analysis of lumped-mass systems (Ref 8). His approach was to pass a third-order curve through node displacements for four consecutive points in time (Fig 4). By differentiating the expression and evaluating the derivatives at the fourth point in time a backwards difference operator was developed. The third-order variation in deflection results in an acceleration which is linear during any time interval:

$$\ddot{w}_k = \frac{1}{h_t^2}\left(2w_k - 5w_{k-1} + 4w_{k-2} - w_{k-3}\right) \qquad (3.6a)$$

$$\dot{w}_k = \frac{1}{6h_k}\left(11w_k - 18w_{k-1} + 9w_{k-2} - 2w_{k-3}\right) \qquad (3.6b)$$

This method was successfully used by Tucker to determine the response of piles to wave loading (Ref 25). The analysis procedure presented by Houbolt leads to an implicit solution for the unknown deflection at the new time station:



Solving for the coefficients a, b, c, and d:

$$\begin{Bmatrix} w_k \\ w_{k-1} \\ w_{k-2} \\ w_{k-3} \end{Bmatrix} = \begin{bmatrix} 1 & 3h_t & 9h_t^2 & 27h_t^3 \\ 1 & 2h_t & 4h_t^2 & 8h_t^3 \\ 1 & h_t & h_t^2 & h_t^3 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} a \\ b \\ c \\ d \end{Bmatrix}$$

$$a = w_{k-3}$$

$$b = \frac{1}{6h_t}(2w_k - 9w_{k-1} + 18w_{k-2} - 11w_{k-3})$$

$$c = \frac{1}{2h_t^2}(-w_k + 4w_{k-1} - 5w_{k-2} + 2w_{k-3})$$

$$d = \frac{1}{6h_t^3}(w_k - 3w_{k-1} + 3w_{k-2} - w_{k-3})$$

Fig 4. Node-point deflection as a third-order function of time (after Houbolt, Ref 8).

$$\left[\frac{2}{h_t^2}\left[M\right] + \frac{11}{6h_t}\left[DF\right] + \left[K\right]\right] \left\{w_k\right\} = \left\{Q_k\right\}$$

$$- \frac{1}{h_t^2}\left[M\right]\left\{-5w_{k-1} + 4w_{k-2} - w_{k-3}\right\}$$

$$- \frac{1}{6h_t}\left[DF\right]\left\{-18w_{k-1} + 9w_{k-2} - 2w_{k-3}\right\} \quad (3.7)$$

Newmark, on the other hand, developed a powerful iterative technique for step-by-step analysis (Ref 16). The acceleration at the end of a time step is estimated, and the velocity and deflection are then calculated by

$$\dot{w}_k = \dot{w}_{k-1} + \frac{h_t}{2}\left(\ddot{w}_{k-1} + \ddot{w}_k\right) \quad (3.7a)$$

$$w_k = w_{k-1} + h_t\dot{w}_{k-1} + \left(\frac{1}{2} - \beta\right) h_t^2\ddot{w}_{k-1} + \beta h_t^2\ddot{w}_k \quad (3.7b)$$

The restoring and damping forces can then be determined at time k and a new estimate of acceleration can be computed:

$$\left[M\right]\left\{\ddot{w}_{k_I}\right\} = \left\{Q_k\right\} - \left[DF\right]\left\{\dot{w}_{k_{I-1}}\right\} - \left[K\right]\left\{w_{k_{I-1}}\right\} \quad (3.8)$$

With the new estimate of acceleration at time k, the process is repeated until the successive values of acceleration agree within a specified tolerance.

The parameter β in Eq 3.7 governs the influence of the acceleration at the end of the time interval $(\ddot{w}_k)$ on the displacement at that point. Furthermore, the value selected for β determines the variation of acceleration during the interval h_t. For $\beta = \frac{1}{6}$, the method becomes a linear acceleration assumption. A β-value of $\frac{1}{4}$ represents constant acceleration throughout the interval and $\beta = \frac{1}{8}$ may be interpreted as a step function having an acceleration $\ddot{w}_{k-1}$ over the first half of the time interval and $\ddot{w}_k$ through the last half.

Because of the iterative technique, the method easily lends itself to nonlinear analysis. However, for linear analysis, a direct solution is possible for deflections at the new time station. Using the β method (Eq 3.7), Chan et al have developed a recurrence relation which eliminates both velocities and accelerations from the equations of motion (Ref 4).

Wilson and Clough presented direct methods for step-by-step vibration analysis which are based on the variation of acceleration during the time-step interval (Ref 27). Methods are presented for constant, linear, and parabolic variations. The step-by-step procedure developed in this report was based on work by these investigators.

Linear Acceleration Algorithm for Step-by-Step Analysis

The basis for the analysis presented herein is the assumption of a linear variation of the acceleration between time steps. As shown in Fig 5, the linear acceleration approach has several appealing properties. First, continuous values of acceleration, velocity, and deflection are obtained. Furthermore, it is the lowest order approximation of acceleration which satisfies these conditions.

The acceleration at the end of the interval, from $k-1$ to k, is equal to the initial acceleration plus a constant v times the time-step increment:

$$\ddot{w}_k = \ddot{w}_{k-1} + h_t v \tag{3.9}$$

Expressions for velocity and deflection are found by integrating Eq 3.9 and eliminating the constant v :

$$\dot{w}_k = \dot{w}_{k-1} + \frac{h_t}{2} (\ddot{w}_{k-1} + \ddot{w}_k) \tag{3.10a}$$

$$w_k = w_{k-1} + h_t \dot{w}_{k-1} + \frac{h_t^2}{3} \ddot{w}_{k-1} + \frac{h_t^2}{6} \ddot{w}_k \tag{3.10b}$$

These relations may then be substituted into the equations of motion (Eq 2.9) for the derivation of a recursive relation for accelerations at time k :

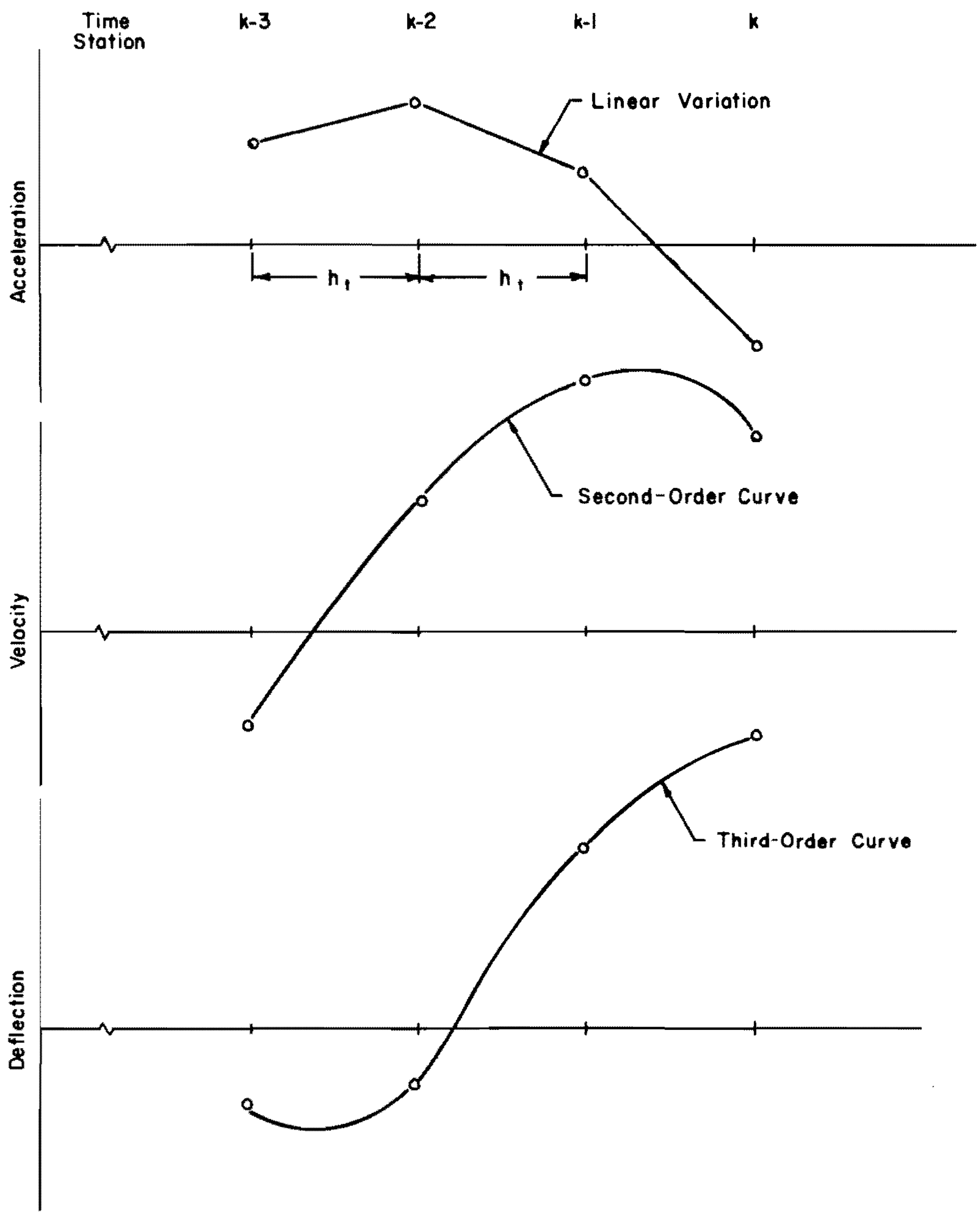


Fig 5. Node-point response for linear acceleration algorithm (after Wilson and Clough, Ref 27).

$$\left[\left[M\right] + \frac{h_t}{2}\left[DF\right] + \frac{h_t^2}{6}\left[K\right] \right] \left\{\ddot{w}_k\right\} = \left\{Q_t\right\}$$

$$- \left[\frac{h_t}{2}\left[DF\right] + \frac{h_t^2}{3}\left[K\right] \right] \left\{\ddot{w}_{k-1}\right\}$$

$$- \left[\left[DF\right] + h_t\left[K\right] \right] \left\{\dot{w}_{k-1}\right\} - \left[K\right]\left\{w_{k-1}\right\} \tag{3.11}$$

The nodal accelerations, from Eq 3.11, are then used to compute velocities and deflections (Eqs 3.10).

It is possible to eliminate both acceleration and velocity terms from Eq 3.11 by combining dynamic equilibrium equations at times $k+1$, k, and $k-1$. The recursive relation, found to be a specialized form of work presented by Chan et al (Ref 4), will include only the displacement at the three time stations:

$$\left[\frac{6}{h_t^2}\left[M\right] + \frac{3}{h_t}\left[DF\right] + \left[K\right] \right] \left\{w_{k+1}\right\} = \left\{Q_{k+1} + 4Q_k + Q_{k-1}\right\}$$

$$- \left[-\frac{3}{h_t^2}\left[M\right] + \left[K\right] \right] \left\{4w_k\right\}$$

$$- \left[\frac{6}{h_t^2}\left[M\right] - \frac{3}{h_t}\left[DF\right] + \left[K\right] \right] \left\{w_{k-1}\right\} \tag{3.12}$$

The derivation of Eq 3.12 is presented in detail in Appendix C.

Comparing Eqs 3.11 and 3.12, it may be seen that Eq 3.12 requires more information for each time step, i.e., loading at the three time steps as well as the two previous deflections. Although Eq 3.11 may be evaluated by knowing the load at the end of the time interval in question, and the acceleration, velocity, and deflection at the start of the interval, two additional calculations are required after $\ddot{w}_k$ is determined. Both the velocity and deflection must be computed for time k before the acceleration at k+1 can be determined. Because of those extra computations, the computer time required to propagate an analysis a given number of time steps would be greater for Eq 3.11. Equation 3.12 is therefore used in the analysis procedure.

Interpretation of Linear Acceleration Algorithm

The behavior of a node point for the assumption of linear acceleration is shown in Fig 5. This response imposes certain load conditions on the structure. First, the inertia force $M\ddot{w}$ is seen to vary linearly between time k-1 and k . Damping, if present, will vary as a second-order curve and the elastic restoring force as a third-order curve. For the equations of motion to be satisfied at all points within the time interval, forces with third-order variation must be applied at all node points.

If dynamic loads are placed at all nodes of the structure, it would not seem unreasonable that they vary as a third-order curve during the interval h_t . However, when one investigates unloaded node points, a condition which may create errors is discovered. To bring the problem into focus, consider a structure in free vibration with no damping. At discrete points in time, k , k+1 , ... , dynamic equilibrium is satisfied and the applied load required is zero. For equilibrium at any instant during the interval k to k+1 , a load is required which is the difference between the inertia force, which is linear, and the restoring force, which has a third-order variation. If this difference is large, serious errors would be introduced. To limit the load error, it is necessary to select a small time increment for propagation of the solution. Furthermore, it is shown in the next chapter that a small value for h_t is required for stability of the numerical procedure.

CHAPTER 4. STABILITY ANALYSIS

There are two reasons for the use of step-by-step procedures for vibration analysis. First and foremost, the method lends itself to nonlinear analysis. A second but less significant reason would be to include the influence of the higher modes of vibration on the total response of the structure. In the preceding chapter it was noted that the accuracy of the method may be seriously influenced by the selection of a time-step increment which is too large. Furthermore, it will be shown in this chapter that small time steps are often required to insure stability of the numerical method. A rational approach based on the stability analysis is proposed for the selection of the time-step magnitude.

Stability of Numerical Solutions for Initial-Value Problems

The problem of stability does not appear in numerical solutions to boundary-value problems since the selection of the increment size does not cause unstable solutions. On the other hand, the stability of numerical solutions to initial-value problems is related directly to the time-step increment. Small time-step increments are required for stable solutions to many initial-value problems. A large time-step increment may cause serious oscillations to appear after a few time steps. Unbounded oscillations are characteristic of an unstable time-step increment and are related to the mode shapes associated with the highest natural frequencies of the model. The stable time-step increment, it will be shown, is a function of x and y-increment size as well as the stiffness and mass properties of the discrete-element model.

Determination of the stability of a numerical procedure is based on the investigation of the propagation of errors introduced at any time step. If, after a large number of time steps, the errors are unbounded, the solution is said to be unstable. However, it has been shown that numerical solutions which are unstable for one time increment are stable for a smaller value (Ref 18).

The basis for stability analysis of a step-by-step method is to solve the equations of motion for the discrete-element model. It is generally possible to assume a solution which is a product of two functions, one dependent only on the time variable, the other dependent on the space variables. For many problems, the time function will be exponential. If this is the case, the exponential must decay as time increases for the numerical method to be stable.

While it is not practical to study the stability of the more complicated structural configurations of plates on foundations, insight into the stability of the numerical procedure can be gained by studying certain simple cases. In this chapter the stability of the linear acceleration algorithm is investigated for the simply supported plate with and without elastic foundation support.

Stability Analysis of Linear Acceleration Algorithm

The stability of the numerical procedure (Eq 3.12) can be studied by assuming a function of the form (Ref 5)

$$w_{i,j,k} = e^{\varphi k} W(i,j) \tag{4.1}$$

The first two subscripts of w represent space coordinates while the last one is the time step. To simplify the analysis of the numerical procedure, a uniform isotropic plate without damping is investigated.

Shown in Fig 6 is a graphical representation of the equation for one node resulting from the substitution of Eqs A.19 through A.31 into Eq 3.12.

If equal increments are taken in both the x and y-directions, the equations given in Fig 6 can be simplified by

$$h = h_x = h_y$$

The equation for free vibration of any i,j node for a rectangular plate becomes

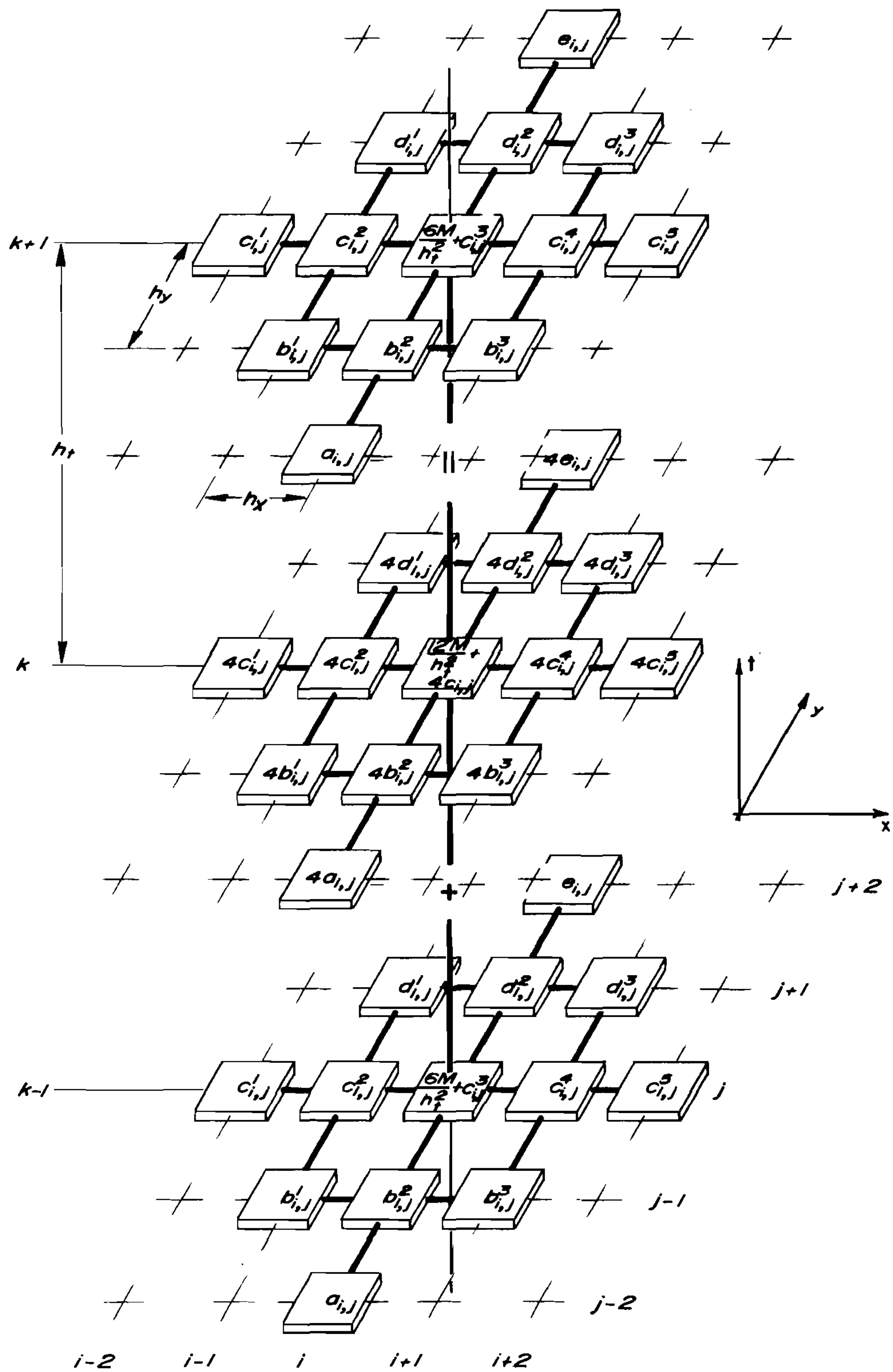


Fig 6. Graphical representation of the linear acceleration algorithm for free vibration.

$$\frac{6M}{h_t^2}\left(w_{i,j,k-1} - 2w_{i,j,k} + w_{i,j,k+1}\right)$$

$$+ \frac{D}{h^2}\left\{\left[w_{i-2,j,k-1} + w_{i+2,j,k-1} + w_{i,j-2,k-1}\right.\right.$$

$$+ w_{i,j+2,k-1} - 8\left(w_{i-1,j,k-1} + w_{i+1,j,k-1}\right.$$

$$\left.+ w_{i,j-1,k-1} + w_{i,j+1,k-1}\right) + 2\left(w_{i-1,j+1,k-1}\right.$$

$$\left.+ w_{i+1,j+1,k-1} + w_{i-1,j-1,k-1} + w_{i+1,j-1,k-1}\right)$$

$$\left.+ 20w_{i,j,k-1}\right] + 4\left[w_{i-2,j,k} + w_{i+2,j,k} + w_{i,j-2,k}\right.$$

$$+ w_{i,j+2,k} - 8\left(w_{i-1,j,k} + w_{i+1,j,k} + w_{i,j-1,k}\right.$$

$$\left.+ w_{i,j+1,k}\right) + 2\left(w_{i-1,j+1,k} + w_{i+1,j+1,k}\right.$$

$$\left.\left.+ w_{i-1,j-1,k} + w_{i+1,j-1,k}\right) + 20w_{i,j,k}\right] + \left[w_{i-2,j,k+1}\right.$$

$$+ w_{i+2,j,k+1} + w_{i,j-2,k+1} + w_{i,j+2,k+1}$$

$$-8\left(w_{i-1,j,k+1} + w_{i+1,j,k+1} + w_{i,j-1,k+1} + w_{i,j+1,k+1}\right)$$

$$+ 2\left(w_{i-1,j+1,k+1} + w_{i+1,j+1,k+1} + w_{i-1,j-1,k+1}\right.$$

$$\left.\left.\left.+ w_{i+1,j-1,k+1}\right) + 20w_{i,j,k+1}\right]\right\} = 0 \qquad (4.2)$$

In the preceding equation the foundation resistance is not included.

Dividing by $\frac{6M}{h_t^2}$, a term r can be defined:

$$r = \frac{Dh_t^2}{6Mh^2} \tag{4.3}$$

It will be shown that the value of r must be restricted to a small number for the solution to be stable.

Substitution of Eq 4.1 into Eq 4.2 gives

$$e^{2\Phi} + e^{\Phi}\left[\frac{-2W(i,j) + 4r\chi}{W(i,j) + r\chi}\right] + 1 = 0 \tag{4.4}$$

The function $W(i,j)$ will be of the form

$$W(i,j) = A \sin(\alpha_m i) \sin(\beta_n j) \tag{4.5}$$

where

A = a bounded constant,

α_m = a value dependent on the boundary conditions $i = 0$ and $i = M$,

β_n = a value dependent on the boundary conditions $j = 0$ and $j = N$.

For the simply supported plate, both zero moment and zero deflection are satisfied along the boundary if

$$\alpha_m = \frac{m\pi}{M}$$

and

$$\beta_n = \frac{n\pi}{N}$$

where M and N are the number of increments, respectively, in the i and j-directions.

The term χ in Eq 4.4 is found to be

$$\chi = A \sin i\alpha_m \sin j\beta_n (20 + 2 \cos 2\alpha_m + 2 \cos 2\beta_n$$

$$- 16 \cos \alpha_m - 16 \cos \beta_n + 8 \cos \alpha_m \cos \beta_n) \qquad (4.6)$$

Before substituting χ into Eq 4.4, it is useful to determine its maximum value. Since m can take on values from 1 to M - 1 and n from 1 to N - 1 , the sum of the terms in parenthesis will vary from 0 for m = n = 1 to a maximum which approaches 64 when m = M - 1 and n = N - 1 . It is important at this point to note that the maximum value corresponds to the highest mode of vibration for the discrete-element plate:

$$W(i,j) = A \sin\left(\frac{(M-1)\pi}{M} i\right) \sin\left(\frac{(N-1)\pi}{N} j\right)$$

The lowest value, on the other hand, corresponds to the fundamental mode of vibration:

$$W(i,j) = A \sin\left(\frac{\pi}{M} i\right) \sin\left(\frac{\pi}{N} j\right)$$

For the fundamental mode shape, Eq 4.4 reduces to

$$e^{2\varphi} - 2e^{\varphi} + 1 = 0 \qquad (4.7)$$

Solving for e^{φ} gives

$$e^{\varphi_1} = e^{\varphi_2} = -1$$

For this condition, the exponential $e^{\varphi k}$ oscillates but is bounded as k increases. However, for the highest mode of vibration, the exponential must satisfy the following relation:

$$e^{2\varphi} + e^{\varphi}\left[\frac{-2 + 256r}{1 + 64r}\right] + 1 = 0 \tag{4.8}$$

Defining the coefficient of the middle term as G the values of e^{φ} are found to be

$$e^{\varphi_{1,2}} = -\frac{G}{2} \pm \sqrt{\left(\frac{G}{2}\right)^2 - 1}$$

In order for Eq 4.1 to have a bounded value as k grows large, the following condition must be satisfied:

$$-1 < e^{\varphi_{1,2}} < 1 \tag{4.9}$$

This condition can be satisfied by

$$-1 < \frac{G}{2} < 1 \tag{4.10}$$

Consider first the lower bound

$$-2 - 128r < -2 + 256r$$

or

$$0 < 384r$$

Since r is a positive number, this condition is always satisfied. For the upper bound

$$-2 + 256r < 2 + 128\ r$$

or

$$r < \frac{1}{32}$$

Substituting Eq 4.3 for r the maximum value for h_t is found to be

$$h_t < h\sqrt{\frac{3M}{16D}} \tag{4.11}$$

For a plate on foundation, the exponential is determined by a method similar to that given above:

$$e^{2\varphi} + e^{\varphi}\left[\frac{-2 + 256r + \dfrac{4Sh_t^2}{6M}}{1 + 64r + \dfrac{Sh_t^2}{6M}}\right] + 1 = 0 \tag{4.12}$$

Again, the coefficient of the middle term must satisfy Eq 4.10. As the lower bound is satisfied by positive values for r and S, the upper bound will be investigated:

$$-2 + 256r + \frac{4SH^2h_t^2}{6M} < 2 + 128r + \frac{2Sh^2h_t^2}{6M}$$

The preceding inequality can be simplified and the limiting value for h_t determined:

$$h_t < h\sqrt{\frac{12M}{64D + Sh^2}} \tag{4.13}$$

It is seen that Eq 4.13 will reduce to Eq 4.11 when $S = 0$. Furthermore, when Sh^2 is large compared with $64D$, the time increment h_t must be smaller than that given by Eq 4.11.

The stability of the numerical procedure has been investigated for a simply supported rectangular plate with and without elastic support. The criterion for stability was that an exponential $e^{\varphi k}$ be bounded as the time coordinate k increased without bound. The value for e^{φ} was found to be related to the highest mode of vibration and, therefore, the smallest period of vibration.

Selection of Time Step for Numerical Integration

The selection of the time increment must be based on the smallest period of vibration of the discrete-element model. It is not within the scope of this work to present an exact method for predicting the highest frequency. On the other hand, it is possible to obtain a reasonable estimate for this value by a simple interpretation of the deflected shape of the plate in the highest mode.

Consider the plate of Fig 7, fixed at all points but i,j . Giving a unit deflection to this point, a restoring force, given by Eq A.25, is developed. For the isotropic plate the force is

$$F_R = \frac{20D}{h^2} + S$$

If released, the node point would vibrate with a frequency

$$\omega = \sqrt{\frac{20D + Sh^2}{Mh^2}} \qquad (4.14)$$

Equation 4.14 is an estimate of the highest frequency of the discrete-element slab. An estimate of the smallest period of vibration is therefore

$$T_{est} = 2\pi h \sqrt{\frac{M}{20D + Sh^2}} \qquad (4.15)$$

The stability criterion is compared with the estimated minimum period by dividing Eq 4.13 by Eq 4.15:

$$\frac{h_t}{T_{est}} < \frac{1}{\pi}\sqrt{\frac{60D + 3Sh^2}{64D + Sh^2}} \qquad (4.16)$$

When

$$Sh^2 < D$$

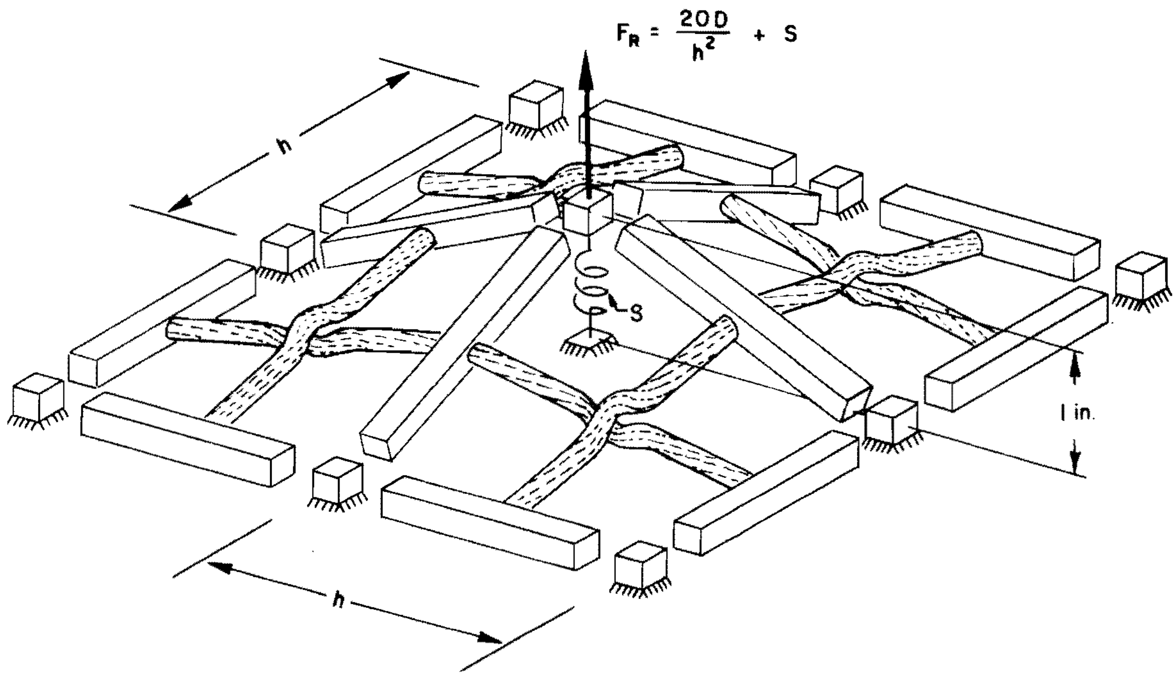


Fig 7. Method for predicting highest frequency of free vibration.

a satisfactory estimate for the maximum time increment will be

$$h_t \le \frac{1}{4} T_{est} \tag{4.17}$$

Summary

The preceding analysis has focused on the simply supported plate, both with and without foundation support. For other structural configurations, the lowest period of the discrete-element model may differ considerably from that given by Eq 4.15. Slabs on foundations, for example, will exhibit a minimum period which is larger than that given by Eq 4.15. If the edges are unrestrained, the slab becomes more flexible than that considered in the preceding analysis, thus increasing the lowest period. If the initial stiffness of the bilinear foundation is used in the analysis, separation of the slab from the foundation will further increase the smallest period. It is clear, therefore, that a time increment selected by Eq 4.17 will be adequate to ensure stability of the numerical procedure. Furthermore, since damping is not included in the stability analysis, its presence will also increase the stable time-step increment given by Eq 4.17.

To select a time step for a bridge structure, it is recommended that the average bending stiffness of the structure be used. A conservative estimate for h_t should result if the minimum node point mass is used in Eq 4.17.

The stability analysis has shown that stable numerical solutions to simply supported rectangular plate problems can always be obtained, providing the time increment satisfies Eq 4.9. Solution instability, if noted, may be corrected by reducing the magnitude of the time step and repeating the analysis.

CHAPTER 5. NONLINEAR ANALYSIS

Although the problems investigated in this work involve support characteristics which allow the slab to lift free of the foundation, the iterative techniques discussed in this chapter can be used for the analysis of structures with more general nonlinear material properties. Nonlinear analysis is therefore discussed with reference to the general nonlinear foundation.

In addition to the secant and tangent methods for solving structures with material nonlinearity, the load iteration technique is presented and discussed. The major difference, and advantage, of load iteration is that the deflection-coefficient matrix of the structure is not modified from one iteration to the next since corrections for nonlinear stiffness effects are made on the load side of the equations.

Foundation Characterization

The foundation is modeled by discrete and independent springs at each of the node points. This idealization, commonly referred to as the Winkler foundation, generates stiffness terms on only the main diagonal of the stiffness matrix. Either linear or nonlinear characteristics can be prescribed for computer analysis (see Chapter 8).

The nonlinear characteristics of each node-point spring are described by a curve consisting of straight line segments. The bilinear foundation studied in this work is shown in Fig 8. The force developed on the model by the foundation is plotted on the vertical axis and the model deflection on the horizontal axis. For both load and deflection, the positive sense is upward. For this characterization, resistance to deflection is developed only when node points deflect in the negative or downward direction.

The computer program has been prepared to accept any type of elastic nonlinearity, such as that shown in Fig 9. There are only two limitations on the nonlinear characterization: (1) the resistance-deflection curve must be continuous and (2) for every value of deflection there must be a unique resistance.

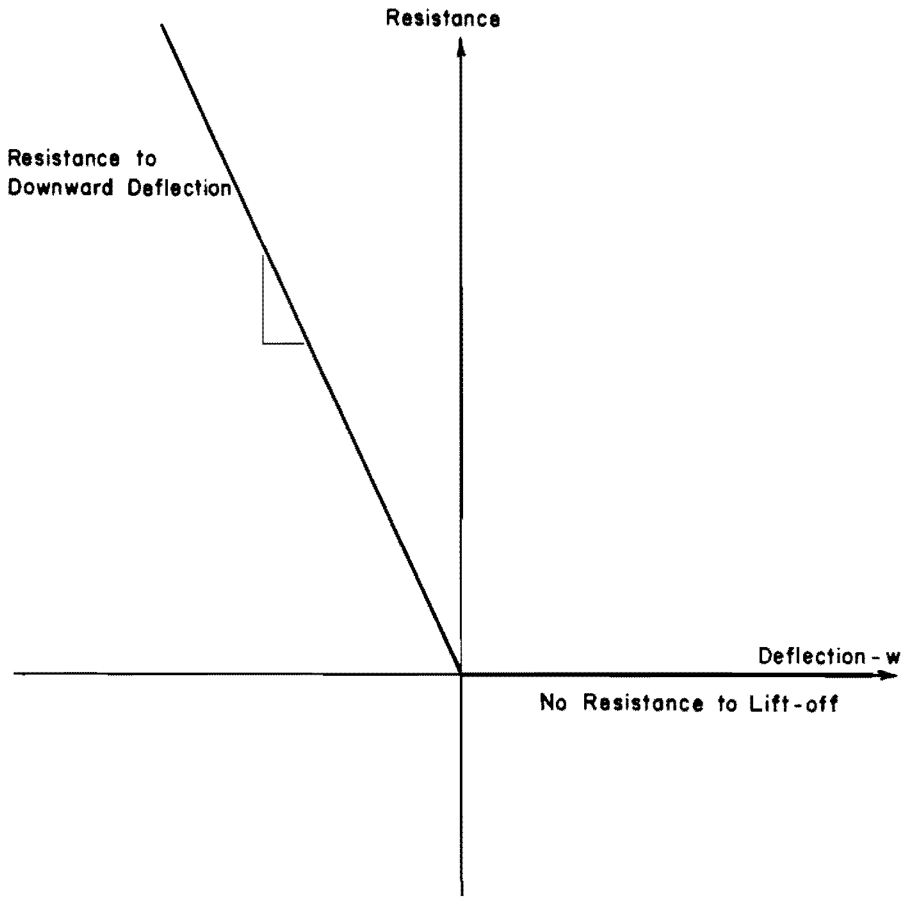


Fig 8. Bilinear foundation characteristics.

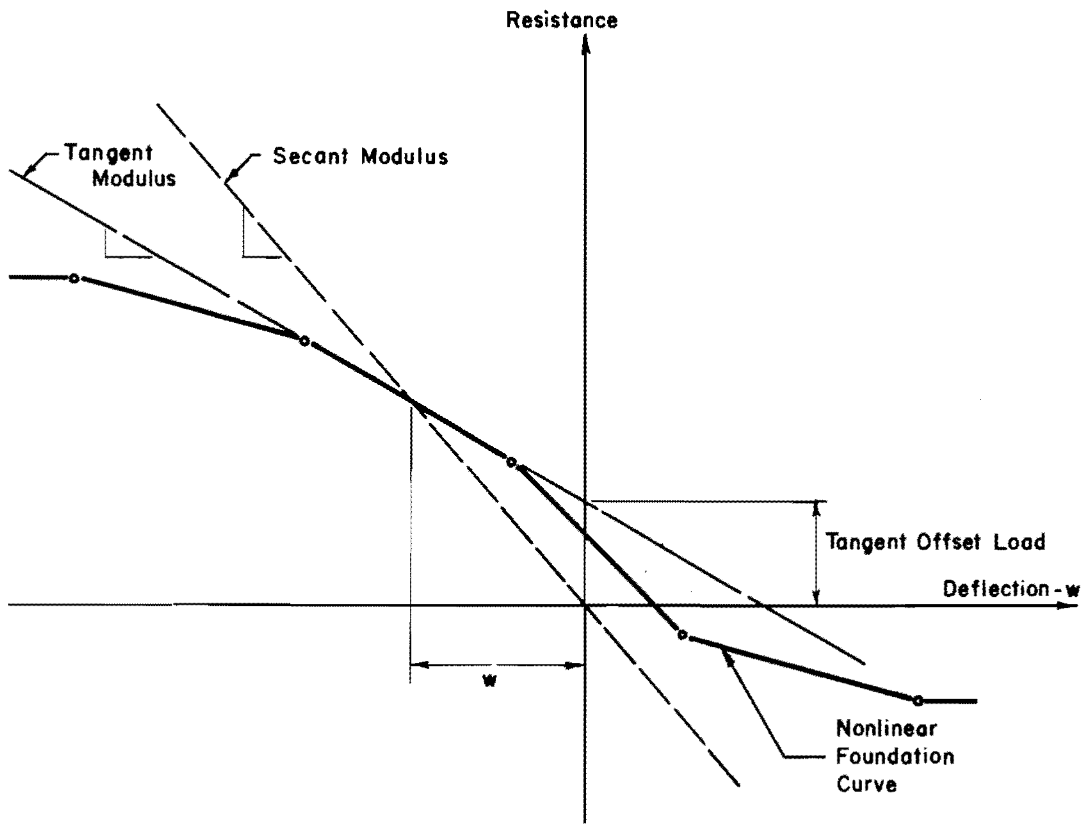


Fig 9. Representation of nonlinear foundation characteristics.

Stiffness Iteration

Nonlinear analysis can be performed by the repeated solution of modified linear equations (Ref 15). The node-point deflections are first calculated for an assumed foundation stiffness. The new deflections are then used to obtain a better estimate of stiffness. Using the new stiffness, deflections are again calculated and compared with the initial set. The iterative procedure is repeated until the deflections of two consecutive iterations agree within a specified tolerance, a condition which is called closure.

While it is often not possible to prove the convergence of stiffness iteration methods, experience has shown that solutions generally are very stable and usually converge. The procedures discussed below have been shown to be convergent for the static analysis of plates supported on soil (Ref 1). Furthermore, with the foundation properly defined, analytical solutions compare very favorably with the experimental results.

An iterative procedure which has application to a wide range of nonlinear elastic problems is the secant modulus method. By this method (shown in Fig 9) the elastic supports are adjusted from one solution to the next until closure is obtained. Although the secant modulus iteration method converges more slowly than the tangent modulus method, to be discussed next, it is very stable. Oscillations are rarely found in the iteration procedure; instead, the procedure creeps toward the equilibrium position. This method may be applied with very satisfactory results to problems with elastic, perfectly plastic material properties.

The tangent modulus method (Fig 9) has been used successfully to analyze beams on nonlinear foundations (Ref 13). This method may adjust both the stiffness and the load from one iteration to the next. The rate of convergence of the tangent modulus method is generally faster than that found for the secant approach. On the other hand, the tangent method may exhibit instability problems in cases of elastic, perfectly plastic material behavior. However, the instability is rarely noted, because the possibility of the complete plastic action for all support points is highly unlikely.

As a general rule, the tangent modulus method would be preferred to the secant approach because of the rapid rate of closure which has been noted for most problems. Studies of both beams and plates on nonlinear foundations have shown this to be true.

Load Iteration

The load iteration method (Fig 10) presents an attractive alternative to the stiffness adjustment methods because the deflection coefficients remain constant during the iteration procedure. The procedure therefore requires only a single inversion of the stiffness matrix. Repetitive solutions are found by multiplying the new load vector for each iteration by the inverted stiffness matrix. The stiffness iteration methods, on the other hand, require an inversion for each iteration.

Although the concept of the inverse of the coefficient matrix will be useful for the discussion of the load iteration method, the equations are solved by a more efficient matrix-decomposition method (see Chapter 7). For the load iteration method, only single decomposition of the coefficient matrix is required while stiffness iteration methods, on the other hand, require a complete decomposition for each iteration.

The nonlinear foundation is initially characterized by a linear spring. The deflections are computed using the linear approximation, and the prescribed resistance for that deflection is determined. The difference between the prescribed resistance and that developed by the linear spring is then added to the load term and a new deflection determined. The process is repeated until equilibrium is established.

With the nonlinear foundation represented by a linear spring, the equilibrium equations for the discrete-element model can be written

$$\left[K\right]\left\{w\right\} = \left\{Q\right\} + \left\{L(w)\right\} \tag{5.1}$$

where

$\left[K\right]$ = the linear stiffness matrix for the slab and foundation, including the linear approximation for the nonlinear curve,

$\left\{Q\right\}$ = the applied lateral load,

$\left\{L(w)\right\}$ = a deflection-dependent load function which is the difference between the nonlinear foundation curve and the linear approximation.

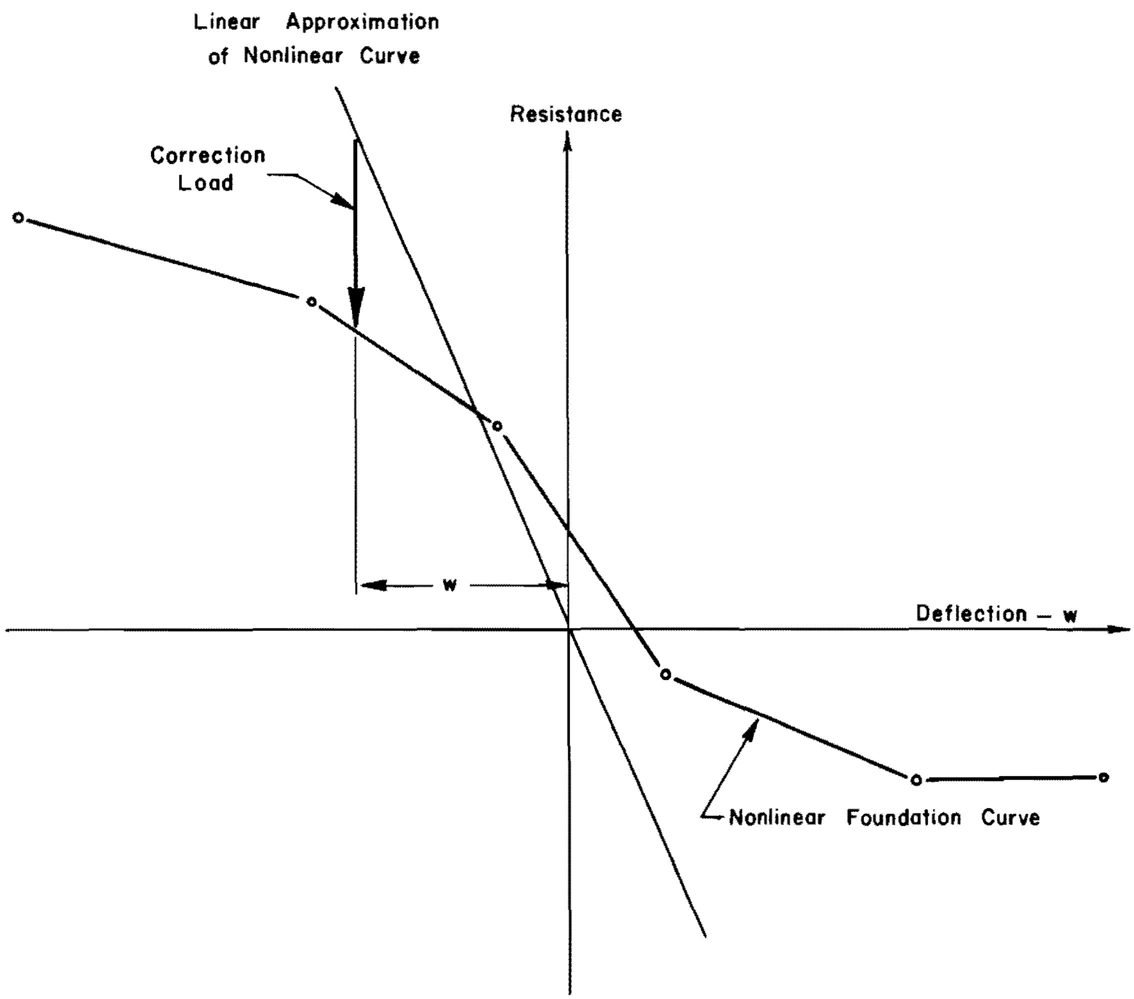


Fig 10. The load iteration method for nonlinear analysis.

The iterative procedure therefore becomes

$$\left[K\right]\left\{w_I\right\} = \left\{Q\right\} + \left\{L_{I-1}\right\} \tag{5.2}$$

or

$$\left\{w_I\right\} = \left[K\right]^{-1}\left\{Q + L_{I-1}\right\} \tag{5.3}$$

Equation 5.3 is repeatedly solved until the difference between successive solutions is less than a prescribed tolerance.

To achieve convergence, load iteration generally requires more iterations than either the secant or tangent methods. However, for many problems, convergence is reached in less computer time than with the stiffness methods. In a typical problem as many as ten load iterations can be performed in the time required for a single cycle of a stiffness iteration.

Although the stability and convergence of the load iteration method have not been rigorously proved, the method has been verified experimentally and a wide variety of problems have been solved. Beams on nonlinear foundations were studied first. The results of this investigation served as guide lines for the plate studies.

The beam studies indicated that the load iteration method would be a useful tool for nonlinear analysis. It was found that the linear approximation of the foundation should be near to the initial tangent of the resistance-deflection curve to insure stable closure. With a spring which was too soft, oscillations were noted in the closure process. A safe approach was found by always using the initial tangent, which, however, exhibited a creeping closure toward the equilibrium position.

When the method was applied for the solution of plate problems, the oscillating closure process was not as common as noted in beam solutions. This can be attributed to the greater redundancy of the plate. At any point on the beam the resistance to deflection is available from both the foundation and the beam stiffnesses. The plate, on the other hand, may be viewed as a grid, so that two crossing beams as well as the foundation offer resistance to the node-point deflection.

Closure of the solution, as noted earlier, using only load iteration may require many cycles of the solution procedure. Experience has shown that for a wide range of static problems, using alternating cycles of load iteration with a single cycle of the tangent modulus method reduces both the number of iterations and the time required for a solution. The number of cycles of load iteration before changing to a tangent modulus depends on the number of increments in the discrete-element model. However, the method demonstrated in this work (Chapter 9) focuses on solution capability by load iteration only.

CHAPTER 6. ALGORITHM FOR NONLINEAR DYNAMIC ANALYSIS

The load iteration method is coupled with the linear acceleration algorithm for numerical integration to develop an interative procedure for nonlinear analysis. Three separate steps are considered in the analysis:

(1) static solution for the initial conditions,

(2) analysis for the first time step, and

(3) the iterative procedure for the general time step.

Nonlinear Equations of Motion

The equilibrium equation for the load iteration method is given by Eq 5.1. The addition of inertia and damping forces to Eq 5.1 will yield the equation of motion for the model:

$$\left[M\right]\left\{\ddot{w}\right\} + \left[DF\right]\left\{\dot{w}\right\} + \left[K\right]\left\{w\right\} = \left\{Q\right\} + \left\{L(w)\right\} \qquad (6.1)$$

Again $\left\{L(w)\right\}$ represents the nonlinear load correction for the linearized resistance-deflection curve. Nonlinear analysis will be performed by adjusting the correction load until equilibrium or closure is satisfied.

Initial Conditions - Static Analysis

The step-by-step analysis is started with the plate at rest. Acceleration and velocity for all node points are zero while the deflection is that due to the dead load of the plate and all other sustained loads. The iteration procedure for the dead load deflection is given by Eq 5.3 or

$$\left\{w_{0_I}\right\} = \left[K\right]^{-1}\left\{Q_s + L_{0_{I-1}}\right\} \qquad (6.2)$$

When equilibrium is established, the correction loads and deflections are saved for use in the calculation of the deflection at the end of the first time increment.

First Time Step

During the program planning phase of this work, consideration was given to starting the propagation of the solution from a condition other than at rest. For example, if initial velocities and accelerations are prescribed, or need to be prescribed for a future extension of this work, the capabilities for a logical starting procedure are required. Therefore, to facilitate the modification of the program for other initial conditions, a special routine for the initial time step was included. Although the starting procedure is discussed with respect to the case of zero acceleration and velocity, which is the case for studies presented herein, it may easily be extended to include values other than zero.

The iteration procedure for the first time step is separated into two parts. First, the acceleration at the end of the time interval is calculated; it varies linearly from zero for $k = 0$ to a value $\ddot{w}_1$ at time $k = 1$. Then the velocity at $k = 1$ is computed and the deflection found by Eq 6.1. New correction loads, corresponding to the calculated deflections, are then used to obtain a new estimate of acceleration at $k = 1$. The derivation of the iterative procedure is given below.

The deflection and velocity at $k = 1$ are given by

$$\dot{w}_1 = \frac{h_t}{2} \ddot{w}_1 \tag{6.3a}$$

and

$$w_1 = w_0 + \frac{h_t^2}{6} \ddot{w}_1 \tag{6.3b}$$

The preceding equations are derived by substituting the initial conditions into Eqs C.1 and C.2 of Appendix C. The equations for dynamic equilibrium can then be written as

$$\left[M\right]\left\{\ddot{w}_{1_I}\right\} + \frac{h_t}{2}\left[DF\right]\left\{\ddot{w}_{1_I}\right\} + \frac{h_t^2}{6}\left[K\right]\left\{\ddot{w}_{1_I}\right\}$$

$$= \left\{Q_s + QD_1 + L_{1_{I-1}}\right\} - \left[K\right]\left\{w_0\right\} \qquad (6.4)$$

or

$$\left[\left[M\right] + \frac{h_t}{2}\left[DF\right] + \frac{h_t^2}{6}\left[K\right]\right]\left\{\ddot{w}_{1_I}\right\} = \left\{QD_1 - L_0 + L_{1_{I-1}}\right\} \qquad (6.5)$$

The right-hand side of Eq 6.4 is simplified by the replacement of $\left[K\right]\left\{w_0\right\}$ with Eq 5.1. From the acceleration, calculated by Eq 6.5, the velocity is determined (Eq 6.3a) and the deflections are found by

$$\left[K\right]\left\{w_{1_I}\right\} = \left\{Q_s + QD_1 + L_{1_{I-1}}\right\} - \left[M\right]\left\{\ddot{w}_{1_I}\right\} - \left[DF\right]\left\{\dot{w}_{1_I}\right\} \qquad (6.6)$$

A new estimate of the correction load $\left\{L_{1_I}\right\}$ is found and substituted into Eq 6.5. The iterative procedure is stopped when the deflections calculated at successive iterations agree within a specified tolerance.

To modify the program to include both initial velocities and accelerations, it is necessary only to replace Eqs 6.3 by the more general Eqs C.1 and C.2 of Appendix C. The logic of the starting method and the iteration procedure for the deflection at the end of the first time step would remain unchanged.

General Time Step

With the deflection and correction load known at $k = 0$ and $k = 1$, an iterative procedure for the deflection at $k = 2$, and all following time stations, can be developed, following the analysis presented in Appendix C.

Dynamic equilibrium equations are first written for times $k-1$, k, and $k+1$:

$$\left[M\right]\left\{\ddot{w}_{k-1}\right\} + \left[DF\right]\left\{\dot{w}_{k-1}\right\} + \left[K\right]\left\{w_{k-1}\right\}$$

$$= \left\{Q_s + QD_{k-1} + L_{k-1}\right\} \tag{6.7a}$$

$$\left[M\right]\left\{\ddot{w}_{k}\right\} + \left[DF\right]\left\{\dot{w}_{k}\right\} + \left[K\right]\left\{w_{k}\right\} = \left\{Q_s + QD_{k} + L_{k}\right\} \tag{6.7b}$$

$$\left[M\right]\left\{\ddot{w}_{k+1}\right\} + \left[DF\right]\left\{\dot{w}_{k+1}\right\} + \left[K\right]\left\{w_{k+1}\right\}$$

$$= \left\{Q_s + QD_{k+1} + L_{k+1}\right\} \tag{6.7c}$$

After multiplying Eq 6.7b by 4, Eqs 6.7 are added and acceleration and velocity terms replaced by Eqs C.6 and C.8:

$$\left[\frac{6}{h_t^2}\left[M\right] + \frac{3}{h_t}\left[DF\right] + \left[K\right]\right]\left\{w_{k+1}\right\} = \left\{6Q_s + QD_{k-1}\right.$$

$$\left. + 4QD_k + QD_{k+1}\right\} + \left\{L_{k-1} + 4L_k + L_{k+1}\right\}$$

$$- \left[-\frac{3}{h_t^2}\left[M\right] + \left[K\right]\right]\left\{4w_k\right\} - \left[\frac{6}{h_t^2}\left[M\right]\right.$$

$$\left. - \frac{3}{h_t}\left[DF\right] + \left[K\right]\right]\left\{w_{k-1}\right\} \tag{6.8}$$

The correction load at time k+1 is not immediately known, and iteration is required. The load iteration procedure for the general time step therefore becomes

$$\left[\frac{6}{h_t^2}\left[M\right] + \frac{3}{h_t}\left[DF\right] + \left[K\right]\right]\left\{w_{k+1_I}\right\} = \left\{L_{k+1_{I-1}}\right\} + \left\{Q'_{k+1}\right\} \tag{6.9}$$

or

$$\left\{w_{k+1_I}\right\} = \left[K'\right]^{-1} \left\{L_{k+1_{I-1}} + Q'_{k+1}\right\} \tag{6.10}$$

where

$\left[K'\right]$ = a modified stiffness matrix,

$\left\{Q'_{k+1}\right\}$ = an equivalent load vector.

During the iteration at any time step, the equivalent load vector $\left\{Q'_{k+1}\right\}$ remains unchanged; only the correction load varies from one iteration to the next. When equilibrium is established, the correction load and deflection are stored for the analysis of deflection at time $k+2$. A new equivalent load vector $\left\{Q'_{k+2}\right\}$ is computed and the iterative procedure repeated.

For the initial conditions of zero velocity and acceleration, the preceding equations could have been employed to start the dynamic analysis of the plate. If the static deflection, static load, and correction load for the static condition were substituted for terms with k and $k-1$ subscripts, the deflection w_{k+1} at the end of the first time step could have been determined by Eq 6.9. However, use of the special starting procedure insures greater flexibility of the program for future developments.

Summary

A method for the dynamic analysis of a discrete-element plate model on nonlinear foundations has been presented. Justification and verification of the method must be based on its rational development and experience with problem solving. Experience with the procedure has shown, for example, that nonlinear static problems can be solved by the load iteration method (Ref 1). Furthermore, it was noted in Chapter 5 that analytical results check favorably with experimental plate test data.

However, in the absence of experimental data for dynamically loaded plates on nonlinear foundations, it becomes necessary to justify the method by both its rational development and the demonstration of its solution capabilities. In Chapter 9 the method is applied to the free vibration of a square plate and the response of a plate to a moving load. Comparisons of computer results for these problems with existing theory will be useful for the evaluation of the method.

CHAPTER 7. SOLUTION OF EQUATIONS

To describe adequately a plate for computer analysis, it may be necessary to make a fine division of the structure, thereby generating a large number of equations to be solved. The deflection coefficient matrix is not inverted, as indicated in the preceding chapter. Instead, an efficient Gaussian elimination procedure for banded matrices is applied for the solution. Moreover, the equations need not be solved for each load vector. Multipliers generated during the first elimination procedure are stored for use with each successive load vector. The recursive process for repeated solutions has been called the multiple load method.

Organization of Equations

For each plate problem a rectangular grid work must be defined to describe the structure (see Chapter 8). The number of increments or rigid bars in the x-direction will be M and in the y-direction N . For the most efficient use of the solution procedure, $M \leq N$. The number of node points or joints therefore becomes $M + 1$ and $N + 1$ for the x and y-directions. Two boundary condition equations are required for each x and y-grid line, bringing the total number of equations to be solved to $(M + 3)(N + 3)$.

The equations generated by the model are shown in Fig 11. Presented in this manner two distinct types of banding are noted. First there is a submatrix banding. This is similar to banding noted when structures are partitioned into substructures and then formulated by the stiffness method. For any constant y-grid line j , the node behavior is influenced by deflections on grids $j-2$, $j-1$, $j+1$, and $j+2$.

Submatrix banding is shown in Fig 12. The terms in the submatrices are given in either Appendix A (static analysis) or Appendix B (dynamic analysis). Only the nonzero terms are computed and stored for the analysis procedure.

The coefficient matrix of Fig 11 is developed by writing either node equilibrium equations or equations of motion starting at node $i = 0$, $j = 0$ and ending with node $i = M + 1$, $j = N + 1$. Each horizontal partition in

GENERAL SLAB EQUATION:

$$a_{i,j}\, w_{i,j-2} + b^1_{i,j}\, w_{i-1,j-1} + b^2_{i,j}\, w_{i,j-1} + b^3_{i,j}\, w_{i+1,j-1} + c^1_{i,j}\, w_{i-2,j} + c^2_{i,j}\, w_{i-1,j} + c^3_{i,j}\, w_{i,j} + c^4_{i,j}\, w_{i+1,j} + c^5_{i,j}\, w_{i+2,j} + d^1_{i,j}\, w_{i-1,j+1} + d^2_{i,j}\, w_{i,j+1} + d^3_{i,j}\, w_{i+1,j+1} + e_{i,j}\, w_{i,j+2} = q_{i,j}$$

OR IN MATRIX FORM:

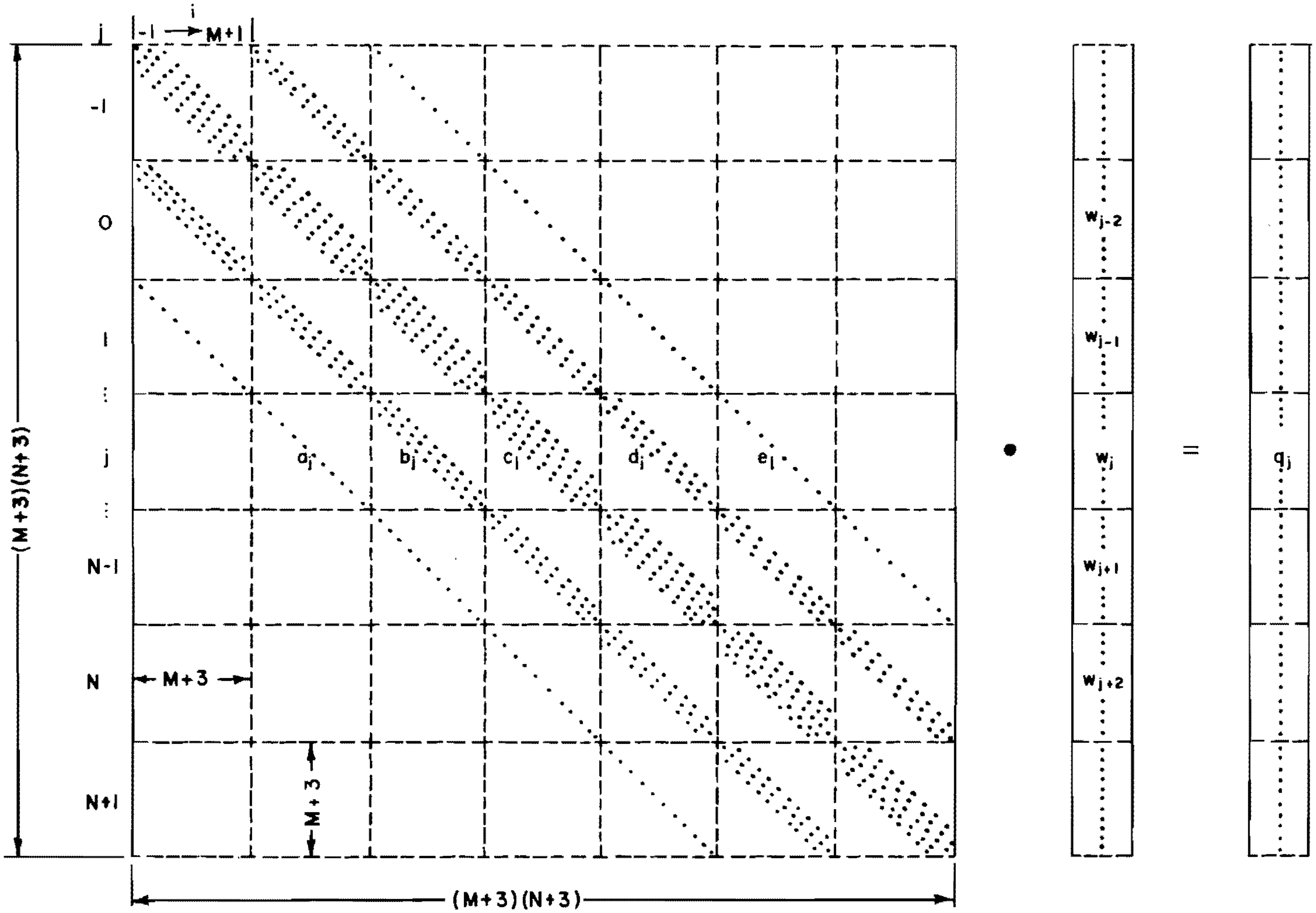


Fig 11. Banding of equations for the dynamic analysis of plates and slabs (after Stelzer, Ref 20).

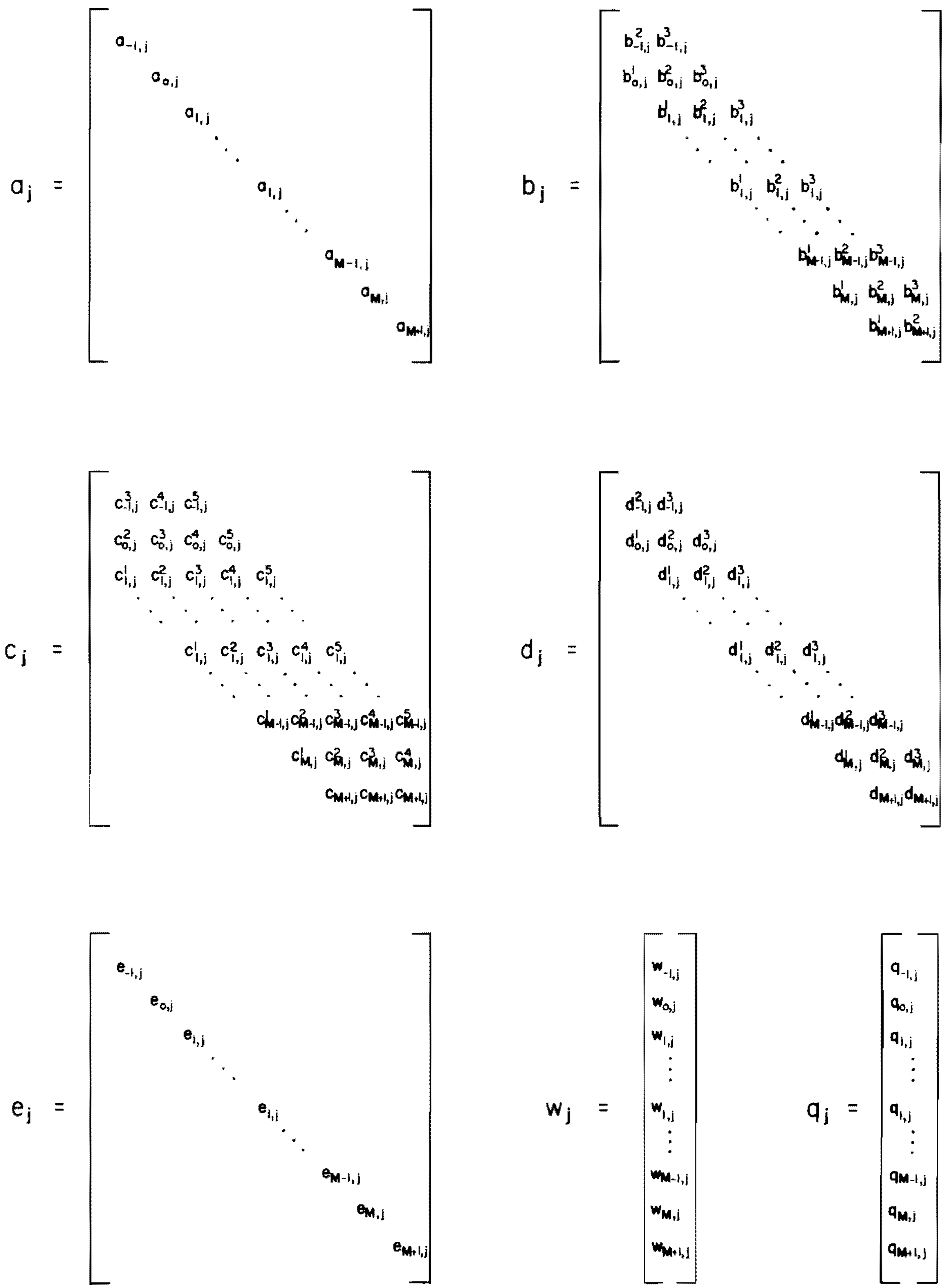


Fig 12. Submatrix banding (after Stelzer, Ref 20).

Fig 11 represents the node equations along a constant j-grid line, consecutively written from $i = 0$ to $i = M + 1$. The horizontal partitions identified as -1 and $N + 2$ contain the boundary equations for edge conditions in the y-direction. For the x-direction, the boundary equations appear as the first and last lines of the partitions 0 through $N + 1$.

Recursion-Inversion Solution Procedure

While the recursion-inversion method has been presented elsewhere (Refs 6 and 17), it is included to complete the discussion of the method for analysis. Consider the j^{th} horizontal partition of either the discretized equations of motion or the static equilibrium equations:

$$\left[a_j\right]\left\{w_{j-2}\right\} + \left[b_j\right]\left\{w_{j-1}\right\} + \left[c_j\right]\left\{w_j\right\} + \left[d_j\right]\left\{w_{j+1}\right\}$$

$$+ \left[e_j\right]\left\{w_{j+2}\right\} = \left\{q_j\right\} \tag{7.1}$$

By substituting a solution of the form

$$\left\{w_j\right\} = \left\{A_j\right\} + \left[B_j\right]\left\{w_{j+1}\right\} + \left[C_j\right]\left\{w_{j+2}\right\} \tag{7.2}$$

into Eq 7.1, it is possible to eliminate the deflections $\left\{w_{j-2}\right\}$ and $\left\{w_{j-1}\right\}$. Solving for $\left\{w_j\right\}$, the recursion matrices are determined:

$$\left\{A_j\right\} = \left[D_j\right]\left[\left[E_j\right]\left\{A_{j-1}\right\} + \left[a_j\right]\left\{A_{j-2}\right\} - \left\{q_j\right\}\right] \tag{7.3}$$

$$\left[B_j\right] = \left[D_j\right]\left[\left[E_j\right]\left[C_{j-1}\right] + \left[d_j\right]\right] \tag{7.4}$$

$$\left[C_j\right] = \left[D_j\right]\left[e_j\right] \tag{7.5}$$

The $\left[D_j\right]$ and $\left[E_j\right]$ matrices can be considered as multiplier matrices. They are found to be

$$\left[D_j\right] = -\left[\left[a_j\right]\left[C_{j-2}\right] + \left[E_j\right]\left[B_{j-1}\right] + \left[c_j\right]\right]^{-1} \tag{7.6}$$

$$\left[E_j\right] = \left[a_j\right]\left[B_{j-2}\right] + \left[b_j\right] \tag{7.7}$$

Panak (Ref 17) shows the similarity between Eqs 7.3 through 7.7 and those derived for the recursive solution of beam-columns (Ref 14). In the latter problem, constants replace the matrices.

For a symmetric stiffness matrix, a similar set of recursive matrices and multipliers can be developed (Ref 6):

$$\left\{A_j\right\} = \left[D_j\right]\left[\left[E_j\right]\left\{A_{j-1}\right\} + \left[e_{j-2}\right]^t\left\{A_{j-2}\right\} - \left\{q_j\right\}\right] \tag{7.8}$$

$$\left[B_j\right] = \left[D_j\right]\left[E_{j+1}\right]^t \tag{7.9}$$

$$\left[C_j\right] = \left[D_j\right]\left[e_j\right] \tag{7.10}$$

where

$$\left[D_j\right] = -\left[\left[e_{j-2}\right]^t\left[C_{j-2}\right] + \left[E_j\right]\left[B_{j-1}\right] + \left[c_j\right]\right]^{-1} \tag{7.11}$$

and

$$\left[E_{j+1}\right] = \left[e_{j-1}\right]^t\left[B_{j-1}\right] + \left[d_j\right]^t \tag{7.12}$$

A close inspection of Eqs 7.7 and 7.12 reveals the matrix $\left[E_{-1}\right]$ to be zero. Furthermore, since $\left[E_{-1}\right]$ is not required for the symmetric form, these calculations are omitted.

Since the equations for the discrete-element model are symmetric, Eqs 7.8 through 7.12 are used in the solution procedure.

Multiple Load Analysis

The multiple load method for analysis was first presented by Panak (Ref 17), and is reviewed to complete the discussion of the procedure.

A careful study of Eqs 7.8 through 7.12 will reveal that the load vector $\{q_j\}$ influences only the calculation of $\{A_j\}$. Since $\{A_j\}$ does not appear in either the remaining coefficient or multiplier matrices, a convenient method for solving a system of linear equations with several loading, or right-hand sides, presents itself. For the first right-hand side, the matrices $[E_j]$, $[C_j]$, and $[B_j]$ are computed and stored on disk or tape files. The $\{A_j\}$ term, however, is dependent on the unique loading condition, and is destroyed when no longer required for the solution process. For the second and all succeeding right-hand sides, the coefficient and multiplier matrices are recalled, as needed, and new $\{A_j\}$ values computed.

CHAPTER 8. COMPUTER PROGRAM

The numerical method described in this report has been coded in FORTRAN language for the Control Data Corporation (CDC) 6600 digital computer. The computer program consists of a main driver program and 27 subroutines. Although several of the subroutines could easily be incorporated into the main program, greater flexibility is achieved with the program in subroutine form. This feature will facilitate the program's extension or modification to include future developments.

Other significant features of the program include the extensive use of peripheral storage units, the method for the description of the dynamic loading, and, finally, the use of Endres' efficient recursion-inversion, multiple load technique for the solution of the linear, simultaneous equations (Ref 6).

To provide the necessary storage for problems with large numbers of increments in x and y-directions, much of the data have been placed on disk files. In addition to program data, the static, dynamic, and correction loads, as well as the structures stiffness matrix, are stored in separate files.

Program SLAB 35

Program SLAB 35 is a FORTRAN program for the CDC 6600 digital computer. This program is the thirty-fifth of a sequence for the analysis of plate structures. All of the preceding programs identified by SLAB were written for the static analysis of plates and slabs. With the exception of the READ and WRITE commands for the peripheral storage requirement, the program was coded in ASA FORTRAN.

A summary flow diagram which indicates the order in which operations are performed is presented in Fig 13. Detailed flow diagrams and listings of the main program and subroutines are given in Appendix E.

The required computer time for any problem is a function of the number of model increments and number of iterations for closure. For the example problems included in this work, 140 time steps for a linear 8 by 8 plate required 2400 seconds. For the 4 by 15 plate with moving load, 200 time steps of the

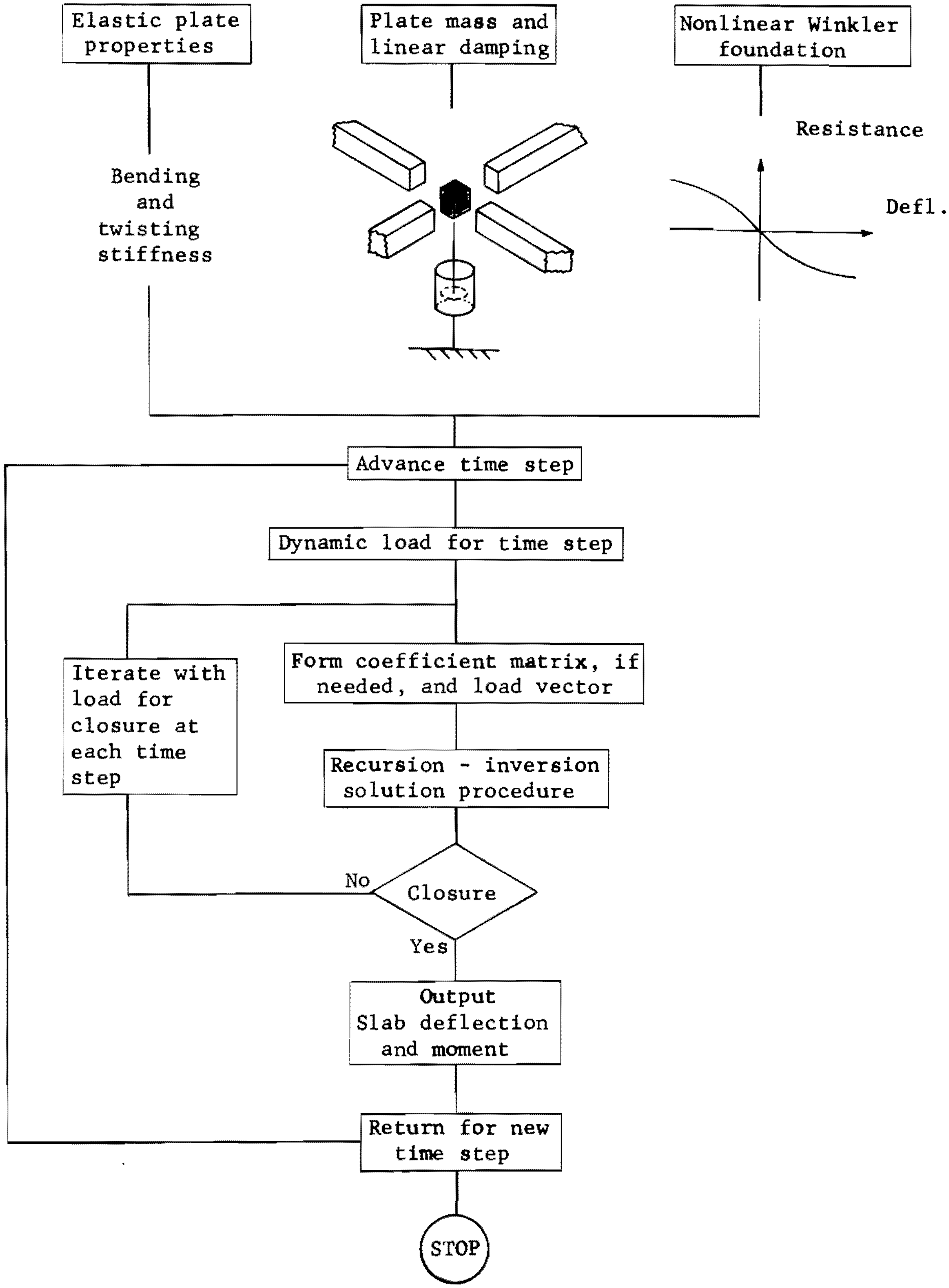


Fig 13. Main features of SLAB 35 program.

nonlinear solution would have required 16,000 seconds. The linear solution, however, required only 2200 seconds for the same number of time steps.

The storage requirements of the program are shown graphically in Fig 14. Note that a problem with equal increments in x and y-directions requires more storage than long, narrow problems containing the same number of node points. For example, a 10 by 65 grid requires approximately the same storage as a 20 by 20 slab, although there is a ratio of 1-1/2 to 1 for the node points.

Data Input

Details of the input form and supplemental instructions are included in Appendix D, which is intended as a self-contained instruction manual for SLAB 35. Furthermore, examples of the preparation of data for the program are presented as a guide for the user.

A tabular form has been developed for the data organization. Following two alphanumeric program description cards and a problem identification card, problem data are separated into seven tables:

Table 1 - Program Control Data

The information on these cards includes the number of cards and curves for the remaining tables, number of increments and increment length, monitor stations, and iteration control information.

Table 2 - Elastic Properties of the Slab

Bending stiffness and linear foundation springs are organized in this table. The number of cards varies, up to 50, depending on the problem.

Table 3 - Axial Thrust and Twisting Stiffness

The distribution of the static axial thrust must be specified by the program user. The plate twisting stiffness is also included in this table. Again, as many as 50 cards may be used to describe the variables.

Table 4 - Mass and Damping Properties

The node point mass and damping are input in Table 4, using as many as 50 cards.

Table 5 - Static or Dead Loads

Loads and moments which are not functions of time are input in Table 5. The weight of the plate will generally be input by this table. As many as 50 cards can be used to define the loading.

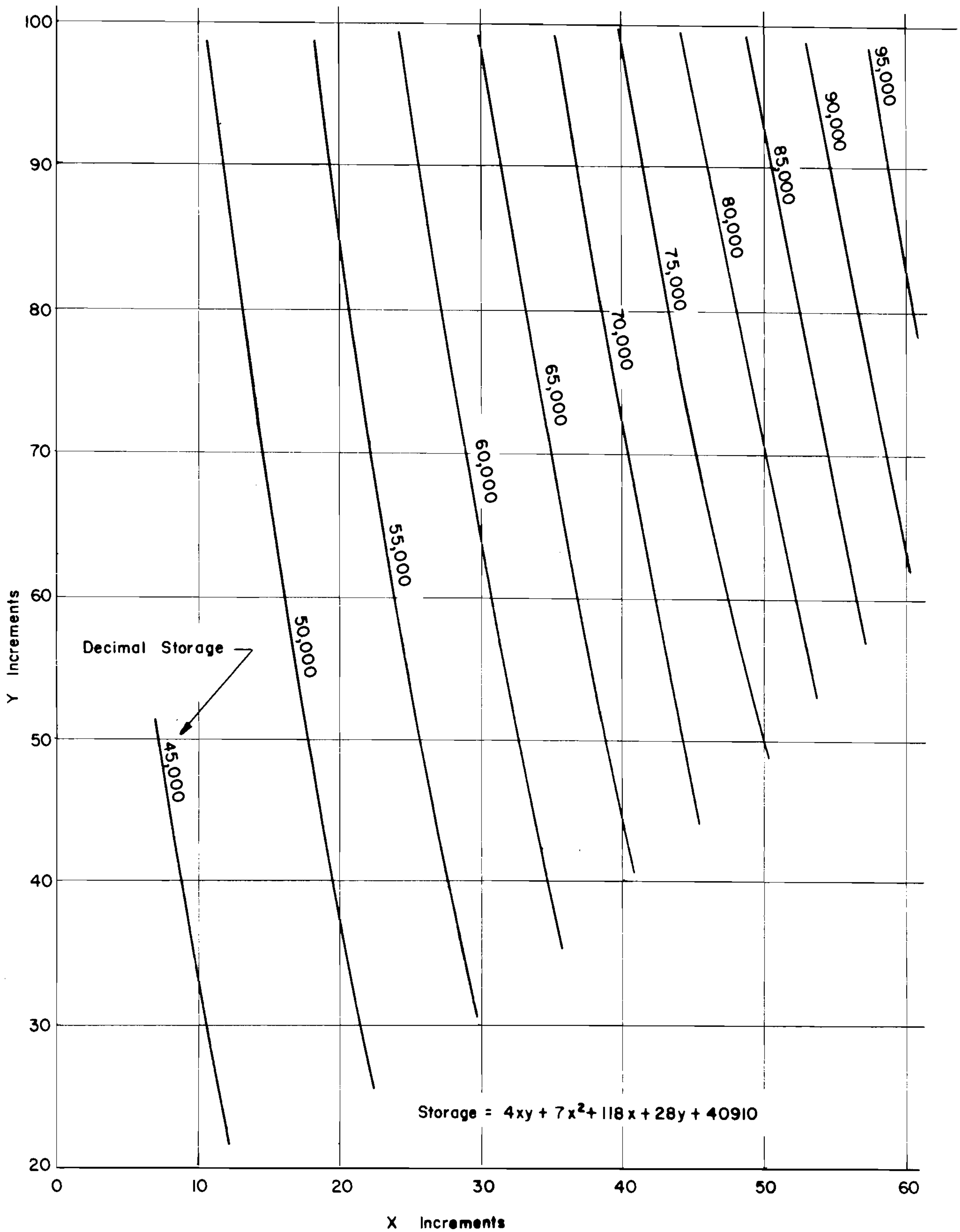


Fig 14. Storage requirements for program SLAB 35.

Table 6 - Dynamic Loading

As many as 20 load-multiplier curves, each of which can control as many as 20 loadings, are input in Table 6. A periodic multiplier is available by the use of an option switch. A moving load option permits the loads to move in either the positive or negative y-direction at a constant velocity.

Table 7 - Nonlinear Support Data

Nonlinear Winkler foundation springs can be prescribed for any area of the plate. The nonlinear curve is described by a simple tabular input which generates a curve of straight line segments.

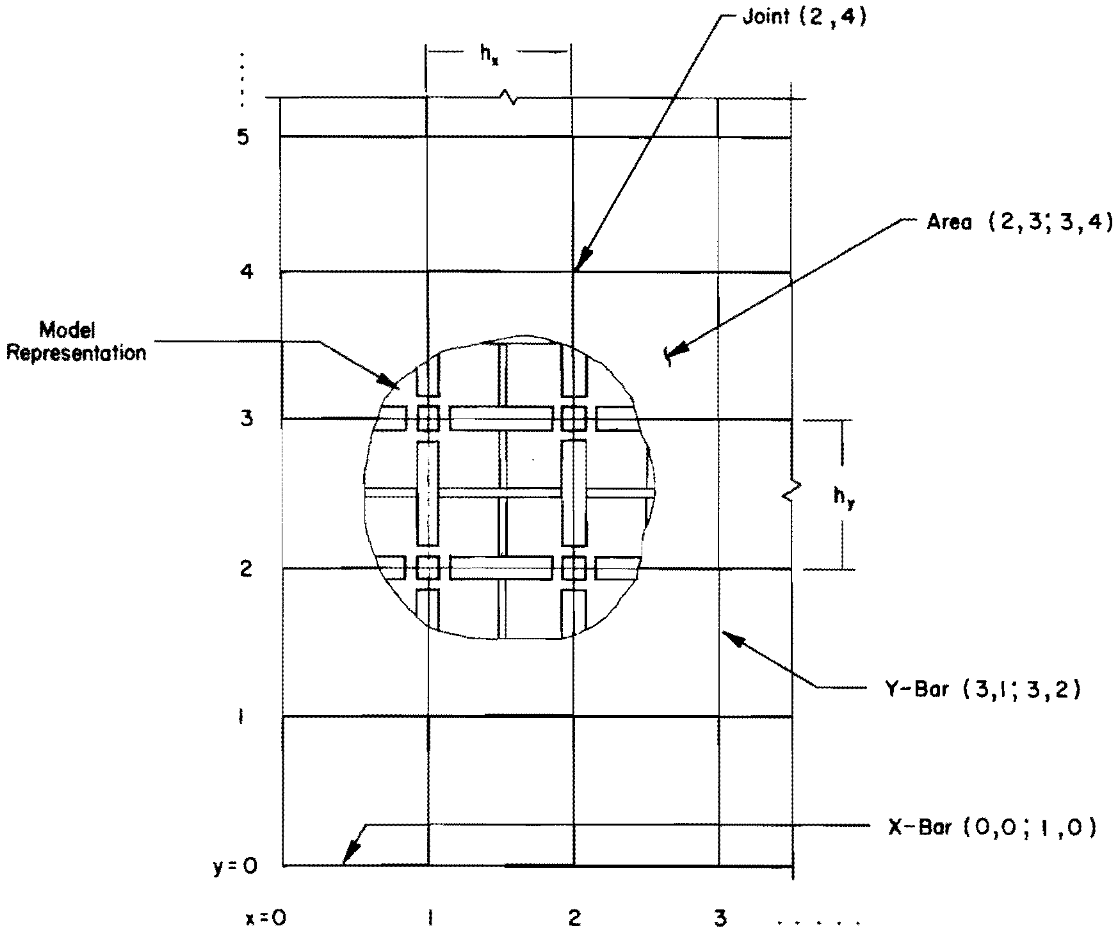
Although example input is presented in Appendix D, it will be useful to focus on the various types of data required for the description of the plate for computer analysis.

On each plate, a rectangular grid must be established. The intersections of grid lines establish node points for the model. When it is recalled that the discrete-element model consists of rigid bars and elastic joints, the grid lines are immediately recognized as bars. Furthermore, the open areas between grid lines contain model torsion bars. In Fig 15 an area of the model has been superposed on the continuum to be analyzed.

The inputs required for the description of the plate are bending stiffness, twisting stiffness, axial thrust, elastic support springs, mass, damping, and dead load. The data are logically identified by node point coordinates. With the exception of axial thrust and twisting stiffness, the variables are concentrated at nodes. Mass, damping, and linear foundation springs exist only at nodes, as do dead load and bending stiffness. A table can be compiled which contains the node point and the corresponding value of these variables. However, if these data are constant over an area of the plate, it will be convenient to specify an area by the node points and call on the computer to perform the distribution. This is, in fact, what is done.

The program accepts conventional plate stiffness properties and internally converts them to model values. The other variables, however, must be input as discrete or concentrated values. For example, the units of bending and twisting stiffness are $lb\text{-}in^2/in$, or continuum units, while those for mass are $lb\text{-}sec^2/in$, or concentrated values.

It will be convenient to describe twisting stiffness in an area between grid lines. This is logically accomplished with the use of node coordinates.



Node Data : D^x , D^y , Q , S , M , DF

(D^x and D^y are lb-in^2/in., Q and S are lb, and lb/in., respectively, M is lb-sec^2/in. and DF is lb-sec/in.)

Area Data : D^{xy}

(D^{xy} is lb-in^2/in.)

Bar Data : P^x, P^y

(P^x and P^y are lb)

Fig 15. Node coordinate identification of model properties (after Panak, Ref 17).

An area is identified by the coordinates of the lower left-hand and upper right-hand node points. For example, the twisting stiffness in the area shown on Fig 15 would be identified by 2,3; 3,4. Furthermore, it would not be appropriate to define twisting stiffness by a single node point. As noted in Appendix A, twisting stiffness does not exist in the model at nodes.

The axial thrust is in pounds rather than pounds per inch as in conventional plate theory. The distribution of the axial thrust must be prescribed by the user since the program does not perform an in-plane or axial analysis. Axial tension is given a positive sign while compression is identified by a negative sign. Axial thrust does not uniquely exist at a node point, but within a bar or bars between node points. It is therefore defined by the coordinates of two points, the first being the point of application of the load and the second the point of reaction. For example, a value P^x applied to the left edge (0,2) and reacted at 2,2 would be located on the plate by (0,2; 2,2) with the smaller x-coordinate given first. In the y-direction, the force is described in a similar manner, with the smaller y-coordinate listed first.

Area definitions are available for the description of a uniform axial thrust in several bars. For example, if a uniform axial thrust P^y is applied to the plate of Fig 15 at nodes 0,1; 1,1; and 2,1, and reacted at nodes 0,4; 1,4; and 2,4, the area description 0,1; 2,4 identifies the loaded bars.

The user has been given considerable flexibility for specification of dynamic loading. Periodic or nonperiodic as well as stationary or moving loads can be described. To define the dynamic loading for Table 6, both a load and a load amplitude multiplier are required. Since this study is intended to focus on problems with highway structures, the loads would be the static wheel loads of vehicles and the multiplier would give the variation of the wheel loads with time. An example of the development of the multiplier curve is given in Fig 16. In the example the static weight on the wheel is 5,000 pounds. The multiplier curve varies around 1.0, according to the measured dynamic loading and is constructed from straight line segments. The multiplier curve can be applied to either point, line, or area descriptions of load.

Foundation Description

Either linear or nonlinear foundation characteristics can be described. The linear foundations are input in Table 2 while the nonlinear characteristics are described in Table 7.

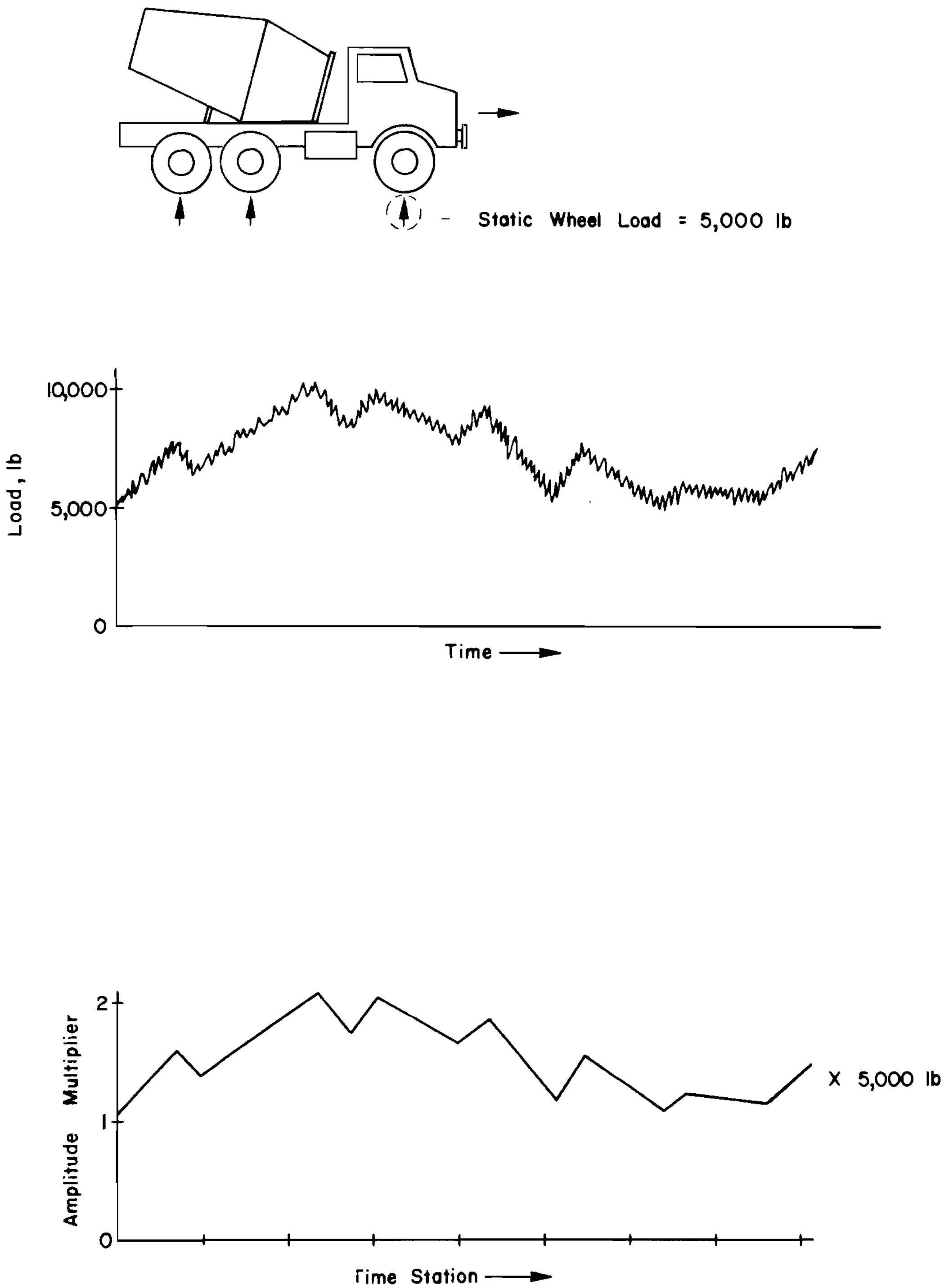


Fig 16. Load-multiplier curve for dynamic load variation.

The nonlinear resistance-deflection curve is constructed from straight line segments. Units of pounds and inches must be used for the development of curves. As resistance is developed only at node points, a single coordinate can define the location of the foundation reaction. However, both line and area descriptions, as well as concentrated curves, are available when the foundation characteristics are uniform over a line or area.

The limitations imposed on the construction of the curves were given in Chapter 5; resistance-deflection curves must be continuous, and a unique resistance must exist for any value of deflection. For deflections which exceed the prescribed end points of the curve, the resistance is determined by a straight line extrapolation of the last straight line segment of the curve. When this condition exists, a message is printed to warn the user that an off-curve condition exists.

Summary

Nodal coordinates are utilized to logically identify locations of slab, foundation, and load variables. Three types of descriptions are required: (1) node, (2) area, and (3) bar. Properties which exist at nodes are

(1) bending stiffness;
(2) elastic support, both linear and nonlinear;
(3) load;
(4) mass; and
(5) damping.

Area identification are required for the twisting stiffness while axial thrust is a bar property.

Furthermore, both discrete and continuous data are used in the program. For the convenience of the user, the bending and twisting stiffnesses of the plate are input as continuum plate values or lb-in^2/in. All other data are input as concentrated or discrete values.

A self-contained user's manual is given in Appendix D. Included in this appendix are examples of data organization and the input format.

CHAPTER 9. EXAMPLE PROBLEMS

Four types of example problems are presented to illustrate the accuracy and solution capability of the program: (1) free vibration of a simply supported square plate, (2) moving line load on a simply supported rectangular plate, (3) moving line load on a rectangular plate resting on both linear and nonlinear foundations, and (4) response of highway bridge approach slab to moving wheel loads.

Free Vibration of a Square Plate

The free vibration study was performed on a 48-inch-square plate, simply supported along its edges. The bending stiffness, uniform in both x and y-directions, was 2.5×10^6 lb-in^2/in and Poisson's ratio was 0.25. The mass density of the plate was 7.5×10^{-4} lb-sec^2/in^3. The plate was divided into an 8 by 8 grid with h_x and h_y equal to 6 inches. The time-step increment, based on Eq 4.17, was 2.0×10^{-4} second. The theoretical period for the fundamental mode of vibration was 64 time steps (Ref 19).

To develop a free vibration condition which would illustrate the fundamental frequency, a static or dead load approximating a double sine function was applied to the plate:

$$Q_{i,j} = Q \sin \frac{i\pi}{8} \sin \frac{j\pi}{8} \tag{9.1}$$

Lateral deflections were developed which approximated the fundamental mode shape. The dynamic loading was a constant force (the negative of the dead load) which canceled the dead load, causing the plate to vibrate in the first mode shape.

The results of this problem are presented in Fig 17. The deflection of the center node (4,4) is presented as a function of the time step for almost two cycles of the fundamental period, or 120 time steps.

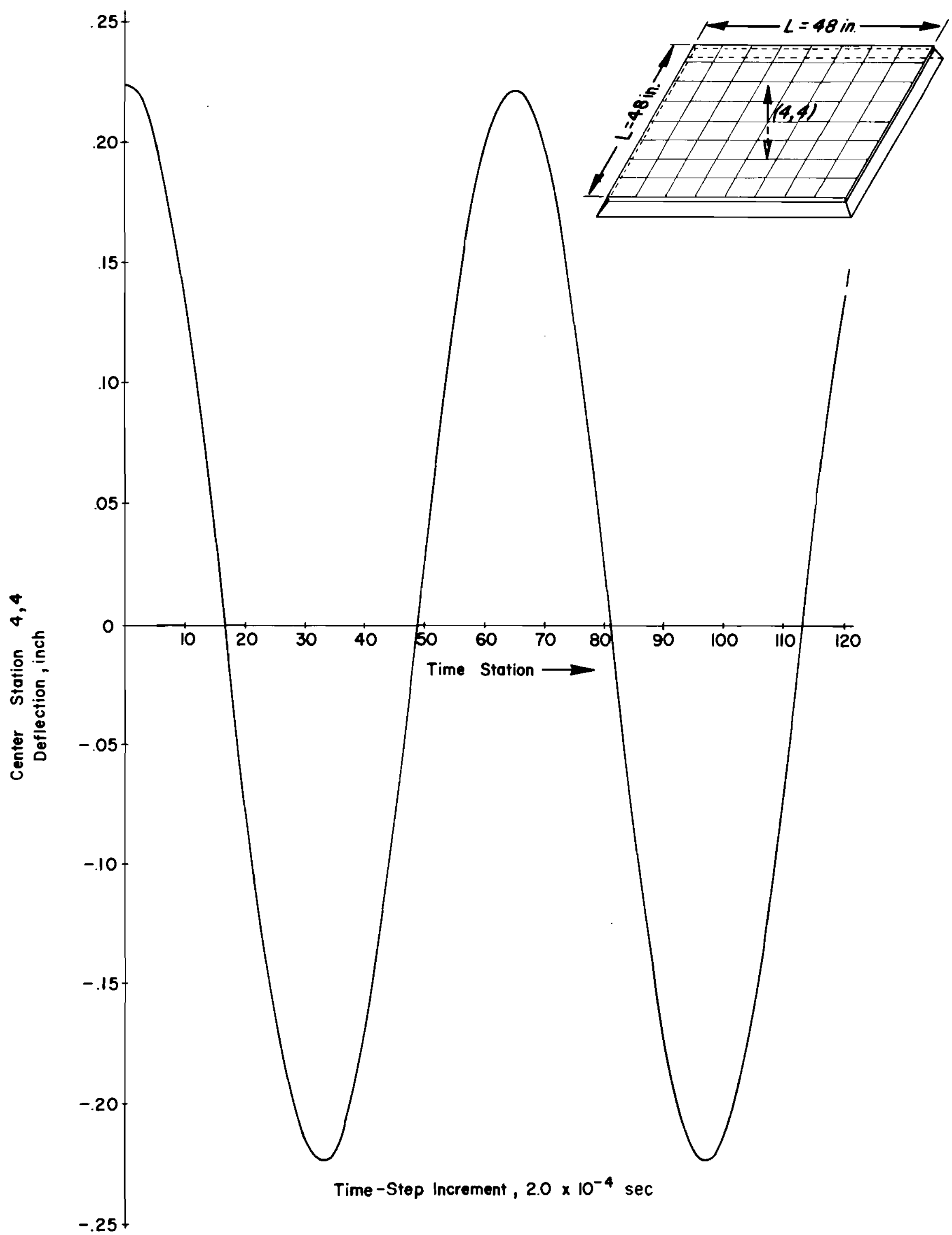


Fig 17. Free vibration of a simply supported square plate, station 4,4.

Two important features should be noted in Fig 17. First, the displacement history of node 4,4 shows no indication of instability. Second, the fundamental period for the 8 by 8 model is noted to be about 65 time steps, or one more than the theoretical, and the second cycle of deflection repeats almost exactly the first.

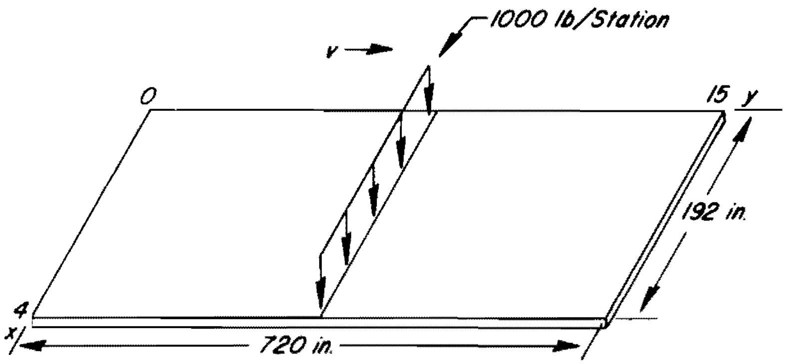
Another problem was run with a time-step increment of 4.0×10^{-4} second, or almost twice the maximum time step required for stability of the numerical solution. Instability was noted in the results before one complete cycle of free vibration.

Moving Load on a Simply Supported Rectangular Plate

The procedure was further verified by study of the traveling wave caused by moving loads. The plate was loaded by a line load in the x-direction of 1,000 lb/station. The load was moved across the plate at 53.7 mph for one problem and 214.8 mph for another, and the effect of the velocity of the moving load on the response of the plate was studied. The results are shown in Figs 18, 19, and 20.

Figure 18 shows the plate configuration, data, and center line deflection of the structure when the line load reached y-station 7. The general shape of the deflection curves compares favorably with those reported by Salani (Ref 19). As he noted, the deflected shape for the low velocity approached that of static deflection. Dynamic effects were noted by the amplification of the deflections in the center of the span and the positive deflections at the ends of the plate. This last feature indicated the traveling wave caused by the moving load preceeded the load along the plate. For the higher velocity, on the other hand, the traveling wave trailed the load.

Figures 19 and 20 show the deflection history of station 2,7 for the two load velocities. The dynamic response of the plate to the lower velocity was not as significant as to a velocity of 214.8 mph. In Fig 19 it can be noted that some vibration remained as the load moved off of the plate, but the deflection at station 2,7 was considerably smaller when the load was on that station. For the higher velocity there was little change in the maximum deflection of station 2,7 with time. With the traveling wave lagging the load, free vibration with the maximum deflection was noted after the load moved off of the plate.



(a) Simply supported rectangular plate.

4 × 15 grid

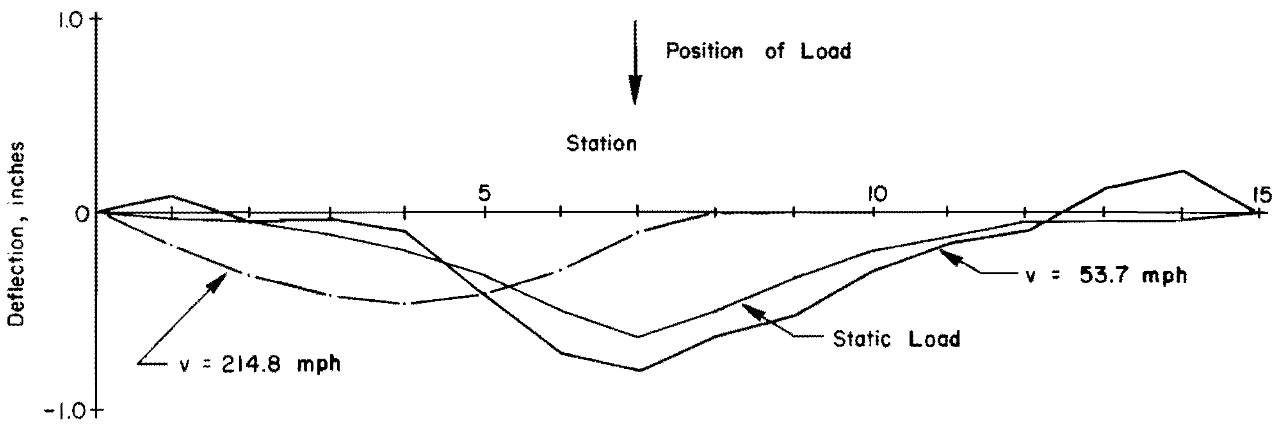
$\rho = 7.5 \times 10^{-4}$ lb-sec^2/in^3

$D_x = D_y = 2.5 \times 10^6$ lb-in^2/in

$\nu = 0.25$

$h_x = h_y = 48$ in.

$h_t = 0.01$ sec



(b) Deflection profile of longitudinal center line.

Fig 18. Moving load on simply supported rectangular plate.

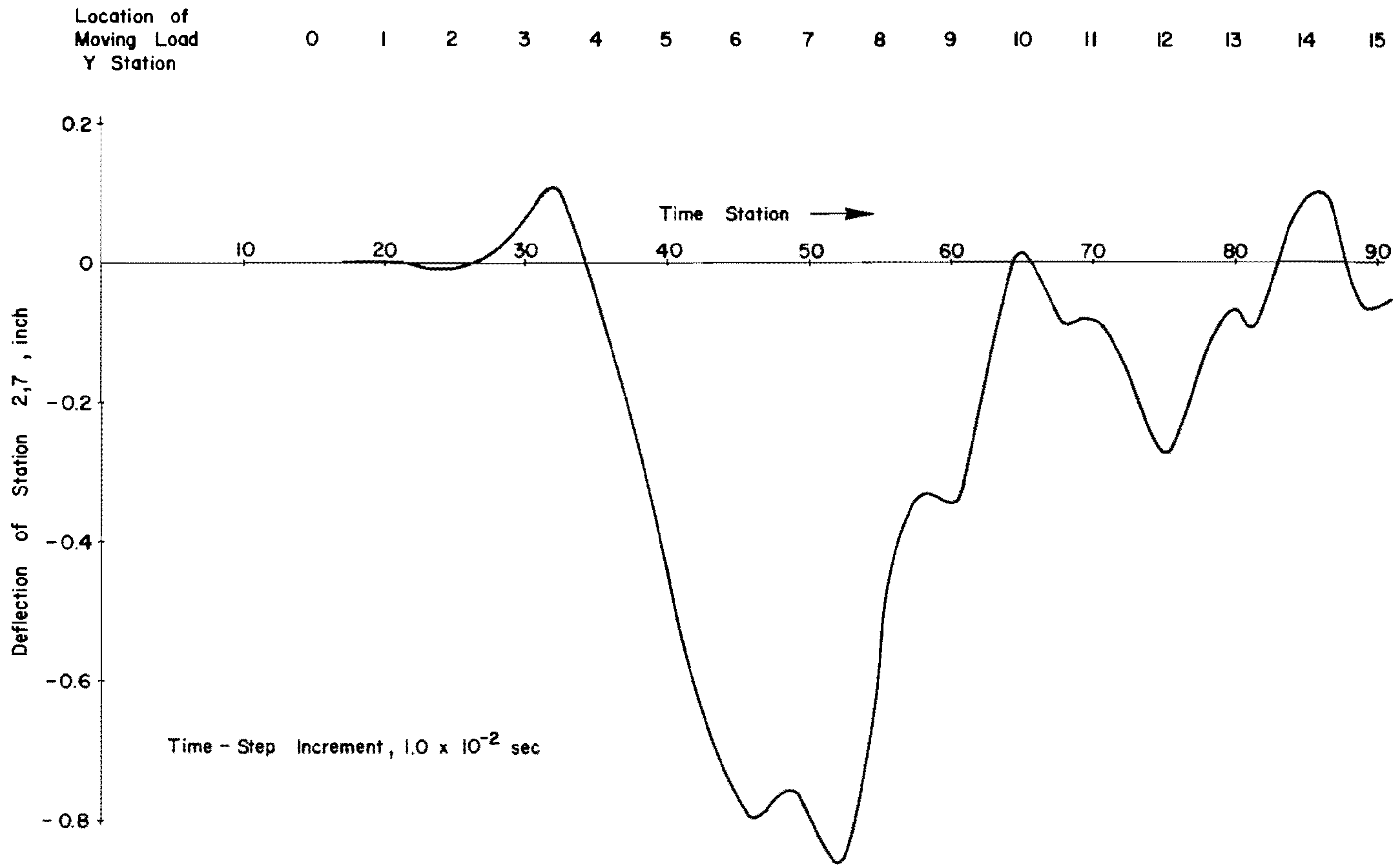


Fig 19. Response of station 2,7 to moving load, velocity = 53.7 mph.

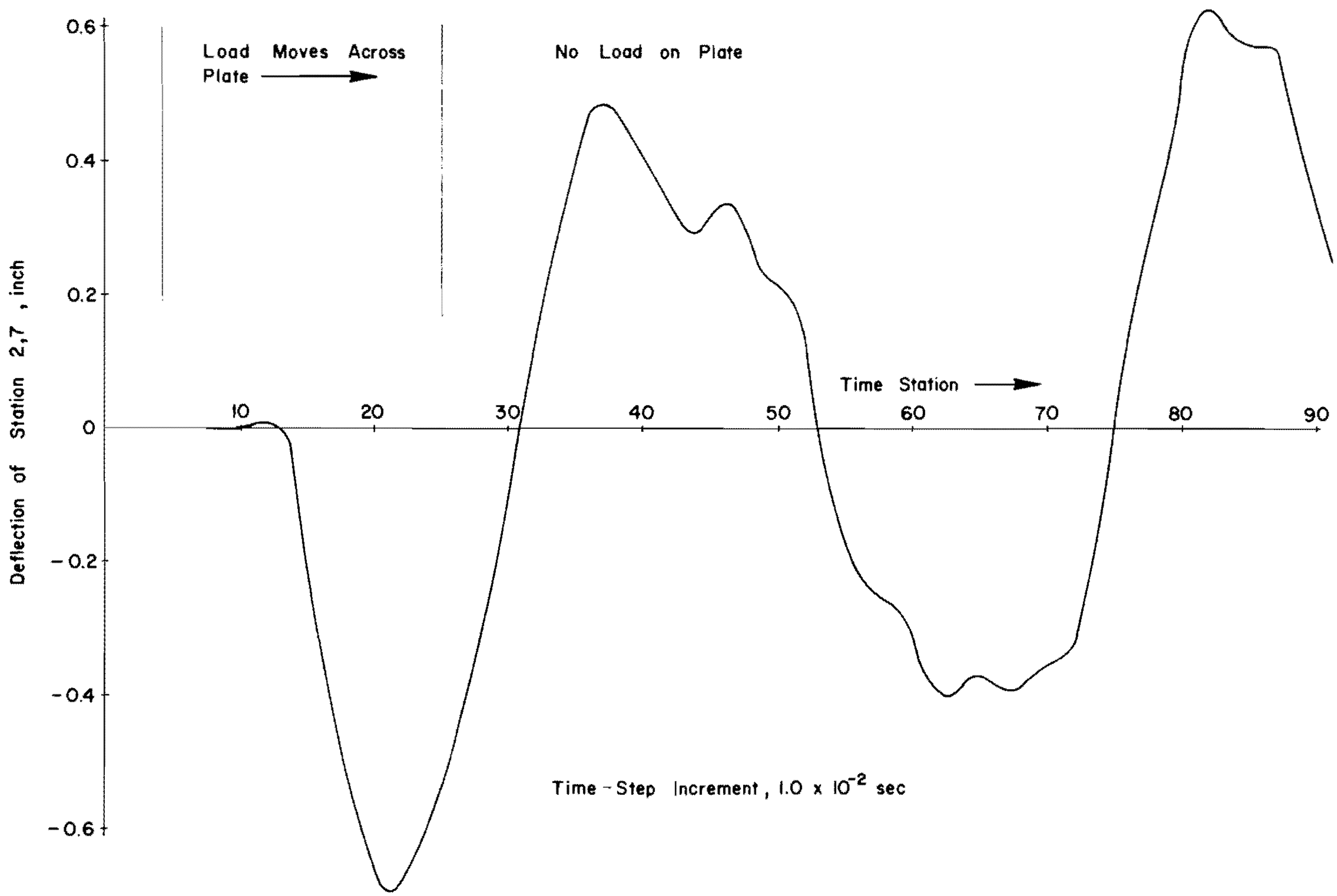


Fig 20. Response of station 2,7 to moving load, velocity = 214.8 mph.

Moving Load on a Rectangular Plate Resting on a Nonlinear Foundation

The preceding problem was modified to study the responses of plates on nonlinear foundations. Edge support along the longitudinal edges was removed and support springs were placed under each node point. Two problems were run to demonstrate the solution capability of the program, one with linear springs and a second with springs which resisted downward deflection but not lift-off or upward deflection. The loading was increased from 1,000 lb/station to 100,000 lb/station to accentuate the difference between the linear and non-linear solutions.

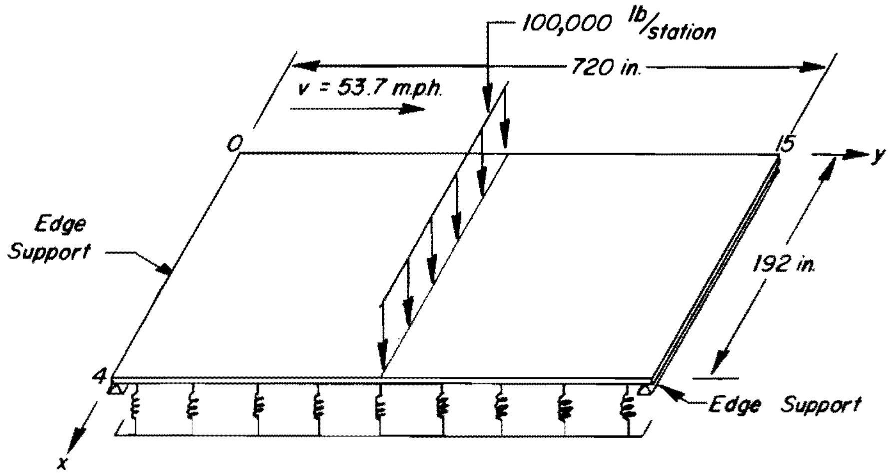
The plate configuration and data, as well as the longitudinal center-line deflection, are shown in Fig 21. Although the deflections appear to be larger than the increment length (Fig 21b) this is not the case and is due to the scale selected for deflection. The increment length is almost eight times the largest deflection shown in this figure.

In the linear problem, the plate appeared to oscillate with small deflections about the zero-deflection line. However, the small deflections at stations 1, 2, and 3 along the center line (about 0.3 inch) developed foundation forces of about 150,000 pounds at each station. For stations 1 and 3 this force acted down on the plate while at station 2 the force was upward.

As the load moved across the plate on the nonlinear foundation, the hold-down forces were not available for positive deformations, and the deflections increased until the kinetic energy of each node point was transformed to strain energy in the model which caused the large deflections for stations 1, 2, and 3.

The linear approximation for the nonlinear foundation was taken as the spring stiffness in the negative deflection range, that is, 460,800 lb/in. For station 2 (Fig 21b) the correction load at closure was approximately 2,250,000 pounds. This load was required in order to satisfy a zero foundation resistance for the positive deflection. The load error at closure for this station was an upward force of 1.295 pounds, well within the desired accuracy for the solution.

An example of the closure process is shown in Fig 22. These data are for station 2,3 at time station 40. The line load at this time station was located at $j = 3$. The creeping behavior of the closure, noted in Chapter 5, is clearly seen in this curve. Twenty iterations were required to achieve



(a) Plate on foundation.

4 × 15 grid

$\rho = 7.5 \times 10^{-4}$ lb-sec^2/in^3

$D_x = D_y = 2.5 \times 10^6$ lb-in^2/in

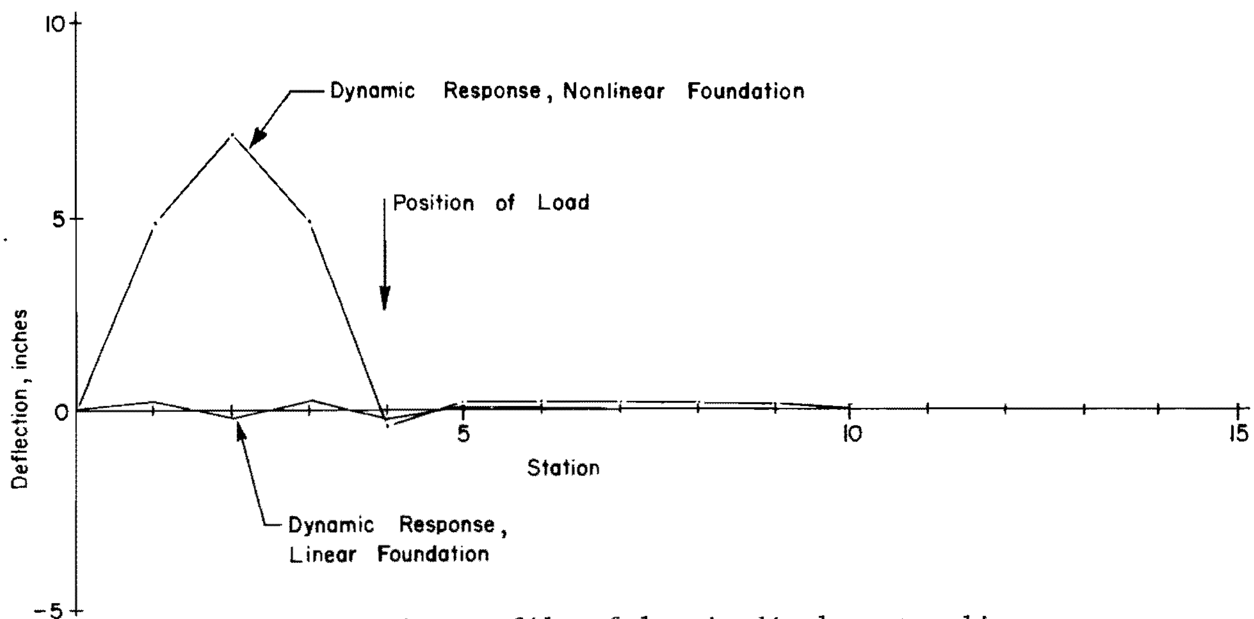
$\nu = 0.25$

s = 200 lb/in^3

$h_x = h_y = 48$ in.

$h_t = 0.005$ sec

Closure tolerance = 10^{-5} in.



(b) Deflection profile of longitudinal center line.

Fig 21. Moving load on a plate on foundation.

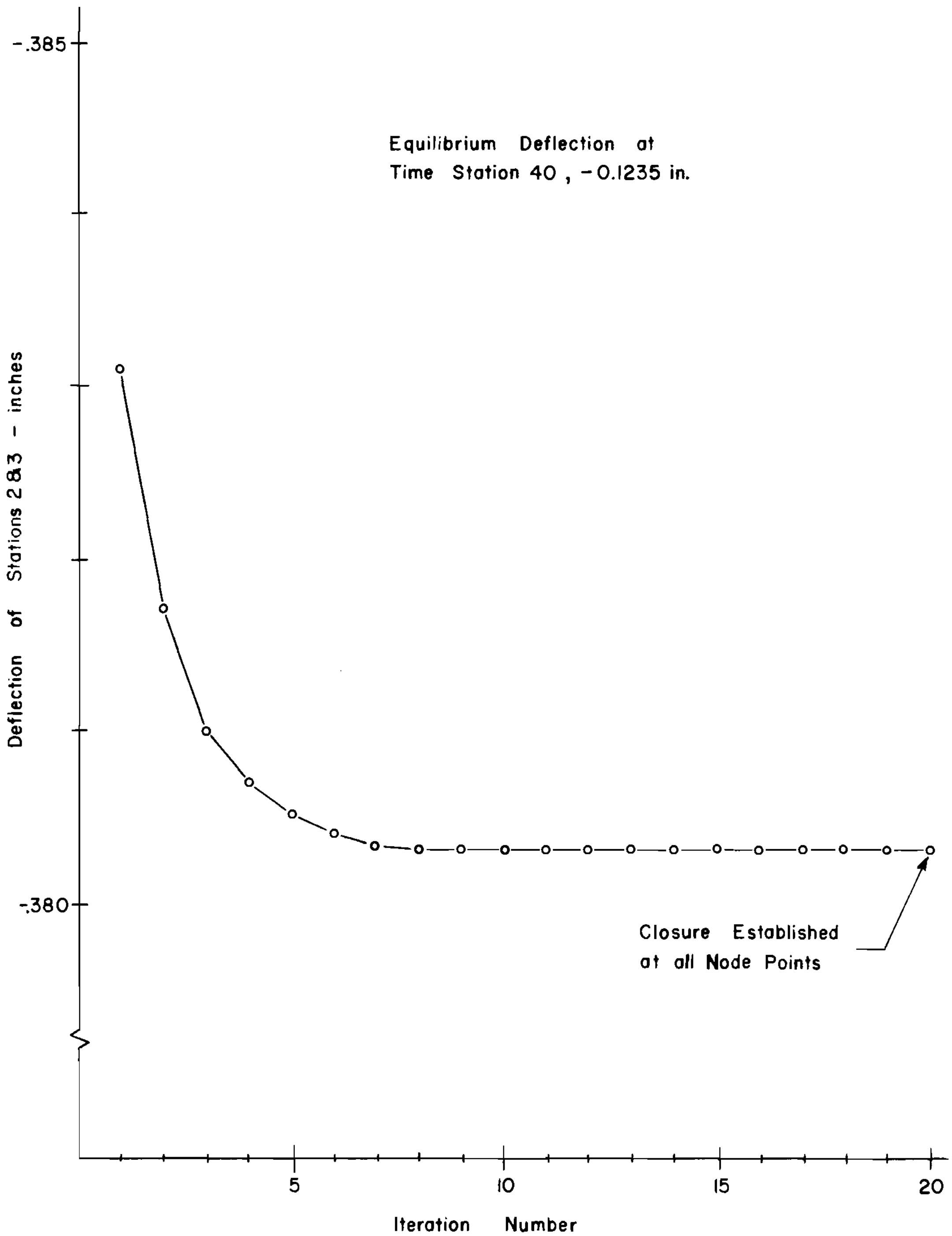


Fig 22. Closure plot for station 2,3
at time station 40.

closure for all node points, even though the curve of Fig 22 indicates closure for station 2,3 within 10 iterations.

Bridge Approach Slab

An example of how the results of this study might be applied to a highway pavement problem is shown in Fig 23. The approach slab connects the pavement with the bridge deck and is supported on one end by the abutment bent and by the base material over half of its length.

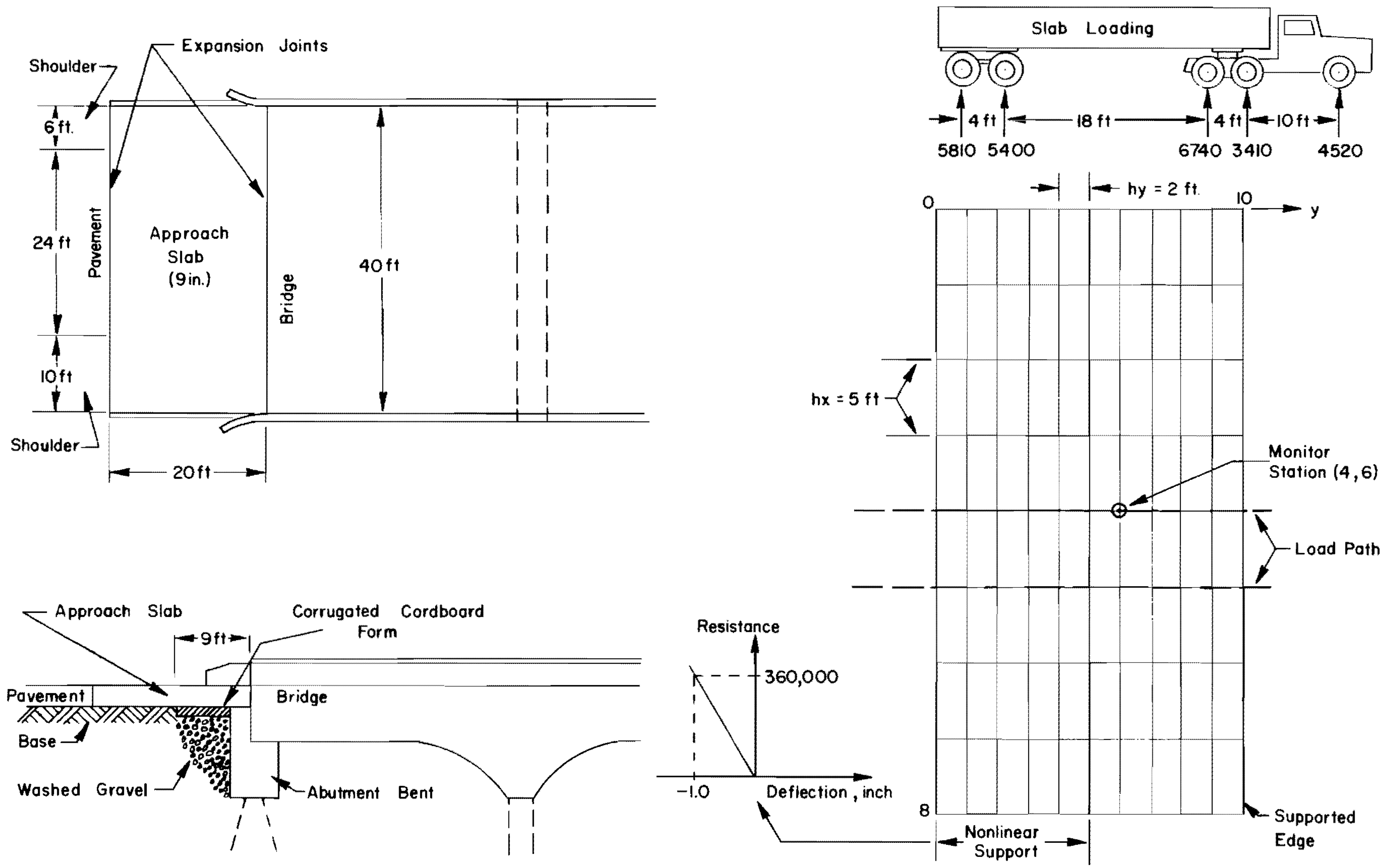
A coefficient of subgrade reaction for the base material of 250 lb/in^3 was selected for this example. The plate bending stiffness was 2.278×10^8 $lb\text{-}in^2/in$ in both the x and y-directions of the slab, typical for a 9-inch pavement slab. The resistance-deflection characteristics of the base material were represented by a bilinear curve (Fig 23). The connection of the slab and the abutment bent was a hinge support. No resistance was offered the slab in the area of the cardboard form material. A closure tolerance of 10^{-5} inch was selected for this study.

Loads, representing the truck shown in Fig 23, were moved across the slab at 60 mph. The response of a point in the path of the load for both static and dynamic loads is shown in Fig 24.

Although the mean curve through the dynamic response data approaches the static deflection curve, considerable dynamic amplification is noted by the peak values. These peak deflections and stresses resulting from the dynamic response of the system may be responsible for fatigue damage to the slab material.

Summary

Four types of example problems were solved to demonstrate the method of analysis. The free vibration problem illustrated the stability of the method as well as its ability to predict the theoretical fundamental period of vibration. A second set of problems demonstrated the propagation of the traveling wave in a simply supported rectangular plate. Comparisons were made between the response of the plate resting on both linear and bilinear foundations, thereby demonstrating the capability to solve nonlinear problems. Finally a practical highway problem, a bridge approach slab, was solved to demonstrate the application of the method to a typical engineering problem.



(a) Location of approach slab.

(b) Discrete-element model of approach slab.

Fig 23. Bridge approach slab.

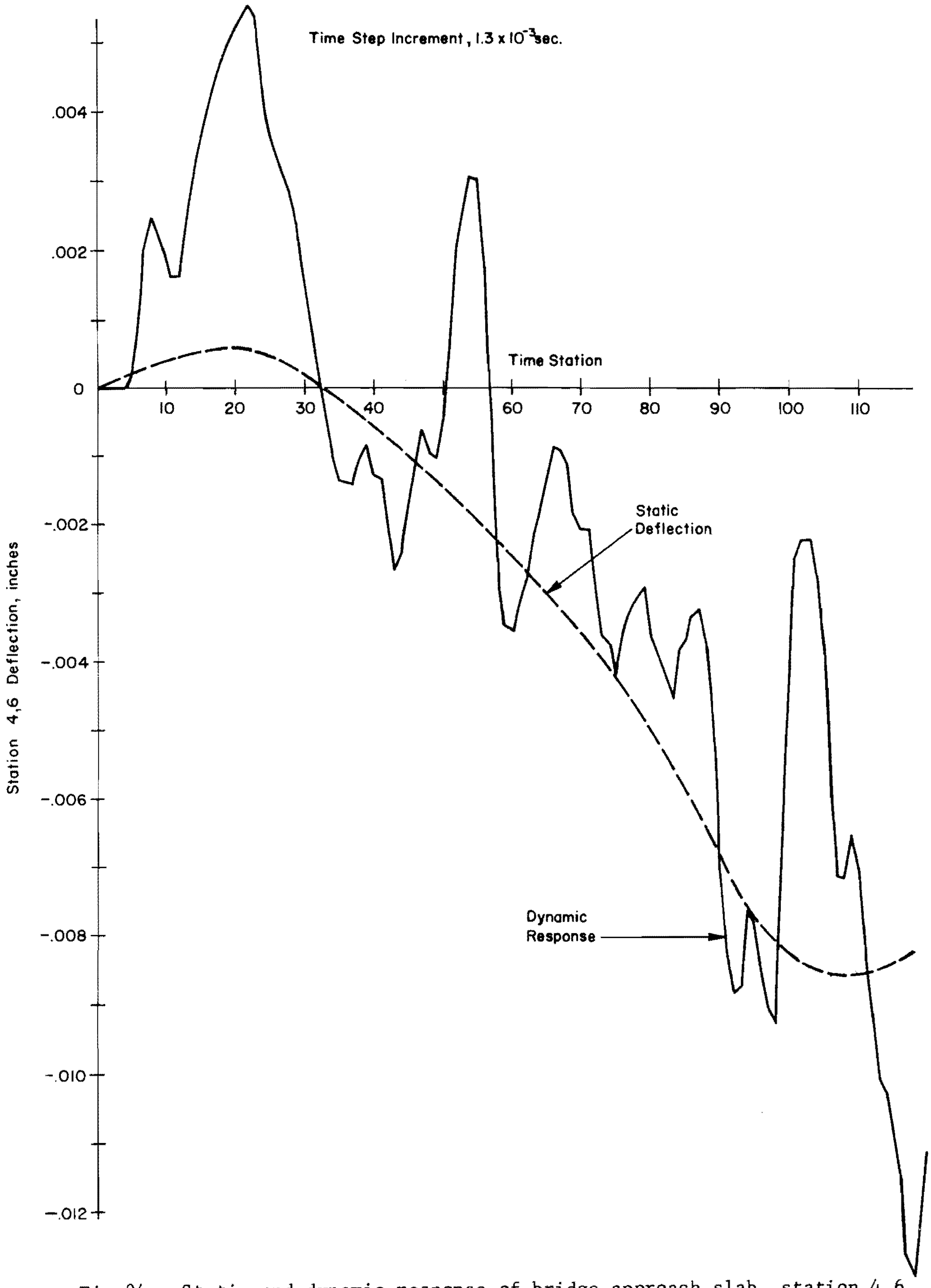


Fig 24. Static and dynamic response of bridge approach slab, station 4,6.

The first two examples serve to develop confidence in the method for solving linear problems. The third example, on the other hand, presents the solution capability of the algorithm for plates on nonlinear foundations. Although experimental data are lacking for a correlation study of the proposed nonlinear procedure, the nonlinear results appear reasonable when compared with solutions for the plate on a linear foundation.

CHAPTER 10. SUMMARY AND RECOMMENDATIONS FOR FURTHER RESEARCH

The result of this work was the development of a method for the dynamic analysis of plates resting on nonlinear foundations. A step-by-step numerical integration method was utilized to propagate in time the response of a discrete-element model representing the plate.

The implementation of the analysis procedure was made possible by the high-speed digital computer. The numerical method described in this work was coded in FORTRAN language for the Control Data Corporation (CDC) 6600 digital computer. To permit the analysis of plates with many increments in the x and y-directions, the peripheral storage facilities of this computer were utilized. At the present time, the CDC 6600 at The University of Texas at Austin will handle a 50-increment square plate.

The step-by-step numerical integration procedure was based on the rational assumption of linear acceleration for each node during the time-step interval. The stability of the linear acceleration algorithm was investigated and a method presented for the selection of the time-step increment.

An iterative method for nonlinear analysis, which does not require the adjustment of the stiffness matrix of the structure, was presented. Nonlinear adjustments were made by correction loads which were added to the right-hand-side of the equations. The load iteration technique utilizes an efficient solution procedure known as the multiple load method for the repetitive solutions of the equations. The multiple load method may permit as many as ten load iterations to be performed in the time required for a single stiffness iteration.

The numerical method was organized and programmed in a manner which will facilitate the modification and extension of the method. Future extensions to the model and the program might include relative damping, to represent material damping properties of the plate, and, for highway pavement analysis, the coupling of a vehicle model and pavement roughness characteristics to the plate model for the generation of dynamic loads.

The nonlinear solution capabilities should be extended to the plate bending and twisting stiffness variables. Nonlinear moment-curvature and moment-twist relations could be incorporated in the iterative procedure, thereby permitting analysis of the plate or slab material for stresses in the nonlinear range. Furthermore, capabilities for inelastic analysis should be developed for both the foundation and slab properties.

Studies of the nonlinear closure procedure should be continued. Although the load iteration method appears attractive because of the multiple load solution procedure, methods for accelerating closure should be developed. One method has been mentioned - that of alternating cycles of load iteration with a single cycle of the tangent modulus method. However, it is possible that other, more natural, methods may exist, such as the use of the curvature or slope of the iteration curve for prediction of the equilibrium position.

The existing discrete-element model requires the user to know and specify the distribution of axial or in-plane thrust throughout the plate. A valuable extension of this work would be the modification of the model to include axial deformations, and the development of the force-deformation equations for in-plane thrust. Not only could the axial and bending solutions be coupled for combined axial-bending analysis of plates, but the in-plane analysis could be applied to plane-stress problems. Furthermore, an in-plane solution would be required for the analysis of plates subjected to thermal gradients.

Although this study was not performed for the evaluation of the existing computer equipment, comments are in order about the peripheral storage capabilities and the time required to access this storage. Because the study was performed with the CDC 6600 digital computer, the following remarks should be reviewed with respect to this third generation computer.

If peripheral storage had not been utilized, the problem size would have been limited by the available core storage. To overcome this, disk files were used extensively for data storage. Although the problem size was significantly increased, the access time for reading and writing files was found to be an order of magnitude greater than the time required for the arithmetic operations. To overcome the access time problem, a special, machine-dependent subroutine was incorporated in the program. The fourth generation machines will hopefully not have this limitation.

Finally, experimental data are required for further evaluation of the method. Carefully controlled model studies and field tests are required for dynamic response data. Research of this nature will not only aid in the evaluation of the numerical method but also furnish additional data on material behavior, which could be applied to the discrete-element idealization of the problem.

REFERENCES

1. Agarwal, Sohan L., and W. R. Hudson, "Experimental Verification of Discrete-Element Solutions for Plates and Pavement Slabs," Research Report No. 56-15, Center for Highway Research, The University of Texas at Austin, August 1969.

2. Ang, A. H.-S., and Wallace Prescott. "Equations for Plate-Beam Systems in Transverse Bending," Journal of the Engineering Mechanics Division, Vol 87, No. EM6, American Society of Civil Engineers, December 1969, p 1.

3. Ang, A. H.-S., and N. M. Newmark, "A Numerical Procedure for the Analysis of Continuous Plates," Proceedings of the Second Conference on Electronic Computation, American Society of Civil Engineers, Pittsburgh, September 1960, p 379.

4. Chan, S. P., H. L. Cox, and W. A. Benfield, "Transient Analysis of Forced Vibration of Complex Structural-Mechanical Systems," Journal of the Royal Aeronautical Society, Vol 66, July 1962, p 457.

5. Crandall, Stephen H., Engineering Analysis, McGraw-Hill, 1956.

6. Endres, Frank L., and Hudson Matlock, "An Algebraic Equation Solution Process Formulated in Anticipation of Banded Linear Equations," Research Report 56-19, Center for Highway Research, The University of Texas at Austin, June 1970.

7. Harr, M. E., "Assuming the Vehicle Moves," a report to the Annual Meeting of the Highway Research Board, Special Conference Session on Distress and Failure of Pavements, 1969.

8. Houbolt, John C., "A Recurrence Matrix Solution for the Dynamic Response of Aircraft in Gusts," National Advisory Committee for Aeronautics, Technical Note 2060, March 1950.

9. Hudson, W. Ronald, and Hudson Matlock, "Discontinuous Orthotropic Plates and Slabs," Research Report No. 56-6, Center for Highway Research, The University of Texas at Austin, May 1966.

10. Hurty, Walter C., and Moshe F. Rubinstein, Dynamics of Structures, Prentice-Hall, 1964.

11. Ingram, Wayne B., and Hudson Matlock, "A Finite-Element Method for Bending of Layered Structural Systems," Research Report No. 56-5, Center for Highway Research, The University of Texas at Austin, June 1967.

12. Jones, R., E. N. Lister, and E. N. Thrower, "Dynamic Behaviour of Soils and Foundations," Vibration in Civil Engineering, B. O. Skipp, Editor, Butterworths, 1966.

13. Matlock, Hudson, "Applications of Numerical Methods to Some Structural Problems in Offshore Operations," Proceedings, First Conference on Drilling and Rock Mechanics, The University of Texas, Austin, January 1963.

14. Matlock, Hudson, and T. Allan Haliburton, "A Finite-Element Method of Solution for Linearly Elastic Beam Columns," Research Report 56-1, Center for Highway Research, The University of Texas, Austin, 1965.

15. Matlock, Hudson, and T. Allan Haliburton, "A Program for Finite-Element Solution of Beam-Columns on Nonlinear Supports," 1335 Bonham Terrace, Austin, Texas, 1964.

16. Newmark, Nathan M., "A Method of Computation for Structural Dynamics," Journal of the Engineering Mechanics Division, Vol 85, No. EM3, American Society of Civil Engineers, July 1959, p 67.

17. Panak, John J., and Hudson Matlock, "A Discrete-Element Method of Multiple-Loading Analysis for Two-way Bridge Floor Slabs," Research Report No. 56-13, Center for Highway Research, The University of Texas at Austin, January 1970.

18. Richtmyer, Robert D., Difference Methods for Initial-Value Problem, Interscience Publishers, New York, 1957.

19. Salani, Harold J., and Hudson Matlock, "A Finite-Element Method for Vibrating Beams and Plates," Research Report No. 56-8, Center for Highway Research, The University of Texas at Austin, June 1967.

20. Stelzer, C. Fred, Jr., and W. Ronald Hudson, "A Direct Computer Solution for Plates and Pavement Slabs," Research Report 56-9, Center for Highway Research, The University of Texas at Austin, October 1967.

21. "The AASHO Road Test — History and Description of Project," Special Report 61A, Highway Research Board, 1961.

22. Timoshenko, S., and S. Woinowsky-Krieger, Theory of Plates and Shells, McGraw-Hill, 1959.

23. Troitsky, M. S., "Orthotropic Bridges Theory and Design," The James F. Lincoln Arc Welding Foundation, Cleveland, Ohio, 1967.

24. Tucker, Richard L., "A General Method for Solving Grid-Beam and Plate Problems," Ph.D. Dissertation, The University of Texas, Austin, 1963.

25. Tucker, Richard L., "Lateral Analysis of Piles with Dynamic Behavior," Conference on Deep Foundation, Mexico City, Mexico, December 1964.

26. Volterra, Enricco, and E. C. Zachmanoglou, Dynamics of Vibrations, Charles E. Merrill Books, Columbus, Ohio, 1965.

27. Wilson, Edward L., and Ray W. Clough, "Dynamic Response by Step-by-Step Matrix Analysis," Symposium on the Use of Computers in Civil Engineering, Lisbon, Portugal, October 1962.

APPENDIX A

DERIVATION OF EQUILIBRIUM EQUATION FOR DISCRETE-ELEMENT PLATE MODEL

APPENDIX A. DERIVATION OF EQUILIBRIUM EQUATION FOR DISCRETE-ELEMENT PLATE MODEL

The basic derivation of the equilibrium equation has been presented elsewhere (Refs 17 and 20), but is presented here in detail, for the benefit of the reader.

The discrete-element model is shown in Fig 2. It consists of rigid bars, elastic restraints at joints or nodes, and torsion bars connecting the middle of the rigid bars.

An expanded view of a joint is shown in Fig A1. The elastic elements are replaced by the forces and moments which are developed as the node points of the model undergo deformations. All forces and moments are shown in their positive direction.

Equilibrium of the joint of Fig A1 in the z or w-direction is satisfied by

$$\Sigma F_z = Q_{i,j} + V^x_{i,j} + V^y_{i,j} - V^x_{i+1,j} - V^y_{i,j+1} - S_{i,j}w_{i,j} \qquad (A.1)$$

Moment equilibrium for each of the bars will yield

$$-h_x V^x_{i,j} = M^{yx}_{i,j} - M^{yx}_{i,j+1} + M^x_{i-1,j} - M^x_{i,j}$$

$$+ P^x_{i,j}(-w_{i-1,j} + w_{i,j}) \qquad (A.2)$$

$$-h_x V^x_{i+1,j} = M^{yx}_{i+1,j} - M^{yx}_{i+1,j+1} + M^x_{i,j} - M^x_{i+1,j}$$

$$+ P_{i+1,j}(-w_{i,j} + w_{i+1,j}) \qquad (A.3)$$

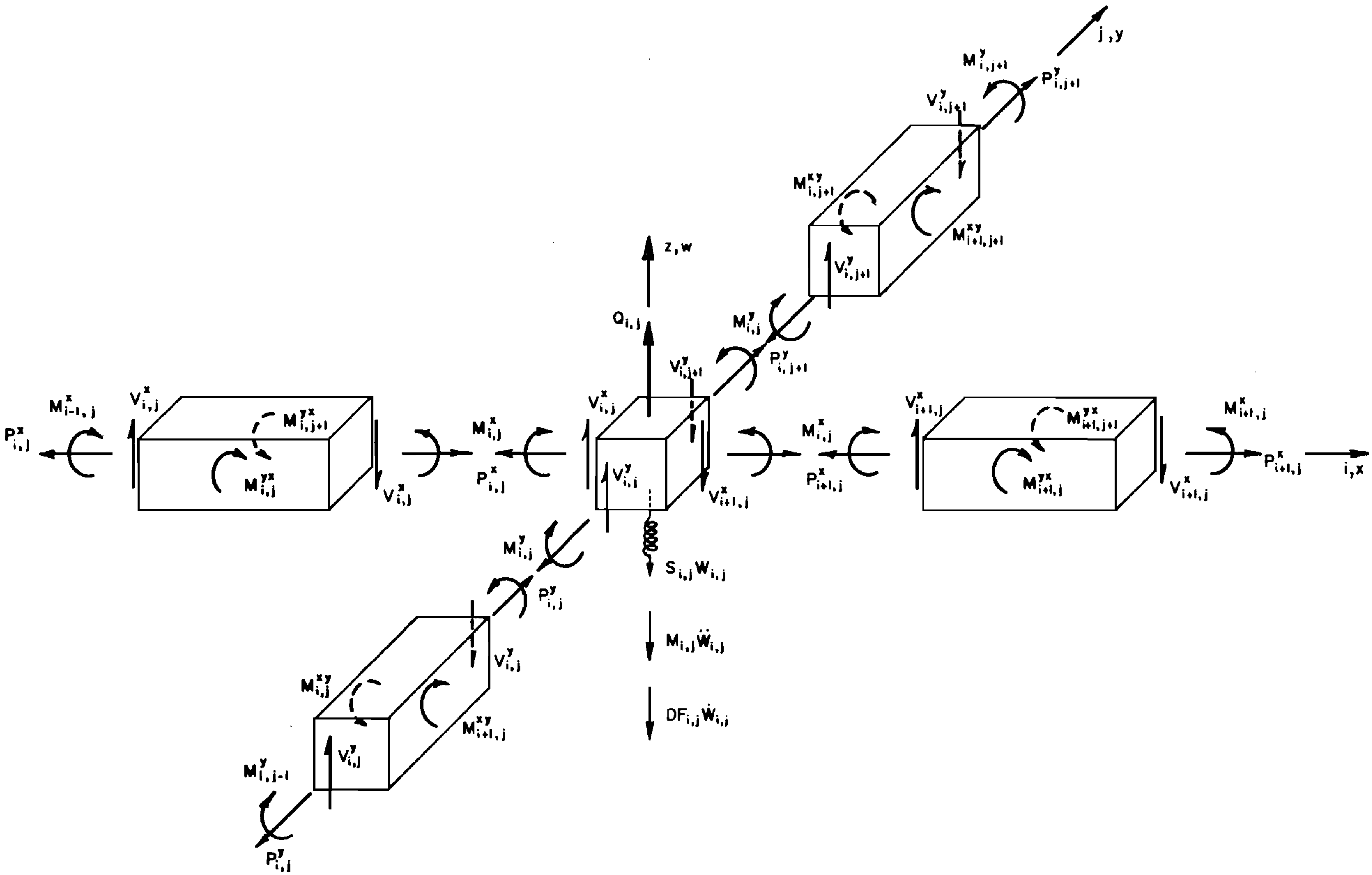


Fig A1. Expanded view and free bodies of model joint and connecting bars.

$$-h_y V^y_{i,j} = -M^{xy}_{i,j} + M^{xy}_{i+1,j} + M^y_{i,j-1} - M^y_{i,j}$$

$$+ P^y_{i,j}(-w_{i,j-1} + w_{i,j}) \qquad (A.4)$$

$$-h_y V^y_{i,j+1} = -M^{xy}_{i,j+1} + M^{xy}_{i+1,j+1} + M^y_{i,j} - M^y_{i,j+1}$$

$$+ P^y_{i,j+1}(-w_{i,j} + w_{i,j+1}) \qquad (A.5)$$

Substituting Eqs A.2 through A.5 into Eq A.1

$$\frac{1}{h_x}\Big[M^{yx}_{i,j} - M^{yx}_{i,j+1} - M^{yx}_{i+1,j} + M^{yx}_{i+1,j+1} + M^x_{i-1,j} - 2M^x_{i,j}$$

$$+ M^x_{i+1,j} + P^x_{i,j}(-w_{i-1,j} + w_{i,j}) - P^x_{i+1,j}(-w_{i,j}$$

$$+ w_{i+1,j})\Big] + \frac{1}{h_y}\Big[-M^{xy}_{i,j} + M^{xy}_{i+1,j} + M^{xy}_{i,j+1}$$

$$- M^{xy}_{i+1,j+1} + M^y_{i,j-1} - 2M^y_{i,j} + M^y_{i,j+1} + P^y_{i,j}(-w_{i,j-1}$$

$$+ w_{i,j}) - P^y_{i,j+1}(-w_{i,j} + w_{i,j+1})\Big] = Q_{i,j} - S_{i,j}w_{i,j} \qquad (A.6)$$

Node point and torsion bar elastic constants are related to the continuum plate constants through finite-difference approximations. As noted in Chapter 2, the continuum variables are represented by subscripts x and y and terms with superscripts pertain to discrete or concentrated data.

The continuum bending moment is related to curvature by elastic stiffness constants D_x and D_y and Poisson's ratio values ν_x and ν_y :

$$M_x = D_x \left(\frac{\partial^2 w}{\partial x^2} + \nu_y \frac{\partial^2 w}{\partial y^2} \right) \qquad (A.7)$$

$$M_y = D_y \left(\frac{\partial^2 w}{\partial y^2} + \nu_x \frac{\partial^2 w}{\partial x^2} \right) \tag{A.8}$$

In Eq A.7, ν_y represents the influence of curvature in the y-direction on curvature in the x-direction. Similarly, in Eq A.8, the cross sensitivity of x on y-curvature is ν_x. Furthermore, the Poisson's ratio values are not independent, but related to the bending stiffness by

$$D_x \nu_y = D_y \nu_x \tag{A.9}$$

The above relationship can be proved by the Maxwell-Betti theorem. The bending moments may therefore be expressed as a function of three variables, D_x, D_y, and α, given by Eq A.9:

$$M_x = D_x \frac{\partial^2 w}{\partial x^2} + \alpha \frac{\partial^2 w}{\partial y^2} \tag{A.10}$$

$$M_y = D_y \frac{\partial^2 w}{\partial y^2} + \alpha \frac{\partial^2 w}{\partial x^2} \tag{A.11}$$

Replacing the curvature by a central difference approximation gives the model moment:

$$M^x_{i,j} = D_{x_{i,j}} \left(\frac{h_y}{h_x^2} \right) (w_{i-1,j} - 2w_{i,j} + w_{i+1,j})$$

$$+ \alpha_{i,j} \left(\frac{h_y}{h_y^2} \right) (w_{i,j-1} - 2w_{i,j} + w_{i,j+1}) \tag{A.12}$$

$$M^y_{i,j} = D_{y_{i,j}} \left(\frac{h_x}{h_y^2} \right) (w_{i,j-1} - 2w_{i,j} + w_{i,j+1})$$

$$+ \alpha_{i,j} \left(\frac{h_x}{h_x^2} \right)(w_{i-1,j} - 2w_{i,j} + w_{i+1,j}) \tag{A.13}$$

The bending moment at adjacent node points is given by similar expressions.

It is possible to relate the model and continuum twisting moments. First, consider the continuum relationship between twisting moment and plate twist.

$$M_{xy} = -D_{xy} \frac{\partial^2 w}{\partial x \partial y} \tag{A.14}$$

The first subscript defines the direction of the moment vector while the second indicates the surface to which the moment is applied. The moment vector is parallel to the axis defined by the first subscript and acting on a vertical plane which is parallel to the second (Fig 1). For equilibrium, the moment vector parallel to the y-axis is related to that in the x-direction by:

$$M_{yx} = -M_{xy} \tag{A.15}$$

If the partial derivative is replaced by a difference expression, the discrete-element twisting moment is obtained:

$$M^{xy}_{i,j} = -D_{xy_{i,j}} \left(\frac{h_y}{h_x h_y} \right)(w_{i-1,j-1} - w_{i,j-1}$$

$$- w_{i-1,j} + w_{i,j}) \tag{A.16}$$

$$M^{yx}_{i,j} = D_{xy_{i,j}} \left(\frac{h_x}{h_x h_y} \right)(w_{i-1,j-1} - w_{i,j-1}$$

$$- w_{i-1,j} + w_{i,j}) \tag{A.17}$$

Again, similar expressions are found for the twisting moments acting on adjacent bars.

It is important to note that there is a fundamental difference between the bending and twisting moments for the discrete-element model. Bending moments are generated at the node points while the twisting moments are developed in torsion bars attached to the midpoints of adjacent parallel bars. It is not possible therefore to refer to the twisting moment at a node point.

The equilibrium equation for a node point is found by substituting the model bending equations (A.12 and A.13) and twisting moment equations (A.16 and A.17) into Eq A.6. A relationship between stiffness and node point deflection is established.

$$\begin{aligned} & a_{i,j} w_{i,j-2} + b^1_{i,j} w_{i-1,j-1} + b^2_{i,j} w_{i,j-1} + b^3_{i,j} w_{i+1,j-1} \\ & + c^1_{i,j} w_{i-2,j} + c^2_{i,j} w_{i-1,j} + c^3_{i,j} w_{i,j} + c^4_{i,j} w_{i+1,j} \\ & + c^5_{i,j} w_{i+2,j} + d^1_{i,j} w_{i-1,j+1} + d^2_{i,j} w_{i,j+1} \\ & + d^3_{i,j} w_{i+1,j+1} + e_{i,j} w_{i,j+2} = q_{i,j} \end{aligned} \tag{A.18}$$

The coefficients of the deflection terms are

$$a_{i,j} = \frac{h_x}{h_y^3} (D_{y_{i,j-1}}) \tag{A.19}$$

$$b^1_{i,j} = \frac{1}{h_x h_y} (2D_{xy_{i,j}} + \alpha_{i-1,j} + \alpha_{i,j-1}) \tag{A.20}$$

$$\begin{aligned} b^2_{i,j} = & -2 \frac{h_x}{h_y^3} (D_{y_{i,j-1}} + D_{y_{i,j}}) - \frac{2}{h_x h_y} (D_{xy_{i,j}} \\ & + D_{xy_{i+1,j}} + \alpha_{i,j} + \alpha_{i,j-1}) - \frac{1}{h_y} P^y_{i,j} \end{aligned} \tag{A.21}$$

$$b^3_{i,j} = \frac{1}{h_x h_y}(2D_{xy_{i+1,j}} + \alpha_{i,j-1} + \alpha_{i+1,j}) \qquad (A.22)$$

$$c^1_{i,j} = \frac{h_y}{h_x^3}(D_{x_{i-1,j}}) \qquad (A.23)$$

$$c^2_{i,j} = -\frac{2h_y}{h_x^3}(D_{x_{i-1,j}} + D_{x_{i,j}}) - \frac{2}{h_x h_y}(\alpha_{i-1,j}$$

$$+ \alpha_{i,j} + D_{xy_{i,j}} + D_{xy_{i,j+1}}) - \frac{P^x_{i,j}}{h_x} \qquad (A.24)$$

$$c^3_{i,j} = \frac{h_y}{h_x^3}(D_{x_{i-1,j}} + 4D_{x_{i,j}} + D_{x_{i+1,j}})$$

$$+ \frac{h_x}{h_y^3}(D_{y_{i,j-1}} + 4D_{y_{i,j}} + D_{y_{i,j+1}})$$

$$+ \frac{2}{h_x h_y}(D_{xy_{i,j}} + D_{xy_{i+1,j}} + D_{xy_{i,j+1}} + D_{xy_{i+1,j+1}}$$

$$+ 4\alpha_{i,j}) + \frac{1}{h_x}(P^x_{i,j} + P^x_{i+1,j}) + \frac{1}{h_y}(P^y_{i,j}$$

$$+ P^y_{i,j+1}) + S_{i,j} \qquad (A.25)$$

$$c^4_{i,j} = -\frac{2h_y}{h_x^3}(D_{x_{i,j}} + D_{x_{i+1,j}}) - \frac{2}{h_x h_y}(D_{xy_{i+1,j}}$$

$$+ D_{xy_{i+1,j+1}} + \alpha_{i,j} + \alpha_{i+1,j}) - \frac{P^x_{i+1,j}}{h_x} \qquad (A.26)$$

$$c^5_{i,j} = \frac{h_y}{h_x^3} (D_{x_{i+1,j}}) \tag{A.27}$$

$$d^1_{i,j} = \frac{1}{h_x h_y} (2D_{xy_{i,j+1}} + \alpha_{i-1,j} + \alpha_{i,j+1}) \tag{A.28}$$

$$d^2_{i,j} = - \frac{2h_x}{h_y^3} (D_{y_{i,j}} + D_{y_{i,j+1}}) - \frac{2}{h_x h_y} (D_{xy_{i,j+1}}$$

$$+ D_{xy_{i+1,j+1}} + \alpha_{i,j} + \alpha_{i,j+1}) - \frac{P^y_{i,j+1}}{h_y} \tag{A.29}$$

$$d^3_{i,j} = \frac{1}{h_x h_y} (2D_{xy_{i+1,j+1}} + \alpha_{i+1,j} + \alpha_{i,j+1}) \tag{A.30}$$

$$e_{i,j} = \frac{h_x}{h_y^3} (D_{y_{i,j+1}}) \tag{A.31}$$

$$q_{i,j} = Q_{i,j} \tag{A.32}$$

These equations may be written in matrix notation:

$$\left[K\right] \left\{w\right\} = \left\{Q\right\} \tag{A.33}$$

where $\left[K\right]$ is the stiffness matrix of the plate. The terms in $\left[K\right]$ are given by Eqs A.19 through A.31. The load vector $\left\{Q\right\}$ is given by Eq A.32.

APPENDIX B

EQUATION OF MOTION FOR DISCRETE-ELEMENT PLATE MODEL

APPENDIX B. EQUATION OF MOTION FOR DISCRETE-ELEMENT PLATE MODEL

A free-body diagram of the discrete-element plate model for dynamic analysis is shown in Fig B1. A dashpot, to represent viscous damping, is attached to the node point as well as the fixed reference plane. The equation of motion for each node is developed by adding the inertia and damping forces to the node equilibrium equation (A.18):

$$
\begin{aligned}
& a_{i,j}w_{i,j-2} + b^1_{i,j}w_{i-1,j-1} + b^2_{i,j}w_{i,j-1} + b^3_{i,j}w_{i+1,j-1} \\
& + c^1_{i,j}w_{i-2,j} + c^2_{i,j}w_{i-1,j} + c^3_{i,j}w_{i,j} + c^4_{i,j}w_{i+1,j} \\
& + c^5_{i,j}w_{i+2,j} + d^1_{i,j}w_{i-1,j+1} + d^2_{i,j}w_{i,j+1} \\
& + d^3_{i,j}w_{i+1,j+1} + e_{i,j}w_{i,j+2} = q_{i,j} \\
& - M_{i,j}\left(\frac{d^2w_{i,j}}{dt^2}\right) - DF_{i,j}\left(\frac{dw_{i,j}}{dt}\right) \qquad \text{(B.1)}
\end{aligned}
$$

The new terms in Eq B.1 ($M_{i,j}$ and $DF_{i,j}$) are the nodal mass and damping. The units of mass must be lb-sec^2/in and those of damping lb-sec/in.

It will be convenient to combine the equations of motion for the node points and write in matrix form:

$$\left[K\right]\left\{w\right\} = \left\{Q\right\} - \left[M\right]\left\{\ddot{w}\right\} - \left[DF\right]\left\{\dot{w}\right\} \qquad \text{(B.2)}$$

or in the more familiar form

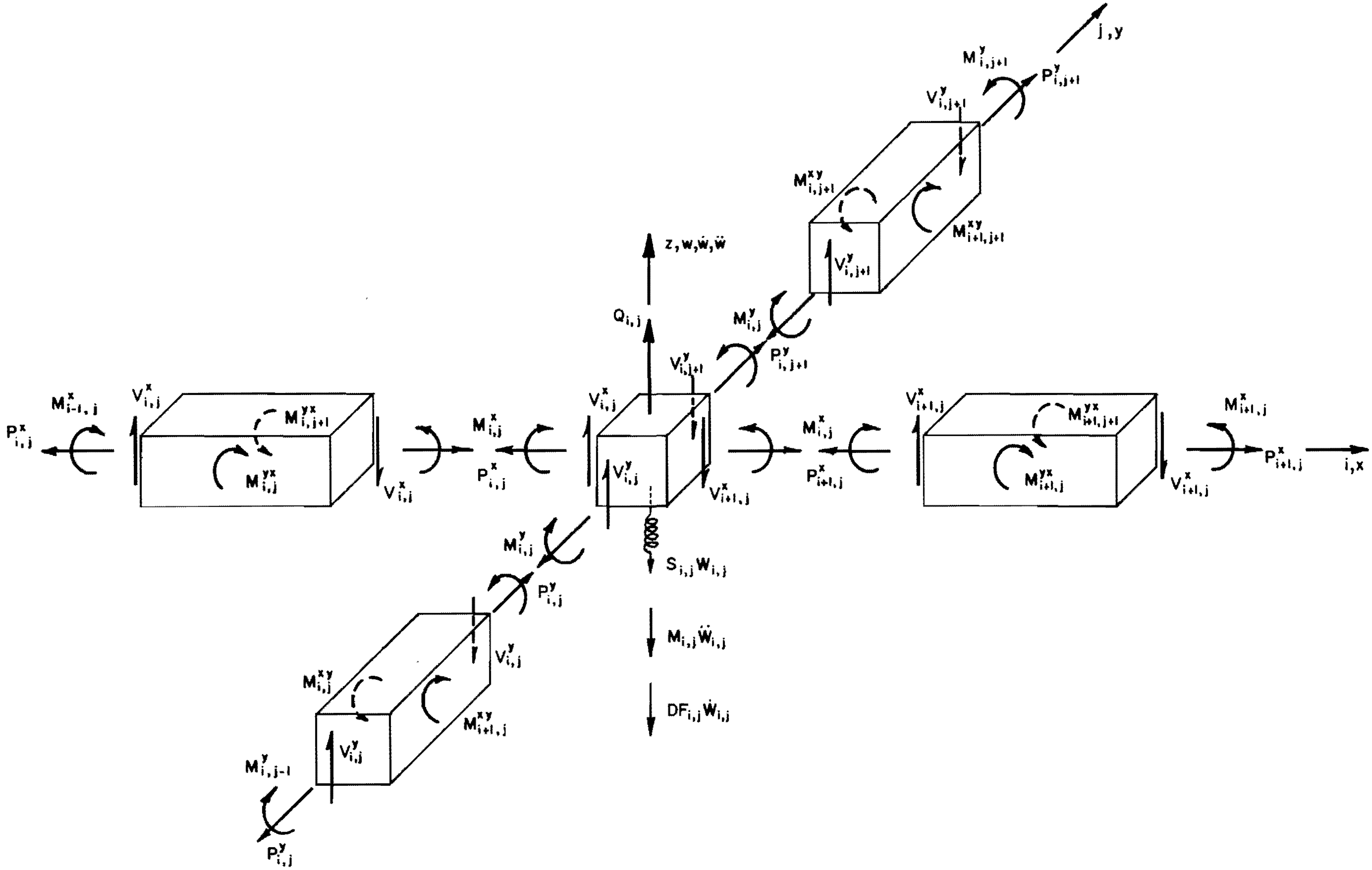


Fig B1. Expanded view and free bodies of dynamic model joint and connecting bars.

$$\left[M \right] \left\{ \ddot{w} \right\} + \left[DF \right] \left\{ \dot{w} \right\} + \left[K \right] \left\{ w \right\} = \left\{ Q \right\} \qquad (B.3)$$

In Eqs B.2 and B.3, differentiation with respect to time is conveniently represented by the dot above the deflection.

The mass matrix $\left[M \right]$ and damping matrix $\left[DF \right]$ are diagonal. This is the result of structural idealization given in Fig 3. The mass matrix would take a different form if the mass were lumped in the bars. Furthermore, relative damping would change the form of the damping matrix.

APPENDIX C

STEP-BY-STEP NUMERICAL INTEGRATION PROCEDURE

APPENDIX C. STEP-BY-STEP NUMERICAL INTEGRATION PROCEDURE

The equations included in this appendix were derived especially for use with the method developed during this study. They are coincidentally a specialized form of the equations presented by Cox et al (Ref 4) for step-by-step analysis of structural systems.

The numerical integration of the equations of motion is based on the assumption that the acceleration varies linearly during each time step. The velocity and deflection, therefore, are dependent on the conditions at the beginning of the time step and the acceleration at the end of the interval:

$$\dot{w}_{k+1} = \dot{w}_k + \frac{h_t}{2}\ddot{w}_k + \frac{h_t}{2}\ddot{w}_{k+1} \tag{C.1}$$

$$w_{k+1} = w_k + h_t\dot{w}_k + \frac{h_t^2}{3}\ddot{w}_k + \frac{h_t^2}{6}\ddot{w}_{k+1} \tag{C.2}$$

It is possible to combine Eqs C.1 and C.2 with those for time increment $k-1$ to k and express the velocity and acceleration as a function of deflection and the time increment length h_t :

$$\dot{w}_k = \dot{w}_{k-1} + \frac{h_t}{2}\ddot{w}_{k-1} + \frac{h_t}{2}\ddot{w}_k \tag{C.3}$$

$$w_k = w_{k-1} + h_t\dot{w}_{k-1} + \frac{h_t^2}{3}\ddot{w}_{k-1} + \frac{h_t^2}{6}\ddot{w}_k \tag{C.4}$$

Subtracting Eq C.4 from Eq C.2,

$$w_{k+1} - 2w_k + w_{k-1} = h_t(\dot{w}_k - \dot{w}_{k-1}) + \frac{h_t^2}{3}(\ddot{w}_k - \ddot{w}_{k-1})$$

$$+ \frac{h_t^2}{6} (\ddot{w}_{k+1} - \ddot{w}_k) \quad (C.5)$$

The term $h_t(\dot{w}_k - \dot{w}_{k-1})$ may be replaced by $\frac{h_t^2}{2} (\ddot{w}_k + \ddot{w}_{k-1})$ (from, Eq C.3) giving

$$\frac{6}{h_t^2} (w_{k+1} - 2w_k + w_{k-1}) = \ddot{w}_{k+1} + 4\ddot{w}_k + \ddot{w}_{k-1} \quad (C.6)$$

A similar relation between velocity and deflection is found by first adding Eqs C.2 and C.4:

$$w_{k+1} - w_{k-1} = h_t(\dot{w}_k + \dot{w}_{k-1}) \times \frac{h_t^2}{3} (\ddot{w}_k + \ddot{w}_{k-1})$$

$$+ \frac{h_t^2}{6} (\ddot{w}_{k+1} + \ddot{w}_k) \quad (C.7)$$

Next, the terms $(\ddot{w}_k + \ddot{w}_{k-1})$ and $(\ddot{w}_{k+1} + \ddot{w}_k)$ are replaced by Eqs C.3 and C.1. Combining terms leads to the relationship between velocity and deflection

$$\frac{3}{h_t} (w_{k+1} - w_k) = \dot{w}_{k+1} + 4\dot{w}_k + \dot{w}_{k-1} \quad (C.8)$$

The recursive relationship to propagate the solution from one time step to the next is developed by writing the dynamic equilibrium relationships at time k-1 , k , and k+1 :

$$\left[M\right] \left\{\ddot{w}_{k-1}\right\} + \left[DF\right] \left\{\dot{w}_{k-1}\right\} + \left[K\right] \left\{w_{k-1}\right\} = \left\{Q_{k-1}\right\} \quad (C.9)$$

$$\left[M\right] \left\{\ddot{w}_{k}\right\} + \left[DF\right] \left\{\dot{w}_{k}\right\} + \left[K\right] \left\{w_{k}\right\} = \left\{Q_{k}\right\} \quad (C.10)$$

$$\left[M\right]\left\{\ddot{w}_{k+1}\right\} + \left[DF\right]\left\{\dot{w}_{k+1}\right\} + \left[K\right]\left\{w_{k+1}\right\} = \left\{Q_{k+1}\right\} \tag{C.11}$$

Multiplying Eq C.10 by 4 and then adding Eqs C.9, C.10, and C.11 gives

$$\left[M\right]\left\{\ddot{w}_{k+1} + 4\ddot{w}_k + \ddot{w}_{k-1}\right\} + \left[DF\right]\left\{\dot{w}_{k+1} + 4\dot{w}_k + \dot{w}_{k-1}\right\}$$

$$+ \left[K\right]\left\{w_{k+1} + 4w_k + w_{k-1}\right\} = \left\{Q_{k+1} + 4Q_k + Q_{k-1}\right\} \tag{C.12}$$

Substituting Eqs C.6 and C.8 into Eq C.12 gives the following recursive relationship for step-by-step recursive analysis:

$$\left[\frac{6}{h_t^2}\left[M\right] + \frac{3}{h_t}\left[DF\right] + \left[K\right]\right]\left\{w_{k+1}\right\} = \left\{Q_{k+1} + 4Q_k + Q_{k-1}\right\}$$

$$- \left[-\frac{3}{h_t^2}\left[M\right] + \left[K\right]\right]\left\{4w_k\right\} - \left[\frac{6}{h_t^2}\left[M\right] - \frac{3}{h_t}\left[DF\right]\right.$$

$$\left. + \left[K\right]\right]\left\{w_{k-1}\right\} \tag{C.13}$$

As both $\left[M\right]$ and $\left[DF\right]$ are diagonal matrices, only the main diagonal of the stiffness matrix is modified by the operations shown in Eq C.13.

Eq C.13 can be written as

$$\left[K'\right]\left\{w_{k+1}\right\} = \left\{Q'_{k+1}\right\} \tag{C.14}$$

The right-hand side of Eq C.13 is combined, giving an equivalent load vector $\left\{Q'_{k+1}\right\}$. The terms in the modified stiffness matrix $\left[K'\right]$ are given by Eqs A.19 through A.31 with one exception: for dynamic analysis $c^3_{i,j}$ (Eq A.25) becomes

$$c^3_{i,j} = \frac{h_y}{h_x^3} (D_{x_{i-1,j}} + 4D_{x_{i,j}} + D_{x_{i+1,j}})$$

$$+ \frac{h_x}{h_y^3} (D_{y_{i,j-1}} + 4D_{y_{i,j}} + D_{y_{i,j+1}})$$

$$+ \frac{2}{h_x h_y} (D_{xy_{i,j}} + D_{xy_{i+1,j}} + D_{xy_{i,j+1}} + D_{xy_{i+1,j+1}}$$

$$+ 4\alpha_{i,j}) + \frac{1}{h_x} (P^x_{i,j} + P^x_{i+1,j}) + \frac{1}{h_y} (P^y_{i,j} + P^y_{i,j+1})$$

$$+ S_{i,j} + \frac{6}{h_t^2} M_{i,j} + \frac{3}{h_t} DF_{i,j} \qquad \text{(C.15)}$$

GUIDE FOR DATA INPUT FOR SLAB 35

with Supplementary Notes

extract from

DYNAMIC ANALYSIS OF DISCRETE-ELEMENT PLATES ON NONLINEAR FOUNDATIONS

Research Report No. 56-17

by

Allen E. Kelly and Hudson Matlock

January 1970

APPENDIX D

GUIDE FOR DATA INPUT FOR SLAB 35

SLAB 35 GUIDE FOR DATA INPUT -- Card forms

IDENTIFICATION OF PROGRAM AND RUN (2 alphanumeric cards per run)

1–80	Card 1
1–80	Card 2

IDENTIFICATION OF PROBLEM (One card each problem)

1–5	Prob No.
11–80	Description of problem

TABLE 1. PROGRAM CONTROL DATA (Two or more* cards each problem)

6–10	NCT2	Number of Cards in Table 2
11–15	NCT3	Number of Cards in Table 3
16–20	NCT4	Number of Cards in Table 4
21–25	NCT5	Number of Cards in Table 5
26–30	NCR6	Number of Curves in Table 6
31–35	NCR7	Number of Curves in Table 7
46–50	OP	Print Option Control Switch
56–60	MON	Number of Monitor Stations

6–10	MX	Number of Slab Increments X Direction
11–15	MY	Number of Slab Increments Y Direction
16–20	MT	Number of Time Steps
21–30	HX	Increment Length X-Direction
31–40	HY	Increment Length Y-Direction
41–50	HT	Time-Steps Interval
51–60	PR	Maximum Poisson's Ratio

* The first two cards of this table are required for each problem. For linear problems (when NCR7 = 0) the third card must be omitted. Monitor stations to be read in are controlled by the first card (MON).

Iteration Control Data (Not required for linear problems)

Maximum Number of Iterations for any Time Step	Maximum Allowable Deflection	Deflection Closure Tolerance
ITMX	WMAX	TOL
6–10	21–30	30–40

Monitor Stations (Controlled by MON; no cards if MON is blank or as many as 10)

Station X	Station Y
MSX	MSY
6–10	10–15

TABLE 2. ELASTIC PROPERTIES OF THE SLAB (One or more cards for each problem as shown by NCT2 of Table 1.)

From Station X	From Station Y	Thru Station X	Thru Station Y	Bending Stiffness		Linear Foundation Spring
I1	J1	I2	J2	DXN	DYN	SN
6–10	10–15	15–20	20–25	31–40	40–50	50–60

TABLE 3. AXIAL THRUST AND TWISTING STIFFNESS (The number of cards as shown by NCT3 of Table 1.)

From Station X	From Station Y	Thru Station X	Thru Station Y	Axial Thrust X-Direction	Axial Thrust Y-Direction	Twisting Stiffness
I1	J1	I2	J2	PXN	PYN	CTN
6–10	10–15	15–20	20–25	31–40	40–50	50–60

TABLE 4. MASS AND DAMPING PROPERTIES (The number of cards as shown by NCT4 of Table 1.)

From Station		Thru Station			
X	Y	X	Y	Mass	Viscous Damping
I1	J1	I2	J2	RHON	DFN
6–10	10–15	15–20	20–25	31–40	40–50

TABLE 5. STATIC OR DEAD LOADS (The number of cards as shown by NCT5 of TABLE 1.)

From Station		Thru Station			Bending Moment About	
X	Y	X	Y	Lateral Force	X-Axis	Y-Axis
I1	J1	I2	J2	QN	TXN	TYN
6–10	10–15	15–20	20–25	31–40	40–50	50–60

TABLE 6. DYNAMIC LOADING (The number of curves in this section is shown by NCR6 of TABLE 1. The number of cards in each curve is given by NAM.)

Dynamic Load Curve Control Card (One card for each curve)

Num of Cards for Curve	Initial Displ of Load Y-Dir	Periodic Load Option for Curve	Num of Loads this Curve	Velocity of moving Load
NAM	JSFT	JSYM	NDL	MSPD
6–10	10–15	15–20	20–25	31–40

Amplitude Variation Data (NAM cards for each curve, not to exceed 20)

Time Step From K1	Time Step Thru K2	Cont. KONT	Load Amplitude Multiplier DQM
6 – 10	15	20	30

Dynamic Load (NDL cards, not to exceed 20)

From Station X I1	From Station Y J1	Thru Station X I2	Thru Station Y J2	Dynamic Load DQN
6 – 10	15	20	25	31 – 40

TABLE 7. NONLINEAR SUPPORT DATA (The number of nonlinear curves as shown by NCR7 of TABLE 1. Each curve requires three cards.)

Curve Control Card (One card each curve)

From Station X IN1	From Station Y JN1	Thru Station X IN2	Thru Station Y JN2	Load Multiplier QMP	Deflection Multiplier WMP	Linear Approximation SFN	Num of Points on Crv NPC	Enter "1" if Curve Symmetric about Origin ISYM
6 – 10	15	20	25	31 – 40	50	60	65	70

If = 1, final deflection must be positive and initial load and deflection must be zero

Curve Data (Two cards for each curve)

	Point Number	1	2	3	4	5	6	7	8	9	10
Load Points	LP	31 – 35	40	45	50	55	60	65	70	75	80
Deflection Points	MP	31 – 35	40	45	50	55	60	65	70	75	80

STOP CARD (One blank card of the end of each run) (columns 1 – 80)

GENERAL PROGRAM NOTES

The data cards must be in the proper order for the program to run.

All data except the moving vehicle speed must be in units of pounds, inches, and seconds. Vehicle speed is input in miles per hour.

The variable identification on the guide for data input is consistent with the FORTRAN notation of SLAB 35.

All 2 and 5-space words must be right justified integer numbers: . . . . `- 3 7 6`

All 10-space words are floating-point decimal numbers: . . . . `+ 2 . 3 4 5 E + 0 3`

TABLE 1. CONTROL DATA AND CONSTANTS

The number of <u>cards</u> in Tables 2 through 5 should be carefully checked in the assembled data deck.

The number of <u>curves</u> in Tables 6 and 7 should be verified before submitting the deck for a computer run.

Output listings for deflection and moment are made for all nodes every OP time steps; in the interval between complete listings only monitor station data are printed.

A single value of Poisson's ratio is input. For orthotropic plate analysis, the larger of the two Poisson's ratio values (ν_x and ν_y) is input.

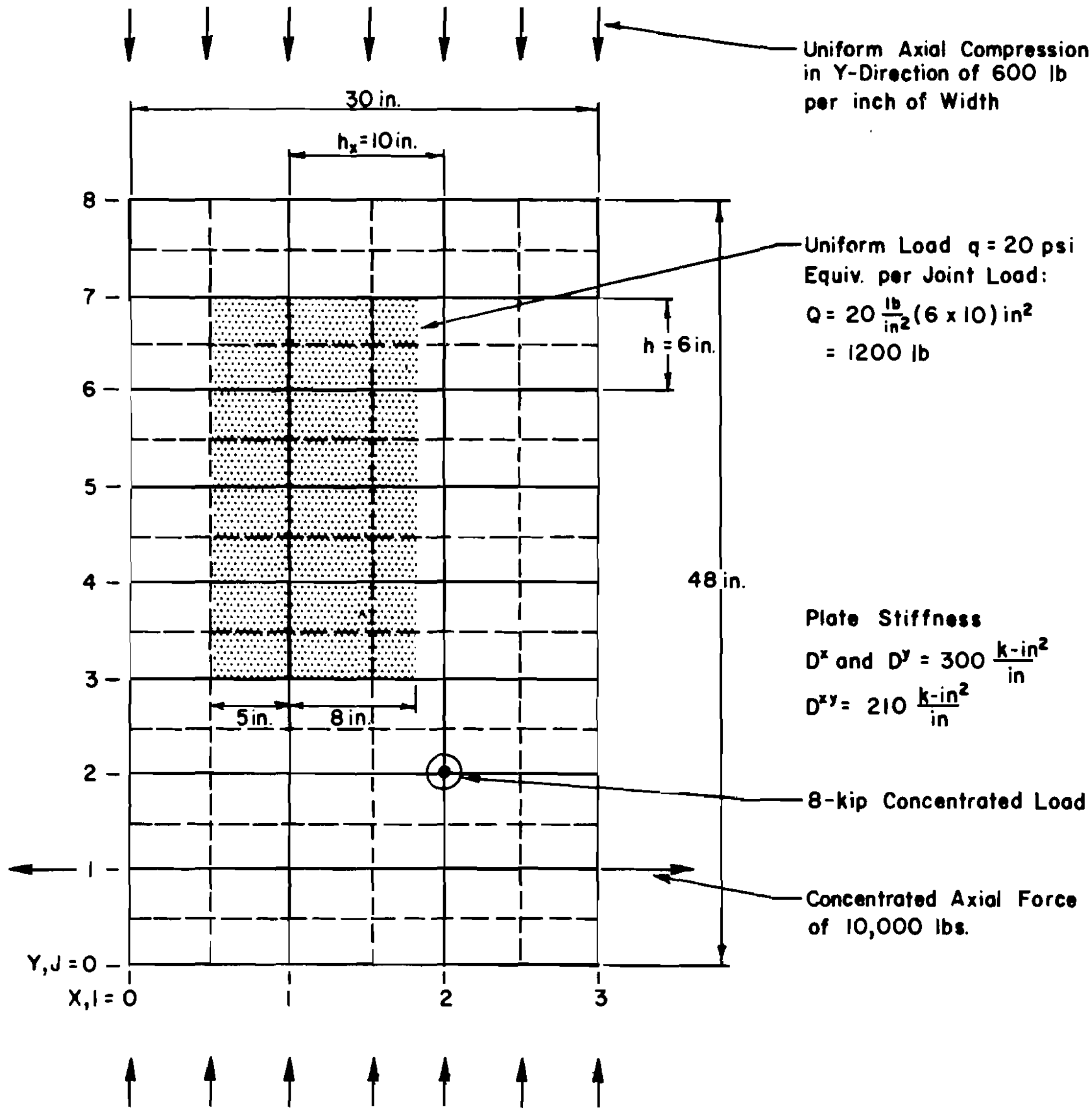
The deflection closure tolerance has the units of inches. For many plate and slab problems a value in the range 10^{-3} to 10^{-6} is adequate to insure closure.

TABLE 2. ELASTIC PROPERTIES OF THE SLAB

Variables:	X-Direction Bending Stiffness DXN	Y-Direction Bending Stiffness DYN	Linear Support Spring SN
Units:	$\frac{\text{lb-in}^2}{\text{in.}}$	$\frac{\text{lb-in}^2}{\text{in.}}$	$\frac{\text{lb}}{\text{in.}}$

The maximum number of cards in Table 2 is 50.

Data are described by a node coordinate identification as shown in Fig D1.



From I1	From J1	Through I2	Through J2	D_x and D_y	D_{xy}	Q	P^x	P^y
0	0	3	8	3.000E + 05	2.100E + 05			
1	3	1	7			-1.200E + 03		
2	3	2	7			-3.600E + 02		
2	2	2	2			-8.000E + 03		
0	1	3	1				+1.000E + 04	
0	0	3	8					-6.000E + 03
0	0	0	8					3.000E + 03
3	0	3	8					3.000E + 03

Note: data incomplete for this sample

Fig D1. Example of organization of plate variables and sustained forces for data input and sample output.

An unyielding support is specified by a support spring greater than 10^{30}.

Data may be distributed to every node in an area by specifying the lower left-hand and upper right-hand coordinates. Quarter-values are automatically placed at corner nodes and half-values at edge nodes. For line specifications, half-values are placed at the starting and ending nodes. Data for a single point will be identified by placing the same node coordinates in both the "From" and "Thru" columns.

Coordinates I2 and J2 must either be equal to or greater than coordinates I1 and J1 .

No restrictions are placed on the Table 2 card order.

Cumulative input is possible (see Fig D1). Data on each card are added to preceding card values.

TABLE 3. AXIAL THRUST AND TWISTING STIFFNESS

Variables:	X-Direction Axial Thrust PXN	Y-Direction Axial Thrust PYN	Twisting Stiffness CTN
Units:	lb	lb	$\frac{\text{lb-in}^2}{\text{in.}}$

The maximum number of cards in Table 3 is 50.

Axial thrusts are bar data, i.e., a single node cannot be used to describe the force.

Tension is positive (+) and compression negative (-).

Area and line specifications are available for axial thrust.

A full value of axial thrust is placed in all bars in the area defined by the "From" and "Through" coordinates, including bars which define the edge of the area.

Twisting stiffness is an area variable and can only be described by area coordinates; I2 and J2 must be greater than I1 and J1 .

Data are distributed with full values to all grid areas defined by the "From" and "Through" coordinates.

Data in this table are cumulative.

TABLE 4. MASS AND DAMPING PROPERTIES

Variables:	Node Point Mass RHON	Node Point Damping DFN
Units:	$\frac{lb\text{-}sec^2}{in.}$	$\frac{lb\text{-}sec}{in.}$

The maximum number of cards for Table 4 is 50.

Mass and damping are node data and are described and distributed as Table 2 data; node point, line, and area descriptions are available. Quarter values of variables are placed at corners of areas and half values of the variables are placed at ends of lines and area edges.

TABLE 5. STATIC OR DEAD LOAD

Variables:	Lateral Load QN	X-Direction Couple Moment TXN	Y-Direction Couple Moment TYN
Units:	lb	lb-in	lb-in

The maximum number of cards for Table 5 is 50.

Variables in this table are node data and are described and distributed as outlined in Table 2.

TABLE 6. DYNAMIC LOADING

Variables:	Dynamic Load Multiplier DQM	Dynamic Load DQN	Speed of Load in Y-Direction MSPD
Units:	-	lb	$\frac{mi}{hr}$

The number of curves in Table 6 cannot exceed 20.

As many as 20 points can be used to define each multiplier curve.

The maximum number of loads which can be input for a single curve is 20.

The multiplier curve is developed as shown in Fig D2 with the requirement that at time station zero the amplitude multiplier must be zero.

A periodic multiplier is generated by a 1 in JSYM; leaving this field blank produces a nonperiodic curve.

Loads moving in the positive y-direction must be entered with a positive speed while those in the opposite direction are negative.

The loads described are shifted in the positive or negative y-direction the number of stations given by JSFT. This feature is used with the moving load capabilities so that a load can be described on the slab then shifted to a point where it can run across the slab.

The loads controlled by the curve are NDL and are input following the multiplier curve.

Loads can be described for a point, line, or area. Rules for the description of loads are given in the discussion of Table 2 data.

TABLE 7. NONLINEAR SUPPORT DATA

Variables:	Linear Approximation of Nonlinear Curve SFN	Scaled Foundation Resistance LP	Scaled Foundation Deflection MP
Units:	$\frac{\text{lb}}{\text{in.}}$	lb	in.

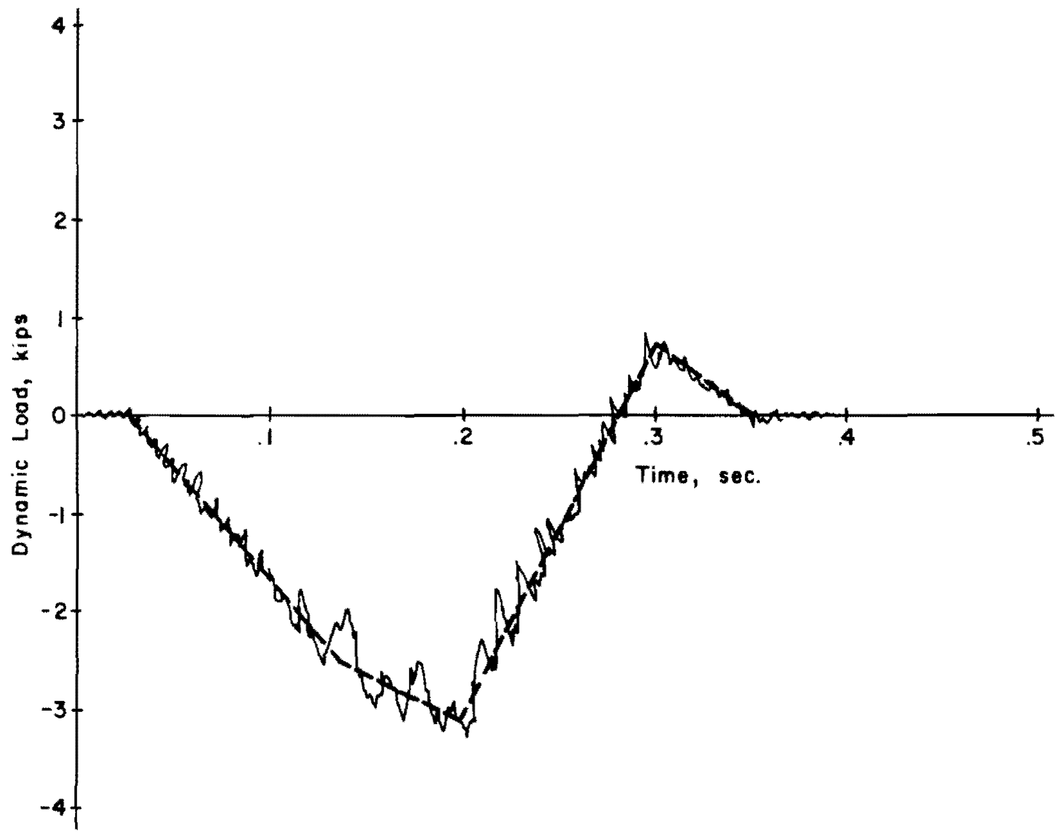
Each curve consists of 3 cards: a curve control card, a card listing foundation resistance, and a card giving corresponding deflections.

The nonlinear foundation can be described for a point, line, or an area, by the use of x and y-coordinates. Rules given in the discussion of Table 2 data apply to the distribution process.

The multipliers QMP and WMP are scaling factors for the resistance-deflection points given by LP and MP. The curve is constructed as QMP × LP for resistance values and WMP × MP for the deflection. Therefore QMP and WMP must not be zero.

The linear approximation (SFN) will be a positive number. This value cannot be omitted.

As many as 10 points can be used to define the nonlinear curve. When the symmetry option is requested, as many as 19 points can be generated.



Time increment = .001 sec

Time Station From K1	Thru K2	Cont. KONT	Load multiplier for a dynamic load of 1000 lb DQM
25		1	0.0
	135	1	-2.500E00
	200	1	-3.000E00
	300	1	0.750E00
	350	0	0.0

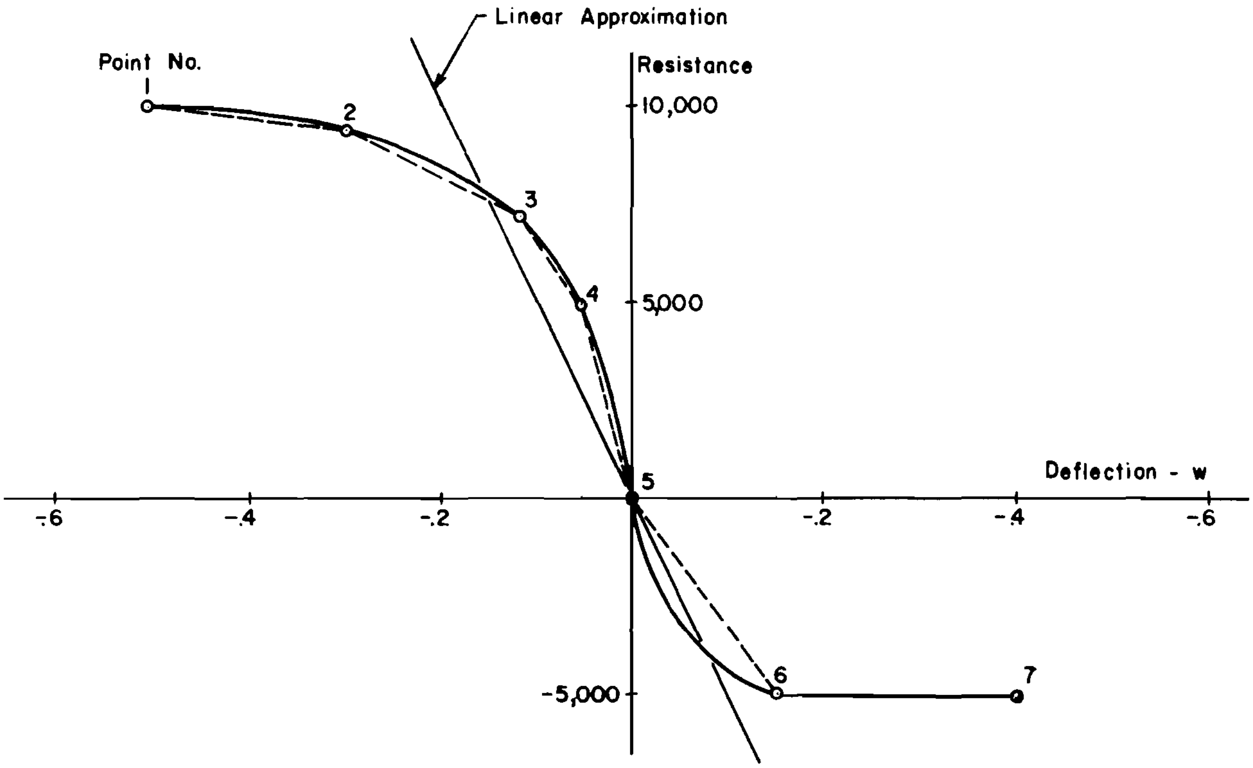
Fig D2. Example of organization of dynamic loading for data input.

The deflection must be input in algebraically increasing order. This can be accomplished by using a positive multiplier and algebraically increasing deflection points or a negative multiplier and decreasing deflections. When symmetry is required the initial resistance and deflection must be zero.

The load and deflection points (LP and MP) must be scaled integer values of the nonlinear resistance-deflection curve, with the scaling factors being the load and deflection multiplier values (QMP and WMP). An example illustrating the organization of the data is given in Fig D3.

The curves must be single-valued functions of deflection, i.e., for each deflection there is a unique load.

Cumulative input is available for the nonlinear curves and their linear approximations. The rules for distribution of both the curve and the linear approximation follow those given in Table 2; quarter values of the variables are assigned to corners of areas and half values to ends of lines and area edges.



	QMP	WMP	SFN
	1.000E+02	-1.000E-03	5.000E+04

Point Number	1	2	3	4	5	6	7
LP	100	90	71	50	0	-30	-50
MP	500	300	120	50	0	-150	-400

Fig D3. Example of organization of foundation resistance-deflection characteristics for data input.

APPENDIX E

SLAB 35 FLOW DIAGRAM AND PROGRAM LISTING

APPENDIX E. SLAB 35 FLOW DIAGRAM AND PROGRAM LISTING

The computer program SLAB 35 consists of the main driver program and 27 subroutines. Twelve of the subroutines were written especially for this program while the remaining 15 are part of a solution package for banded linear equations described elsewhere (Ref 6).

A summary flow diagram of the program SLAB 35 is shown in Fig E1. The major functions of the program are outlined in this figure. Detailed flow diagrams of the main flow program and the 12 subroutines unique to this program follow the summary flow diagram.

Five functions are controlled by the main program: data input and organization, output, nonlinear control, dynamic load generation, and equation generation and solution.

For the data input and organization phase, four subroutines are utilized. INTERP9 interprets data input tables and distributes the elastic and dynamic properties to plate node points, bars, and areas. STIF1 and STFMX construct and store on a disk file the static stiffness matrix. The first generates matrix terms related to bending stiffness and linear foundation springs and the second completes the formation with the addition of axial thrust and twisting stiffness to the coefficients developed by STIF1. STALD forms the sustained static load and dead loads and writes them on a disk file.

Nonlinear control is performed by a single subroutine, NONLIN4, which compares deflections of two iterations to determine if closure has been established. If the solution is not closed within the specified tolerance, a new correction load is computed and stored on a disk file.

At each time station, a new dynamic load is generated by DYNLD, which constructs either periodic or nonperiodic multipliers for stationary and moving load.

The generation and solution of equations are controlled by seven subroutines. Two, MASSAC and INERTIA, compute the right-hand side or load vector related to previous deflection or acceleration and velocity calculations.

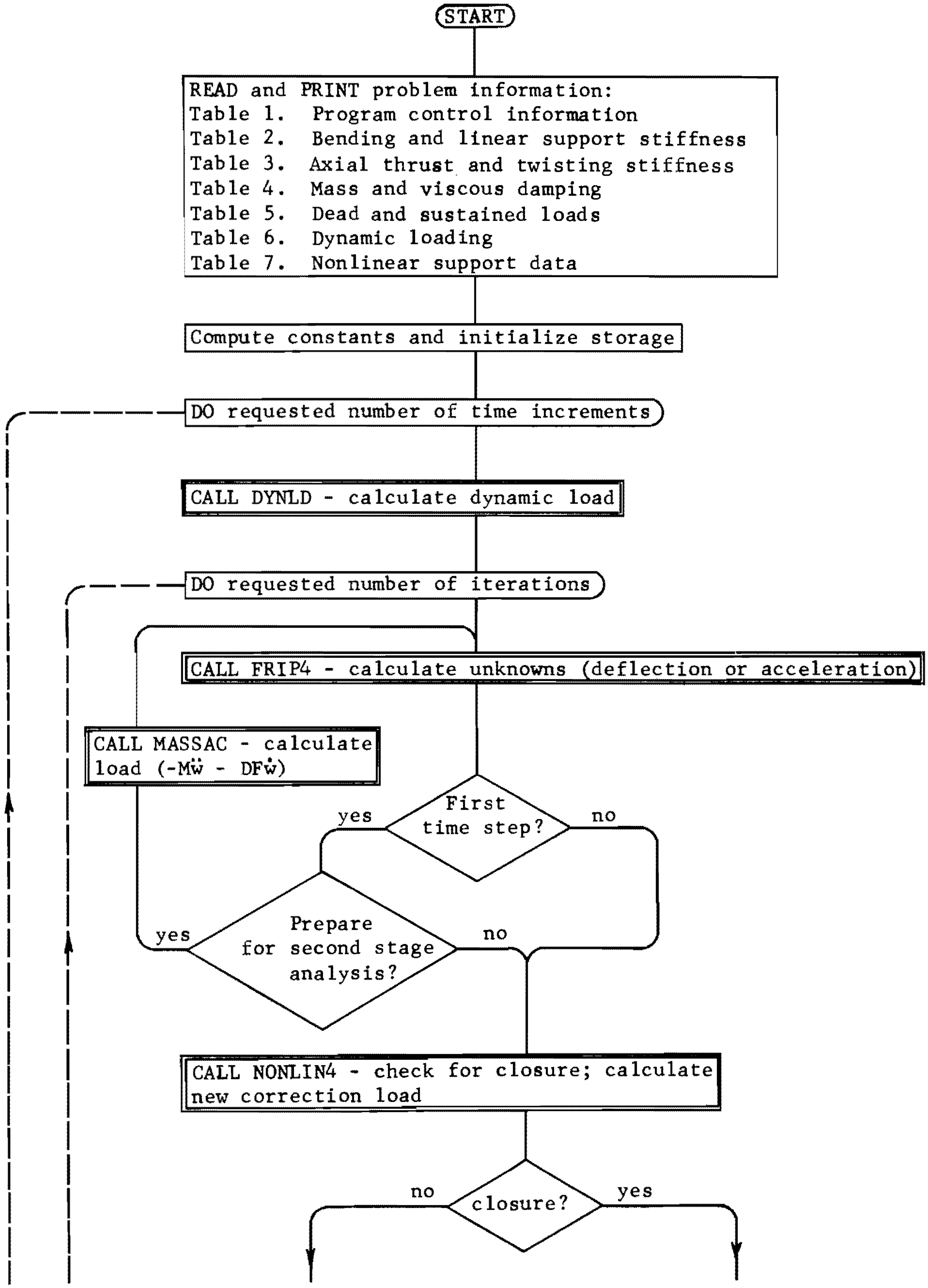


Fig E1. Summary flow diagram of program SLAB 35 (Continued).

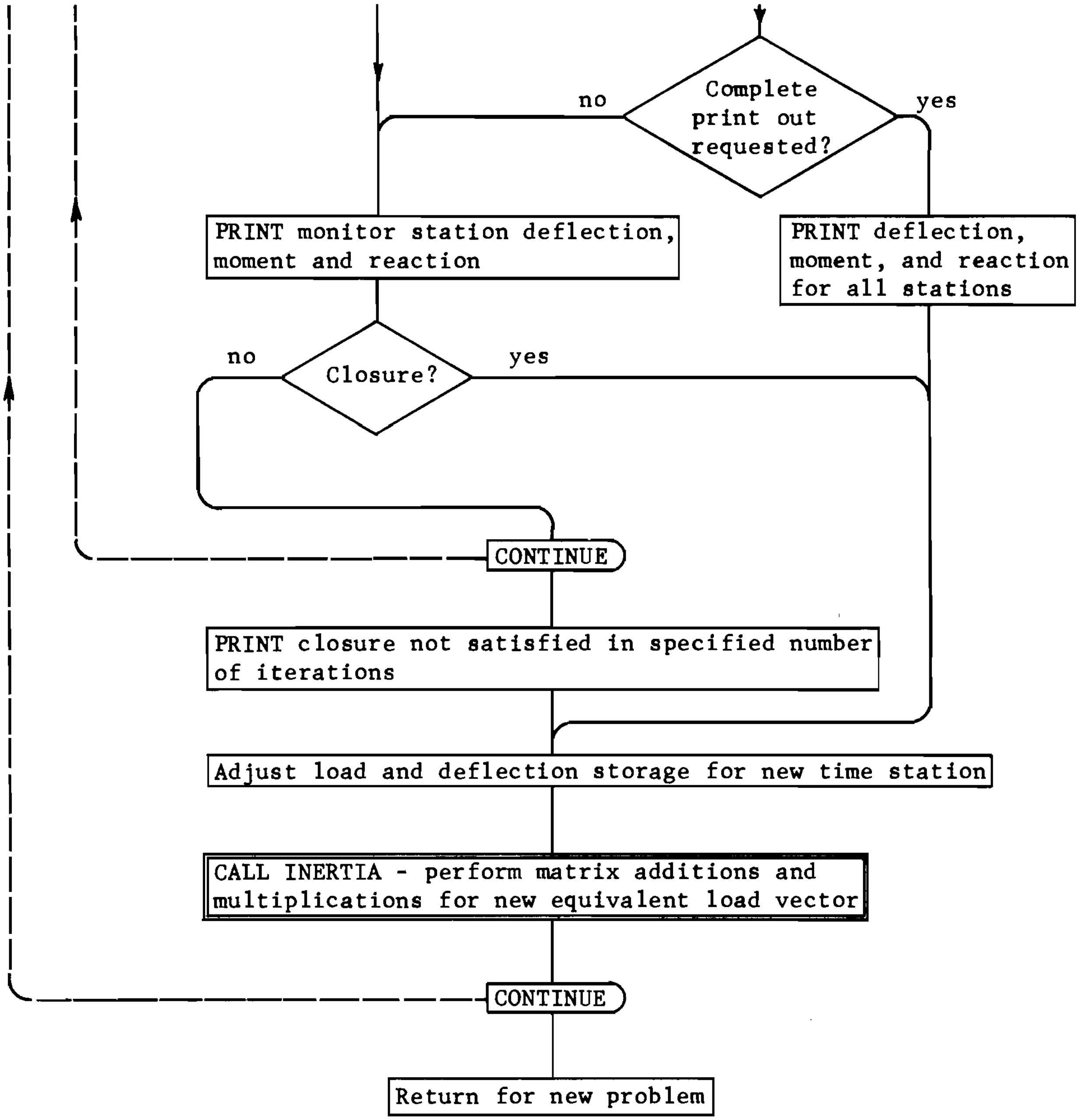


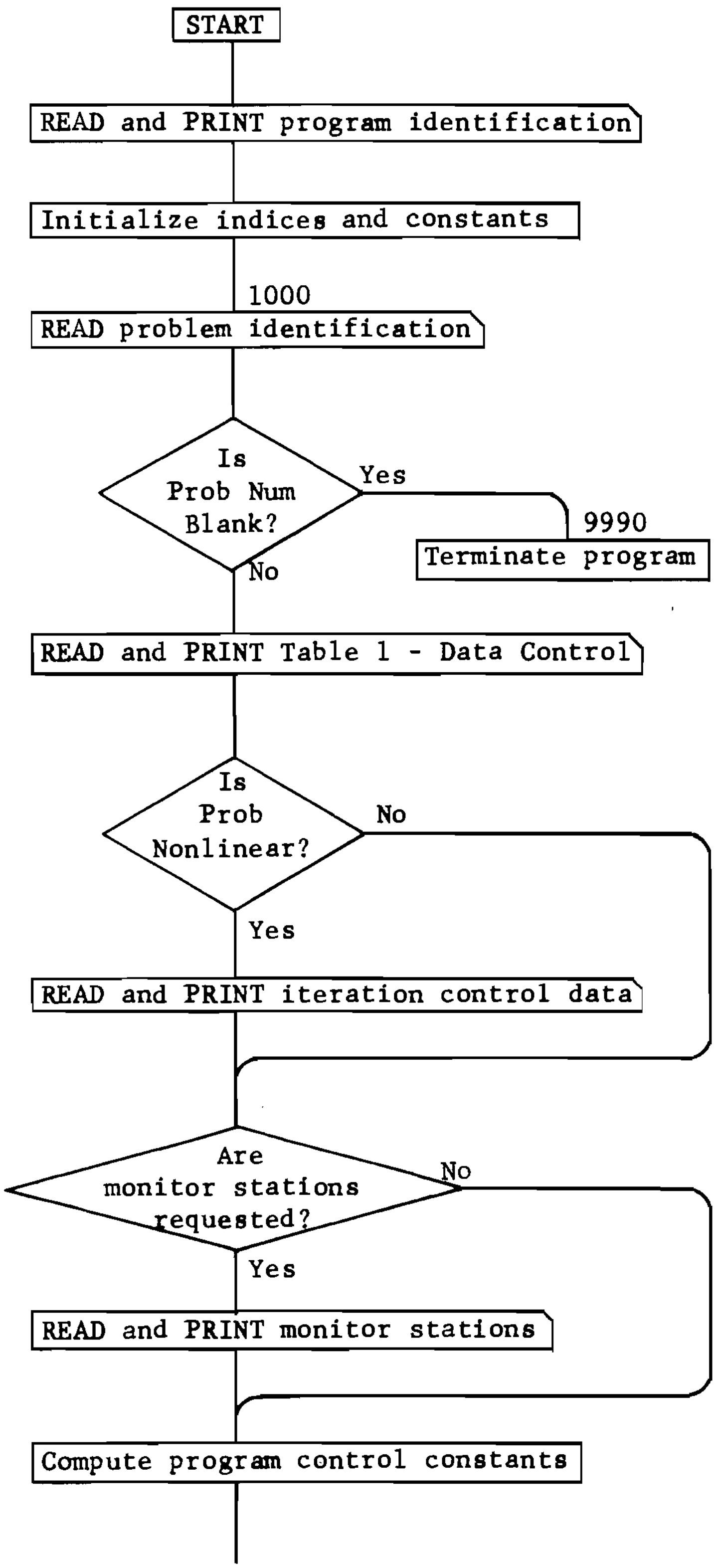
Fig E1. Summary flow diagram of program SLAB 35 (Continued).

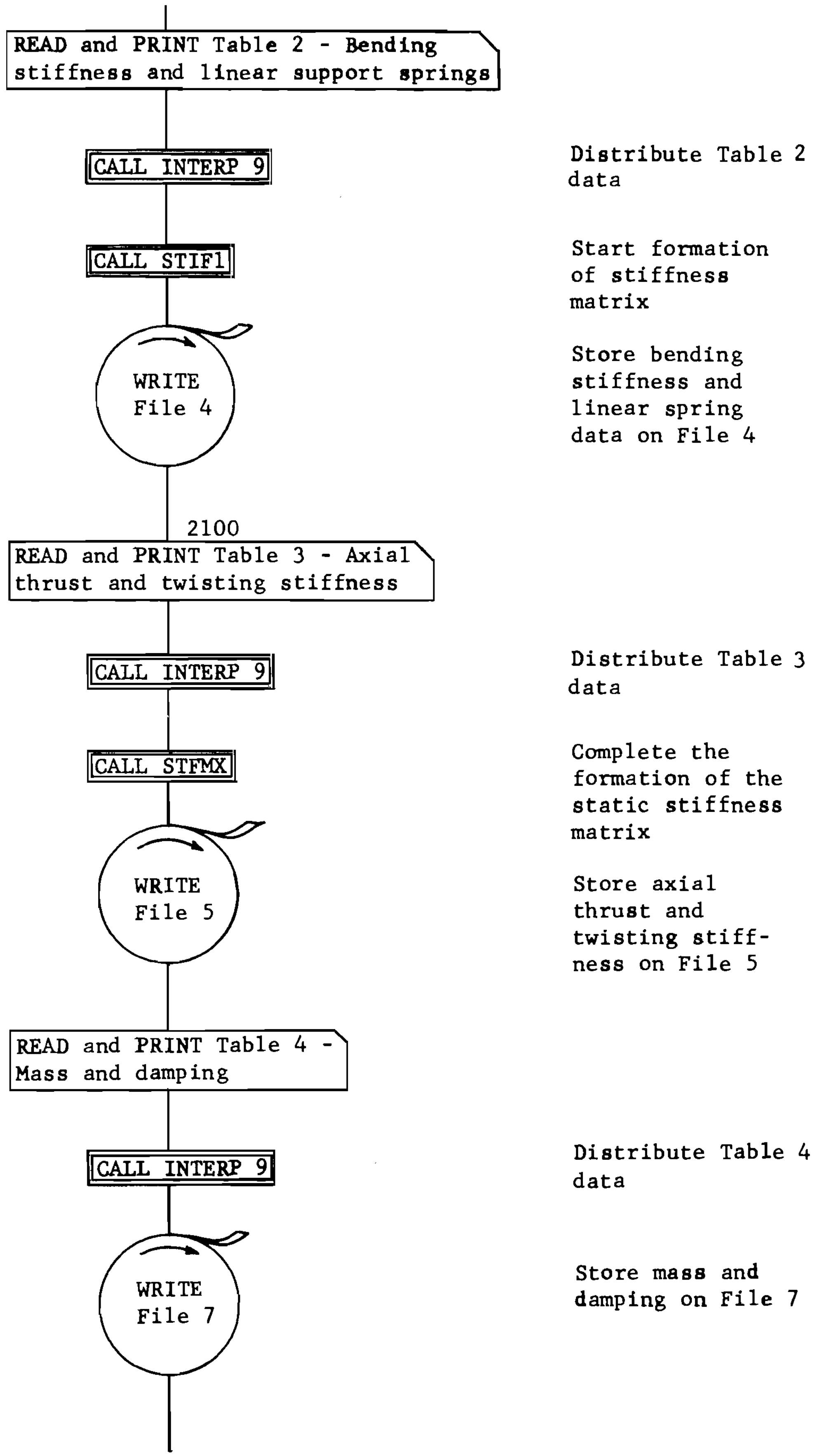
Three, STAT, DYNAM, and ACCEL, generate the equations and form the load vector, and EXCUT directs the solution by selecting the correct equation generator. The equations are solved by subroutine FRIP4 which has been described elsewhere (Ref 6) and therefore its flow diagram and listing are not included. MASSAC computes the products of mass times acceleration and velocity times damping for deflection analysis at the first time step. INERTIA computes the matrix products on the right-hand side of Eq 3.12. STAT forms the static stiffness matrix and constructs the load vector for either static analysis or deflection analysis at the first time step. DYNAM formulates the modified stiffness matrix and load vector for the general time step. ACCEL forms the modified stiffness matrix and load vector for acceleration analysis at the first time step. FRIP4 is an equation solver for matrices with five-wide banding.

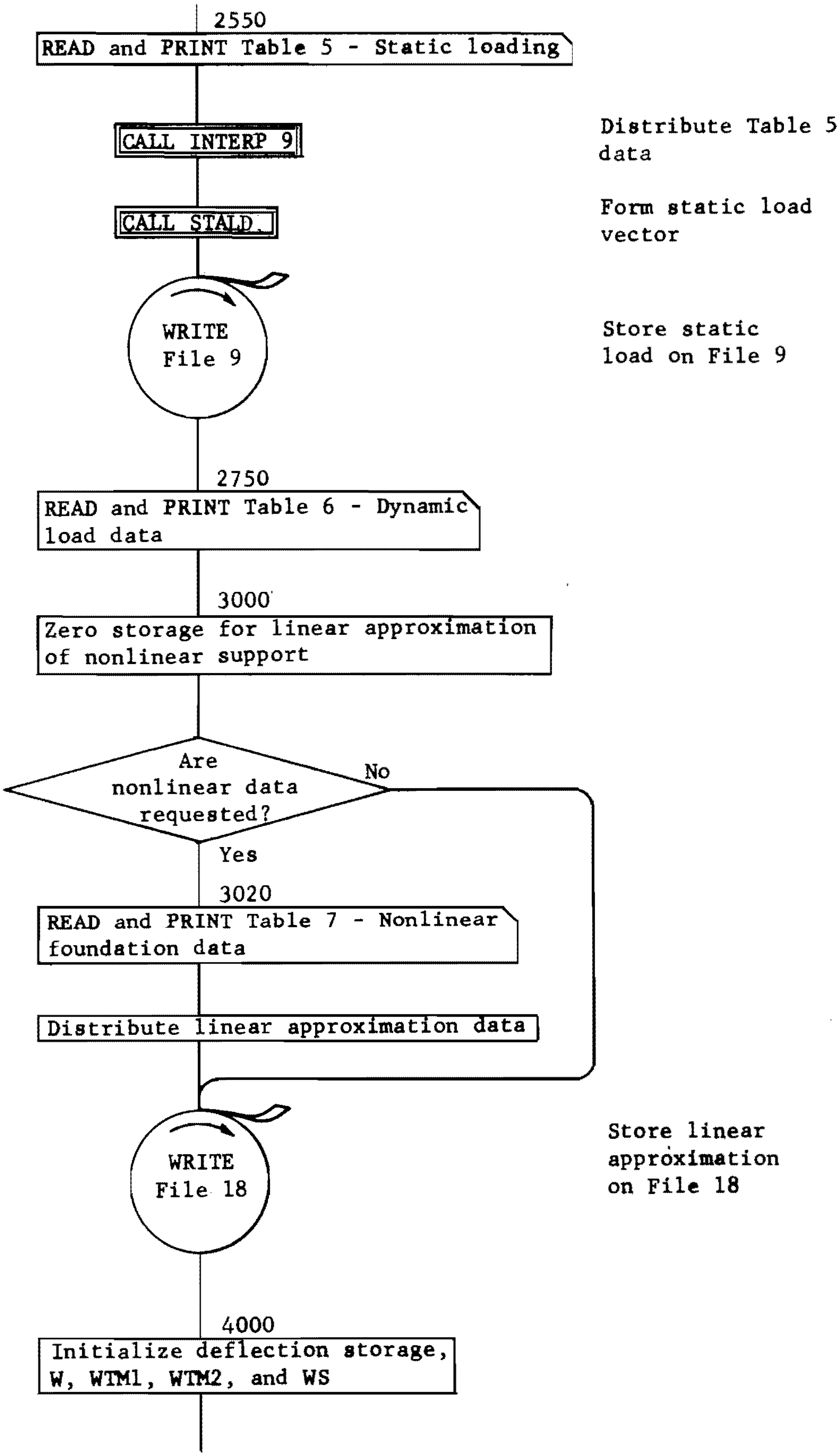
In addition to the subroutines noted above, 14 others are used throughout the program for matrix and vector operations (Ref 6).

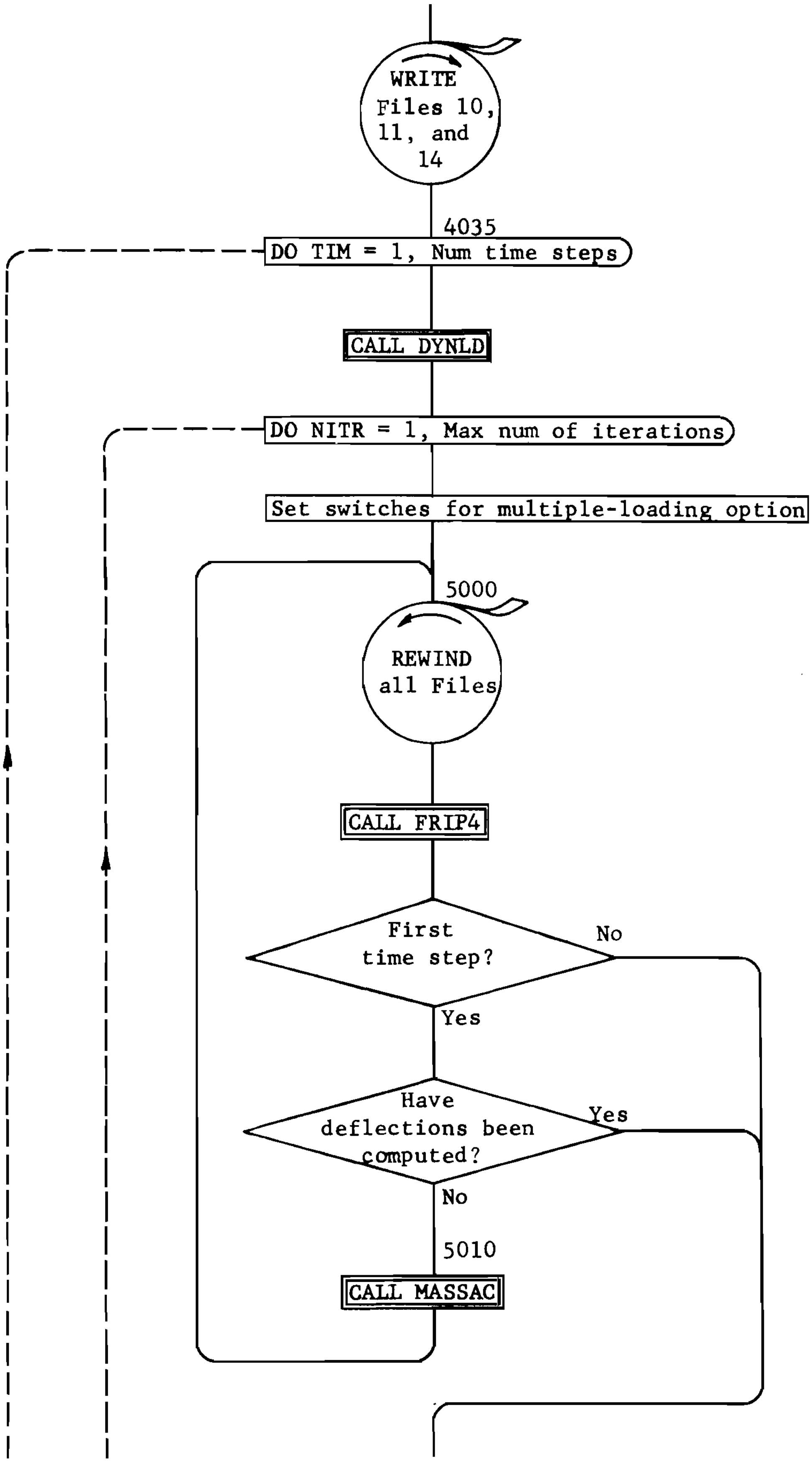
The output functions were coded in the main program. Printed results are moments, reactions, and deflections for either all node points or points selected by the program user.

FLOW DIAGRAM FOR PROGRAM SLAB 35




READ and PRINT Table 2 - Bending stiffness and linear support springs
CALL INTERP 9
Distribute Table 2 data
CALL STIF1
Start formation of stiffness matrix
WRITE File 4
Store bending stiffness and linear spring data on File 4
2100
READ and PRINT Table 3 - Axial thrust and twisting stiffness
CALL INTERP 9
Distribute Table 3 data
CALL STFMX
Complete the formation of the static stiffness matrix
WRITE File 5
Store axial thrust and twisting stiff- ness on File 5
READ and PRINT Table 4 - Mass and damping
CALL INTERP 9
Distribute Table 4 data
WRITE File 7
Store mass and damping on File 7


2550
READ and PRINT Table 5 - Static loading
CALL INTERP 9
Distribute Table 5 data
CALL STALD
Form static load vector
WRITE File 9
Store static load on File 9
2750
READ and PRINT Table 6 - Dynamic load data
3000
Zero storage for linear approximation of nonlinear support
Are nonlinear data requested?
No
Yes
3020
READ and PRINT Table 7 - Nonlinear foundation data
Distribute linear approximation data
WRITE File 18
Store linear approximation on File 18
4000
Initialize deflection storage, W, WTM1, WTM2, and WS



Initialize storage files for dynamic load, load correction, and equivalent load vectors

Start solution procedure

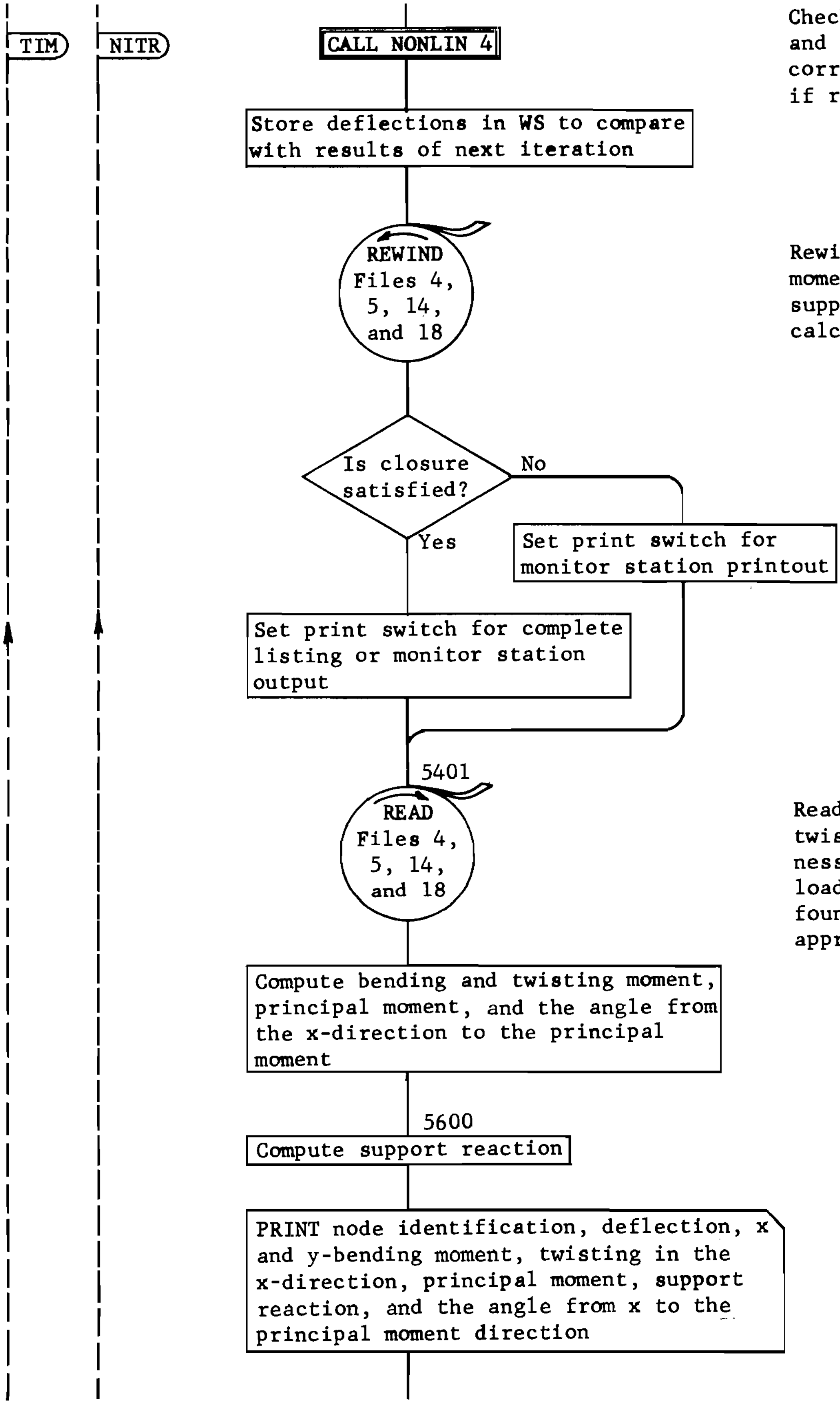
Compute dynamic loading at time step TIM

Files 6, 7, 8, and 10 through 18 are rewound for the solution procedure

Compute unknown deflections or accelerations. FRIP4 calls subroutine EXCUT which selects the coefficient matrix and load vector

Accelerations have been calculated at first time step

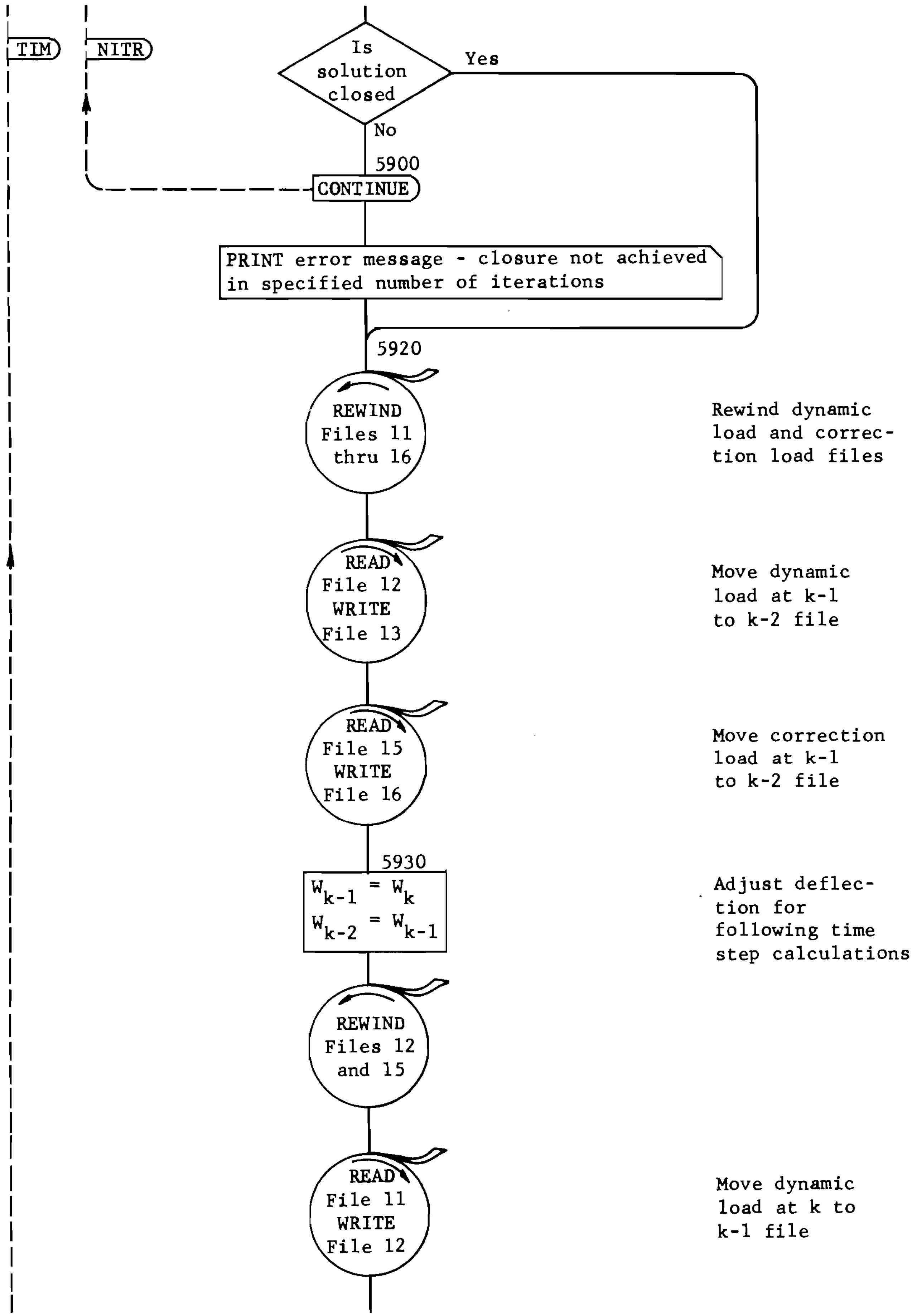
$(-M\ddot{w} - DF\dot{w})$ calculated and stored for deflection analysis

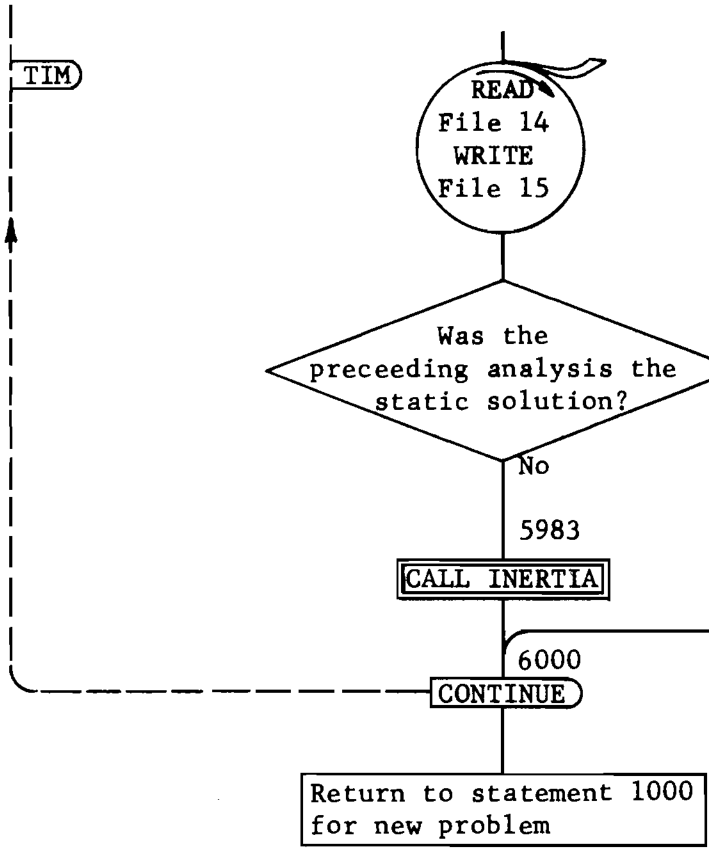


Check for closure and compute new correction load if required

Rewind files for moment and support reaction calculations

Read bending and twisting stiffness, correction load on linear foundation approximation


TIM
NITR
Is solution closed
Yes
No
5900
CONTINUE
PRINT error message - closure not achieved in specified number of iterations
5920
REWIND Files 11 thru 16
Rewind dynamic load and correction load files
READ File 12 WRITE File 13
Move dynamic load at k-1 to k-2 file
READ File 15 WRITE File 16
Move correction load at k-1 to k-2 file
5930
$W_{k-1} = W_k$
$W_{k-2} = W_{k-1}$
Adjust deflection for following time step calculations
REWIND Files 12 and 15
READ File 11 WRITE File 12
Move dynamic load at k to k-1 file



Move correction load at k to k-1 file

Compute equivalent load vector for following time step

```
      PROGRAM SLAB 35 ( INPUT, OUTPUT, TAPE1, TAPE2, TAPE3, TAPE4,       22JL9
     1                  TAPE5, TAPE6, TAPE7, TAPE8, TAPE9, TAPE10,      22JA9
     2                  TAPE11, TAPE12, TAPE13, TAPE14, TAPE15,         22JA9
     3          TAPE16, TAPE18                                          )16OC9
      DIMENSION       AN1( 32),       AN2( 14),      MSX( 10),          11NO8
     1                MSY( 10),       I1( 50),        J1( 50),          11NO8
     2                 I2( 50),       J2( 50),       DXN( 50),          11NO8
     3                DYN( 50),       SN( 50),       PXN( 50),          11NO8
     4                PYN( 50),      CTN( 50),      RHON( 50),          11NO8
     5                DFN( 50),       QN( 50),       TXN( 50),          11NO8
     6                TYN( 50),      NAM( 20),      JSFT( 20),          11NO8
     7               MSPD( 20),     JSYM( 20),       DQN( 20, 20),      16MY9
     8                IN1( 10),      JN1( 10),       IN2( 10),          11NO8
     9                JN2( 10),      SFN( 10),       NPC( 10),          11NO8
     A                 LP( 10),       MP( 10),       NDL( 20),          21MY9
     B                 K1( 20, 20),   K2( 20, 20), KONT( 20, 20),       11NO8
     C                DQM( 20, 20),  QNL( 10, 19),   WNL( 10, 19),      16MY9
     D                I1D( 20, 20),  J1D( 20, 20),   I2D( 20, 20),      16MY9
     E                J2D( 20, 20)                                      16MY9
      DIMENSION K11(20,20) , K22(20,20)                                 08JL0
C                                                                       17DE9
C * * * *  DIMENSIONED FPR A 4 X 15 SLAB                                17DE9
C *$ * * * * * * * *                                                    17DE9
      DIMENSION      DXF ( 11, 22), DYF( 11, 22), SSF( 11, 22),         17DE9
     1               PXF( 11, 22),  PYF( 11, 22),  RHOF( 11, 22),       17DE9
     2           DFF( 11, 22),  QF( 11, 22),  TXF( 11, 22),             17DE9
     3          TYF( 11, 22),  SFF( 11, 22),   W( 11, 22),              17DE9
     4        WTM1( 11, 22),  WTM2( 11, 22),   WS( 11, 22),             17DE9
     5    CTF(11, 22),  QD1F( 11, 22), QI1F( 11, 22)                    17DE9
      DIMENSION   A(7),  AM1( 7),  AM2( 7),  FF( 7),                    17DE9
     1    QI( 7),  QD1( 7),  QD2( 7),  QD3( 7),                         17DE9
     2     QI1( 7),  QI2( 7),  QI3( 7),  RHO( 7),                       17DE9
     3     DF( 7),  DX( 7),  DY( 7),  S( 7),                            17DE9
     4     EEM2( 7),  SF( 7),  AA( 7),  EEM1( 7),                       17DE9
     5     ATM( 7)                                                      17DE9
      DIMENSION    B( 7, 7),  BM1( 7, 7),  EP1( 7, 7),                  17DE9
     1     C( 7, 7),  CM1( 7, 7),  D( 7, 7),                            17DE9
     2    E( 7, 7),   ET2( 7, 1),  DT( 7, 3),                           17DE9
     3    CC( 7, 5),  ET1( 7, 1),   EE( 1, 7),                          17DE9
     4     SK( 7, 9),   BB( 7, 3),                                      17DE9
     5   DD( 7, 3),  DDM1( 7, 3),   CT( 7, 2)                           17DE9
      EQUIVALENCE ( WS, QI1F )                                          01NO9
      EQUIVALENCE ( DX, AA), ( DY, EE), ( S, EEM2)                      11NO8
      EQUIVALENCE ( DYF, PYF, DFF, TXF, WTM1 ), ( ATM, DT )             11NO8
      EQUIVALENCE ( DXF, PXF, RHOF, QF, SFF, QDIF, W )                  11NO8
      EQUIVALENCE ( SSF, CTF, TYF, WTM2 ), ( DXN, PXN, RHON, QN, MSPD ),23AG9
     1            ( DYN, PYN, DFN, TXN, JSFT), ( SN, CTN, TYN, NAM )    29JA9
      EQUIVALENCE (K1, K11), ( K2, K22 )                                02JL0
      COMMON/INCR/ MX, MY, MXP2, MXP3, MXP4, MXP5, MXP7,                11NO8
     1             MYP2, MYP3, MYP4, MYP5, MYP7, MT, HT, HY             20JU9
      COMMON/CON/  HXDHY3, HYDHX3, ODHXHY, ODHX, ODHY, PR, ODHT2, OD2HT,06JU9
     1             HXDHY, HYDHX                                         06JU9
      COMMON/RI/    NK, NL, NF, NT2SW, TIM                              22JA9
      TYPE INTEGER TIM, OP                                              17AP9
      TYPE REAL MSPD, KNTR                                              29JU0
      TYPE REAL K11, K22                                                02JL0
      TYPE REAL MOM3, MX3, MCK                                          18JU9
    1 FORMAT (   52H     PROGRAM SLAB 35 - MASTER DECK - A.E. KELLY     20JL9
     1          / 51H                REVISION DATE 29 JUL 70            )17DE9
   10 FORMAT ( )                                                        07JA9
   20 FORMAT (    5H     , 80X, 10HI-----TRIM )                         07JA9
   30 FORMAT (    5H1    , 80X, 10HI-----TRIM )                         07JA9
  100 FORMAT ( 16A5 )                                                   07JA9
  110 FORMAT ( A5, 5X, 14A5 )                                           07JA9
  120 FORMAT ( 5X, 6I5, 10X, I5, 5X, I5 )                               17AP9
  130 FORMAT ( 5X, 3I5, 4E10.3 )                                        07JA9
  140 FORMAT ( 5X, I5, 10X, 2E10.3 )                                    07JA9
  145 FORMAT ( 5X, 2I5 )                                                07JA9
  150 FORMAT ( 5X, 16A5 )                                               07JA9
  155 FORMAT (///10H     PROB , / 5X, A5, 5X, 14A5 )                    07JA9
  156 FORMAT (    17H     PROB ( CONTD), /  5X, A5, 5X, 14A5            )06JA9
  160 FORMAT (// 30H     TABLE 1.  CONTROL DATA    ,            /       07JA9
     1         / 30H           NUM CARDS TABLE 2    , 43X, I2,   /      07JA9
     2           30H           NUM CARDS TABLE 3    , 43X, I2,   /      07JA9
     3           30H           NUM CARDS TABLE 4    , 43X, I2,   /      07JA9
     4           30H           NUM CARDS TABLE 5    , 43X, I2,   /      07JA9
     5           30H           NUM CURVES TABLE 6   , 43X, I2,   /      07JA9
     6           30H           NUM CURVES TABLE 7   , 43X, I2           )07JA9
  165 FORMAT (    30H           NUM INCREMENTS   MX , 42X, I3,   /      07JA9
     1           30H           NUM INCREMENTS   MY , 42X, I3,   /       07JA9
     2           30H           NUM INCREMENTS   MT , 42X, I3,   /       07JA9
     3           30H           X INCR LENGTH    HX , 35X, E10.3,  /     07JA9
     4           30H           Y INCR LENGTH    HY , 35X, E10.3,  /     07JA9
     5           30H           TIME INCR LENGTH HT , 35X, E10.3,  /     07JA9
     6           30H           MAX POISSONS RATIO  , 35X, E10.3,  /     07JA9
     7           30H           PRINT OPTION     OP , 43X, I2,    /      21MY9
     8           51H             ALL DATA PRINTED EVERY OP TIME STEPS   ,17AP9
     D       /   30H           NUM MONITOR STATIONS, 43X, I2            )07JA9
  170 FORMAT (    30H           MAX NUM ITERATIONS  , 42X, I3,   /      07JA9
     1           30H           MAX ALLOWABLE DEFL  , 35X, E10.3,  /     07JA9
     2           30H           CLOSURE TOLERANCE   , 35X, E10.3         )07JA9
  175 FORMAT ( / 40H           LINEAR FOUNDATION SPRINGS                )07JA9
  180 FORMAT ( / 30H           MONITOR STATIONS     , /                 07JA9
     1           30H              X     Y                               )07JA9
  185 FORMAT ( 13X, I3, 3X, I3 )                                        07JA9
  190 FORMAT ( / 50H               MONITOR STATIONS NOT REQUESTED       )07JA9
  200 FORMAT ( 5X, 4I5, 5X, 3E10.3 )                                    07JA9
  210 FORMAT ( 5X, 5I5 )                                                16MY9
  220 FORMAT ( 5X, 3I5, E10.3 )                                         03JA9
  250 FORMAT ( //51H      TABLE 2.  ELASTIC STIFFNESS AND SUPPORT DATA  ,07JA9
     I         // 48H        FROM   THRU    DX           DY          S , /)07JA9
  260 FORMAT ( 5X, 2( 1X, I2, 1X, I3 ), 3E11.3 )                        07JA9
  270 FORMAT ( //51H      TABLE 3.  AXIAL FORCES AND TWISTING STIFFNESS ,07JA9
     I         // 48H        FROM   THRU    PX           PY          CT, /)07JA9
  275 FORMAT (    51H        NO AXIAL FORCE OR TWISTING STIFFNESS DATA  )03JA9
  280 FORMAT ( //45H      TABLE 4.  MASS AND DAMPING PROPERTIES    ,    03JA9
     1         // 40H        FROM   THRU    RHO          DF     ,      / )03JA9
  285 FORMAT (    51H        NO MASS OR DAMPING DATA - STATIC PROBLEM   )03JA9
  287 FORMAT ( //45H      TABLE 5.  STATIC LOADING ( DEAD LOAD )   ,    03JA9
     1         // 48H        FROM   THRU     Q           TX          TY,/ )03JA9
  290 FORMAT (    51H        NO STATIC LOADING - INITIAL DEFLECTION ZERO)03JA9
  291 FORMAT ( //45H      TABLE 6.  DYNAMIC LOADING                     )12JE9
```

```
292 FORMAT (3X,50H     NO DYNAMIC LOADING IN THIS PROBLEM           )21MY9
293 FORMAT ( / 17H        CURVE NO., I3,                     /     12JE9
    1         45H           NUM CARDS TO DEFINE CURVE          , I10, 12JE9
    2    /    45H           INITIAL SHIFT IN Y DIRECTION       , I10, 12JE9
    3    /    45H           SYMMETRY OPTION ( PERIODIC LOAD )  , I10, 12JE9
    4    / 5X, 52H          IF NOT ZERO, PERIODIC AMPLITUDE MULTIPLIER12JE9
    A   ,/    45H           NUM LOADS THIS CURVE               , I10, 12JE9
    5    /    37H           VELOCITY ( MILES PER HOUR ), 12X   ,F6.1 ,12JE9
    B    /    45H         TIME MULTIPLIER (CONVERTS INTEGER    ,E10.3,02JL0
    C    /    45H           INPUT TO A DECIMAL NUM )          ,      02JL0
    6    //   40H             TIME             LOAD       ,    /     02JL0
    7         40H         FROM   THRU  CONT  MULTIPLIER   ,    /    )03JA9
294 FORMAT ( 8X, I3, 5X, I3, 7X, E11.3 )                            03JA9
295 FORMAT ( 16X, I3, 4X, I1, 2X, E11.3 )                           03JA9
296 FORMAT ( 8X, I3, 12X, I1, 2X, E11.3 )                           03JA9
297 FORMAT ( / 51H * * * *  ERROR IN DATA INPUT - DYNAMIC LOADING   )03JA9
298 FORMAT ( / 40H        FROM   THRU      DYNAMIC LOAD   ,    / )  23JU9
299 FORMAT ( / 51H * * * *  ERROR IN DATA INPUT - STA OUT OF ORDER  )06JA9
300 FORMAT ( 5X, 4I5, 5X, 3E10.3, 3X, I2, 4X, I1 )                  06JA9
301 FORMAT    ( 5X, 2( 1X, I2, 1X, I3 ), 6X, E11.3                  )20JU9
310 FORMAT ( 30X, 10I5 )                                            06JA9
350 FORMAT ( //45H     TABLE 7.  NONLINEAR FOUNDATION CURVES   , /  )25MR9
355 FORMAT (    52H        FROM   THRU    Q-MULT      W-MULT     SPRIN06JA9
    1          25HG      POINTS   SYM OPT  ,  /                     06JA9
    2         5X, 2( 1X, I2, 1X, I3 ), 1X, 3E12.3, 6X, I2, 8X, I1, /  )06JA9
360 FORMAT (    15H        Q       ,   10I7                        )06JA9
370 FORMAT (    15H        W       ,   10I7                        )06JA9
380 FORMAT (    51H        LINEAR PROBLEM - NO NONLINEAR DATA       )06JA9
393 FORMAT (    50H * * * *  ERROR IN NONLINEAR DATA                )22JA9
550 FORMAT (    21H     FOR ITERATION NO, I3,                       07JL9
    1          10H THERE ARE, I5,                                    06JA9
    2          51H STATIONS NOT CLOSED WITHIN SPECIFIED TOLERANCE   )06JA9
552 FORMAT (    50H     DEFLECTIONS FALL OFF Q-W CURVE              )22JA9
554 FORMAT (    50H     COMPUTED DEFLECTIONS EXCEEDS MAX OF TABLE 1 )22JA9
560 FORMAT ( //25H     TABLE 7.  RESULTS    ,                  /    06JA9
    1          52H               X TWISTING MOMENT = - Y TWISTING MOME06JA9
    2          35HMT, - BETA ANGLES ARE CLOCKWISE    ,        //    06JA9
    3          52H                                                X 06JA9
    4          35H                LARGEST    BETA    ,         /    06JA9
    5          52H                           X           Y      TWIST06JA9
    6          35HNG   SUPPORT   PRINCIPAL   X TO    ,         /    06JA9
    7          52H      X , Y    DEFL      MOMENT     MOMENT     MOMEN06JA9
    8          35HT    REACTION   MOMENT   LARGEST   ,         /    )06JA9
561 FORMAT (    51H     RESULTS FOR STATIC AND DEAD LOAD            )06JA9
562 FORMAT (    28H     RESULTS AT TIME STATION, I5                 )06JA9
565 FORMAT (    24H     CLOSURE OBTAINED IN, I3,   15H ITERATIONS   )06JA9
580 FORMAT ( 5X, I2, 1X, I3, 6E11.3, F6.1  )                         12JE9
591 FORMAT (///50H     COLSURE NOT OBTAINED IN SPECIFIED ITERATIONS )30JA9
600 FORMAT (///50H     RETURN THIS AND FOLLOWING PAGE TO A E KELLY  )22JA9
           NCT2 = 0                                                 17AP9
           NCT3 = 0                                                 17AP9
           NCT4 = 0                                                 17AP9
           NCT5 = 0                                                 17AP9
           NCT6 = 0                                                 17AP9
           NCT7 = 0                                                 17AP9
           OP = 0                                                   17AP9
```

```
               MON = 0                                              17AP9
               MX = 0                                               17AP9
               MY = 0                                               17AP9
               MT = 0                                               17AP9
               HX = 0.0                                             17AP9
               HY = 0.0                                             17AP9
               HT = 0.0                                             17AP9
               PR = 0.0                                             17AP9
C * * *   START PROGRAM                                             01SE8
      PRINT 20                                                      22JA9
               ITEST = 5H                                           01SE8
      READ  100, ( AN1(N), N = 1, 32 )                              01SE8
      CALL TIC TOC (1)                                              01SE8
 1000 READ  110, NPROB, ( AN2(N), N = 1, 14 )                       01SE8
          IF ( NPROB  .EQ.  ITEST )      GO TO 9990                 01SE8
      PRINT 30                                                      22JA9
      PRINT 1                                                       01SE8
      PRINT 150, ( AN1(N), N = 1, 32 )                              01SE8
      PRINT 155, NPROB, ( AN2(N), N = 1, 14 )                       01SE8
C                                                                    1SE8
C * * *   INPUT TABLE 1                                             01SE8
C         PROGRAM CONTROL DATA                                      01SE8
      READ  120, NCT2, NCT3, NCT4, NCT5, NCR6, NCR7, OP,      MON   17AP9
      PRINT 160, NCT2, NCT3, NCT4, NCT5, NCR6, NCR7                 01SE8
      READ  130, MX, MY, MT, HX, HY, HT, PR                         01SE8
      PRINT 165, MX, MY, MT, HX, HY, HT, PR, OP,      MON           17AP9
          IF ( NCR7  .EQ.  0 )           GO TO 1050                 18OC8
      READ  140, ITMX, WMAX, TOL                                    18OC8
      PRINT 170, ITMX, WMAX, TOL                                    18OC8
          GO TO 1060                                                01SE8
 1050 PRINT 175                                                     01SE8
 1060 PRINT 180                                                      1SE8
          IF ( MON  .EQ.  0 )            GO TO 1100                 30JA9
          IF ( MON  .GT.  10 )           GO TO 9950                 18OC8
          DO 1080 L = 1, MON                                        01SE8
      READ  145, MSX(L), MSY(L)                                     01SE8
      PRINT 185, MSX(L), MSY(L)                                     01SE8
 1080     CONTINUE                                                  01SE8
          GO TO 1110                                                01SE8
C * * *   COMPUTE CONSTANTS AND PROGRAM CONTROL INDICES             01SE8
 1100 PRINT 190                                                      1SE8
 1110          MXP7 = MX + 7                                        01SE8
               MYP7 = MY + 7                                        01SE8
               MXP5 = MX + 5                                        01SE8
               MYP5 = MY + 5                                        01SE8
               MXP4 = MX + 4                                        18OC8
               MYP4 = MY + 4                                        18OC8
               MXP3 = MX + 3                                        01SE8
               MYP3 = MY + 3                                        01SE8
               MXP2 = MX + 2                                        28JA9
               MYP2 = MY + 2                                        28JA9
               MXP1 = MX + 1                                        17AP9
               MYP1 = MY + 1                                        17AP9
               IPL  = 3                                             01SE8
               ODHXHY = 1.0 / ( HX * HY )                           22JA9
               HXDHY  = HX / HY                                     01SE8
```

```
          HYDHX  = HY / HX                                              01SE8
          HXDHY3 = HX / ( HY**3 )                                       22JA9
          HYDHX3 = HY / ( HX**3 )                                       22JA9
          ODHT2  = 1.0 / ( HT**2 )                                      22JA9
          OD2HT  = 1.0 / ( 2.0 * HT )                                   18OC8
          ODHX2 = ODHXHY * HYDHX                                        16MY9
          ODHY2 = ODHXHY * HXDHY                                        16MY9
          ODHX = 1.0 / HX                                               12JU9
          ODHY = 1.0 / HY                                               12JU9
C                                                                        1SE8
          L1 = MXP3                                                     01SE8
          L2 = MXP7                                                     01SE8
          L3 = MYP7                                                     01SE8
          NF = 3                                                        01SE8
          NK = MXP3                                                     01SE8
          NL = MYP5                                                     25MR9
          N1 = 1                                                        01SE8
          N2 = 3                                                        01SE8
          N3 = 5                                                        01SE8
C                                                                        1SE8
C                                                                        1SE8
C * * *   SECTION 2000 - DATA INPUT                                     01SE8
C                   ELASTIC PROPERTIES                                  01SE8
C                                                                        1SE8
      CALL IOBIN(6HREWIND,4)                                            19AG9I0
      CALL IOBIN(6HREWIND,5)                                            19AG9I0
      CALL IOBIN(6HREWIND,7)                                            19AG9I0
      PRINT 250                                                         01SE8
         IF ( NCT2  .EQ.  0 )           GO TO 9950                      22JA9
         IF ( NCT2  .GT.  50 )          GO TO 9950                      18OC8
         DO 2050 L = 1, NCT2                                            01SE8
      READ  200, I1(L), J1(L), I2(L), J2(L), DXN(L), DYN(L), SN(L)      01SE8
      PRINT 260, I1(L), J1(L), I2(L), J2(L), DXN(L), DYN(L), SN(L)      01SE8
 2050    CONTINUE                                                       01SE8
      CALL INTERP 9 ( I1, J1, I2, J2, DXN, NCT2, DXF, 1, 0, 0, L2, L3,  05JU9
     1               0 , 0  )                                           05JU9
      CALL INTERP 9 ( I1, J1, I2, J2, DYN, NCT2, DYF, 1, 0, 0, L2, L3,  05JU9
     1               0 , 0  )                                           05JU9
      CALL INTERP 9 ( I1, J1, I2, J2,  SN, NCT2, SSF, 1, 0, 0, L2, L3,  05JU9
     1               0 , 0  )                                           05JU9
C                                                                        1SE8
C * * *   FORM MATRIX COEFFICIENTS RELATED TO DX, DY, AND S VALUES      01SE8
C                                                                        1SE8
      CALL STIF1 ( DXF, DYF, SSF, SK , L1, L2, L3 )                     18OC8
C                                                                        1SE8
C * * *   STORE DX AND DY ON DISK FILES                                 01SE8
C                                                                        1SE8
             MTK = MXP3                                                 19AG9I0
         DO 2100 J = 3, MYP5                                            16MY9
      CALL IOBIN(5HWRITE ,4,DXF(3,J),MTK)                               25AG9I0-
      CALL IOBIN(5HWRITE ,4,DYF(3,J),MTK)                               25AG9I0-
      CALL IOBIN(5HWRITE ,4,SSF(3,J),MTK)                               25AG9I0-
 2100    CONTINUE                                                       18OC8
      CALL IOBIN (6HWRITER,4 )                                          22AG9I0
C                                                                        1SE8
C * * *   READ TWISTING STIFFNESS AND IN-PLANE FORCES - SECTION 2200    01SE8
```

```
C                                                                        1SE8
      PRINT 270                                                         01SE8
         IF ( NCT3  .EQ.  0 )           GO TO 2300                      01SE8
         IF ( NCT3  .GT.  50 )          GO TO 9950                      18OC8
         DO 2250 L = 1, NCT3                                            01SE8
      READ  200, I1(L), J1(L), I2(L), J2(L), PXN(L), PYN(L), CTN(L)     18OC8
      PRINT 260, I1(L), J1(L), I2(L), J2(L), PXN(L), PYN(L), CTN(L)     01SE8
 2250    CONTINUE                                                       01SE8
 2300 CALL INTERP 9 ( I1, J1, I2, J2, PXN, NCT3, PXF, 0, 1, 0, L2, L3,  05JU9
     1               1 , 0 )                                            05JU9
      CALL INTERP 9 ( I1, J1, I2, J2, PYN, NCT3, PYF, 0, 1, 0, L2, L3,  05JU9
     1               0 , 1 )                                            05JU9
      CALL INTERP 9 ( I1, J1, I2, J2, CTN, NCT3, CTF, 0, 0, 1, L2, L3,  24JU9
     1               0 , 0  )                                           05JU9
         IF ( NCT3  .EQ.  0 )           PRINT 275                       01SE8
      CALL STFMX ( PXF, PYF, CTF,      SK, L1, L2, L3, NCT3 )           29JA9
         DO 2350 J = 3, MYP5                                            16MY9
      CALL IOBIN(5HWRITE ,5,PXF(3,J),MTK)                               25AG9I0-
      CALL IOBIN(5HWRITE ,5,PYF(3,J),MTK)                               25AG9I0-
      CALL IOBIN(5HWRITE ,5,CTF(3,J),MTK)                               25AG9I0-
 2350    CONTINUE                                                       18OC8
      CALL IOBIN (6HWRITER,5 )                                          22AG9I0
      CALL IOBIN(6HREWIND,9)                                            19AG9I0
C                                                                        1SE8
C * * *   READ MASS AND DAMPING PROPERTIES - SECTION 2400               01SE8
C                                                                        1SE8
      PRINT 280                                                         01SE8
         IF ( NCT4  .EQ.  0 )           GO TO 2500                      01SE8
         IF ( NCT4  .GT.  50 )          GO TO 9950                      18OC8
         DO 2450 L = 1, NCT4                                            01SE8
      READ  200, I1(L), J1(L), I2(L), J2(L), RHON(L), DFN(L)            01SE8
      PRINT 260, I1(L), J1(L), I2(L), J2(L), RHON(L), DFN(L)            01SE8
 2450    CONTINUE                                                       01SE8
 2500 CALL INTERP 9 ( I1, J1, I2, J2, RHON, NCT4, RHOF, 1, 0, 0, L2, L3,27JE9
     1               0 , 0  )                                           05JU9
      CALL INTERP 9 ( I1, J1, I2, J2,  DFN, NCT4, DFF, 1, 0, 0, L2, L3, 27JE9
     1               0 , 0  )                                           05JU9
         IF ( NCT4  .EQ.  0 )           PRINT 285                       01SE8
         DO 2550 J = 3, MYP5                                            18OC8
      CALL IOBIN(5HWRITE ,7,RHOF(3,J),MTK)                              25AG9I0-
      CALL IOBIN(5HWRITE ,7,DFF(3,J),MTK)                               25AG9I0-
 2550    CONTINUE                                                       18OC8
      CALL IOBIN (6HWRITER,7 )                                          22AG9I0
C                                                                        1SE8
C * * *   READ STATIC ( DEAD ) LOAD AND FORM STATIC LOAD VECTOR         01SE8
C                                                                        1SE8
      PRINT 287                                                         01SE8
         IF ( NCT5  .EQ.  0 )           GO TO 2700                      01SE8
         IF ( NCT5  .GT.  50 )          GO TO 9950                      18OC8
         DO 2650 L = 1, NCT5                                            01SE8
      READ  200, I1(L), J1(L), I2(L), J2(L), QN(L), TXN(L), TYN(L)      01SE8
      PRINT 260, I1(L), J1(L), I2(L), J2(L), QN(L), TXN(L), TYN(L)      01SE8
 2650    CONTINUE                                                       01SE8
 2700 CALL INTERP 9 ( I1, J1, I2, J2,  QN, NCT5, QF, 1, 0, 0, L2, L3,   05JU9
     1               0 , 0  )                                           05JU9
      CALL INTERP 9 ( I1, J1, I2, J2, TXN, NCT5,TXF, 1, 0, 0, L2, L3,   05JU9
```

```
     1                    0 , 0  )                                          05JU9
      CALL INTERP 9 ( I1, J1, I2, J2, TYN, NCT5,TYF, 1, 0, 0, L2, L3,       05JU9
     1                    0 , 0  )                                          05JU9
          IF ( NCT5  .EQ.  0 )             PRINT 290                        01SE8
      CALL STALD  ( QF, TXF, TYF, FF, L1, L2, L3 )                          04MR9
               MKL = MXP1                                                   19AG910
          DO 2750 J = 4, MYP4                                               17AP9
      CALL IOBIN(5HWRITE ,9, QF(4,J),MKL)                                   25AG910-
      CALL IOBIN(5HWRITE ,9,TXF(4,J),MKL)                                   25AG910-
      CALL IOBIN(5HWRITE ,9,TYF(4,J),MKL)                                   25AG910-
 2750     CONTINUE                                                          18OC8
      CALL IOBIN (6HWRITER,9 )                                              22AG910
C                                                                            1SE8
C * * *   READ AND STORE DYNAMIC LOADING IN CORE                            01SE8
C              TABLE 6                                                      01SE8
C                                                                            1SE8
      PRINT 291                                                             01SE8
          IF ( NCR6  .EQ.  0 )             GO TO 2995                       25MR9
          IF ( NCR6  .GT.  20 )            GO TO 9950                       18OC8
          DO 2950 L = 1, NCR6                                               01SE8
      READ  200,  NAM(L), JSFT(L), JSYM(L), NDL(L), MSPD(L), TMPL           02JL0
      PRINT 293,  L, NAM(L), JSFT(L), JSYM(L), NDL(L), MSPD(L), TMPL        02JL0
               NM = NAM(L)                                                  01SE8
          IF ( NM    .GT.  20 )            GO TO 9950                       18OC8
               KSW = 0                                                      01SE8
               KSX = 0                                                      01SE8
               KS1 = 0                                                      01SE8
          DO 2940 M = 1, NM                                                 01SE8
      READ 220, K1(L,M), K2(L,M), KONT(L,M), DQM(L,M)                       01SE8
               KSX = KSW                                                    01SE8
               KSW = 2 * KS1 + 1 + KONT(L,M)                                01SE8
               KS1 = KONT(L,M)                                              01SE8
C                                                                            1SE8
C * * *   CHECK ORDER OF TIME STATIONS AND PERIODIC LOAD MULTIPLIER         01SE8
C                                                                            1SE8
 2800     GO TO ( 2810, 2940, 2830, 2840 ), KSW                             01SE8
 2810     IF ( K2(L,M)  .LT.  K1(L,M) ) GO TO 2980                          01SE8
      PRINT 294, K1(L,M), K2(L,M), DQM(L,M)                                 01SE8
          GO TO 2940                                                        01SE8
 2830     IF ( KSX  .EQ.  2 )              GO TO 2835                       01SE8
          IF ( K2(L,M) .LT. K2(L,M-1) ) GO TO 2980                          01SE8
      PRINT 295, K2(L,M-1), KONT(L,M-1), DQM(L,M-1)                         01SE8
      PRINT 295, K2(L,M),   KONT(L,M),   DQM(L,M)                           01SE8
          GO TO 2940                                                        01SE8
 2835     IF ( K2(L,M) .LT. K1(L,M-1) ) GO TO 2980                          01SE8
      PRINT 296, K1(L,M-1), KONT(L,M-1), DQM(L,M-1)                         01SE8
      PRINT 295, K2(L,M),   KONT(L,M),   DQM(L,M)                           01SE8
          GO TO 2940                                                        01SE8
 2840     IF ( KSX  .EQ.  2 )              GO TO 2845                       01SE8
          IF ( K2(L,M) .LT. K2(L,M-1) ) GO TO 2980                          01SE8
      PRINT 295, K2(L,M-1), KONT(L,M-1), DQM(L,M-1)                         01SE8
          GO TO 2940                                                        01SE8
 2845     IF ( K2(L,M) .LT. K1(L,M-1) ) GO TO 2980                          01SE8
      PRINT 296, K1(L,M-1), KONT(L,M-1), DQM(L,M-1)                         01SE8
 2940     CONTINUE                                                          01SE8
          DO 2942 MM = 1, NM                                                02JL0
                 K11(L,MM)  =  K1(L,MM) * TMPL                              02JL0
                 K22(L,MM)  =  K2(L,MM) * TMPL                              02JL0
 2942     CONTINUE                                                          02JL0
C                                                                            1SE8
          IF ( JSYM(L)  .NE.  1 )        GO TO 2900                         19SE9
          IF ( DQM(L,1)  .EQ.  DQM(L,NM) )  GO TO 2900                      19SE9
      PRINT 297                                                             01SE8
          GO TO 9990                                                        01SE8
 2900 PRINT 298                                                             19SE9
          ND = NDL(L)                                                       16MY9
          DO 2945  M = 1, ND                                                16MY9
      READ  200, I1D(L,M), J1D(L,M), I2D(L,M), J2D(L,M), DQN(L,M)           16MY9
          IF ( I2D(L,M)  .LT.  I1D(L,M) )      GO TO 2980                   16MY9
          IF ( J2D(L,M)  .LT.  J1D(L,M) )      GO TO 2980                   16MY9
          IF ( I1D(L,M)  .LT.  0 )         GO TO 2980                       28MY9
          IF ( I2D(L,M)  .GT.  MX )        GO TO 2980                       28MY9
      PRINT 301, I1D(L,M) , J1D(L,M) , I2D(L,M) , J2D(L,M) , DQN(L,M)       23JE9
 2945     CONTINUE                                                          16MY9
 2950     CONTINUE                                                          01SE8
          GO TO 3000                                                        01SE8
 2980 PRINT 299                                                             01SE8
          GO TO 9990                                                        01SE8
 2995 PRINT 292                                                             25MR9
C                                                                            1SE8
C * * *   INPUT TABLE 7 - NONLINEAR Q-W FOUNDATION CURVES                   01SE8
C                                                                            1SE8
 3000 PRINT 350                                                              1SE8
      CALL IOBIN(6HREWIND,18)                                               19AG910
          DO 3020  J = 1, MYP7                                              01SE8
          DO 3010  I = 1, MXP7                                              01SE8
               SFF(I,J) = 0.0                                               19OC8
 3010     CONTINUE                                                          01SE8
 3020     CONTINUE                                                          01SE8
          IF ( NCR7  .EQ.  0 )             GO TO 3700                       01SE8
          IF ( NCR7  .GT.  10 )            GO TO 3995                       19OC8
          DO 3600 L = 1, NCR7                                               01SE8
      READ 300, IN1(L), JN1(L), IN2(L), JN2(L), QMP, WMP,                   01SE8
     1              SFN(L), NPC(L), ISYM                                    01SE8
C * * *   CHECK STATION ORDER                                               01SE8
C                                                                            1SE8
          IF ( IN2(L)  .LT. IN1(L) )       GO TO 3990                       01SE8
          IF ( JN2(L)  .LT. JN1(L) )       GO TO 3990                       01SE8
      PRINT 355, IN1(L), JN1(L), IN2(L), JN2(L), QMP, WMP,                  01SE8
     1              SFN(L), NPC(L), ISYM                                    01SE8
               NP = NPC(L)                                                  01SE8
          IF ( NP   .GT.  10 )             GO TO 3995                       19OC8
      READ  310, ( LP(J), J = 1, NP )                                       01SE8
      PRINT 360, ( LP(J), J = 1, NP )                                       01SE8
      READ  310, ( MP(J), J = 1, NP )                                       01SE8
      PRINT 370, ( MP(J), J = 1, NP )                                       01SE8
          IF ( ISYM  .EQ.  0 )             GO TO 3100                       01SE8
C                                                                            1SE8
C * * *   FORM SYMMETRIC Q-W CURVES                                         01SE8
C                                                                            1SE8
          IF ( LP(1) )                     3995, 3030, 3995                 01SE8
 3030     IF ( MP(1) )                     3995, 3040, 3995                 01SE8
```

```
 3040          NPC(L) = 2 * NPC(L) - 1                              01SE8
               NPS = NPC(L)                                         01SE8
         DO 3080 M = NP, NPS                                        01SE8
               N = M - NP + 1                                       01SE8
               QNL(L,M) = QMP * LP(N)                               01SE8
               WNL(L,M) = WMP * MP(N)                               01SE8
         IF ( QNL(L,M)  .GT.  0.0 )      GO TO 3995                 01SE8
         IF ( WNL(L,M)  .LT.  0.0 )      GO TO 3995                 01SE8
         IF ( M  .EQ.  NP )              GO TO 3080                 01SE8
         IF ( MP(N) - MP(N-1) )          3995, 3995, 3060           01SE8
 3060          N = NPS + 1 - M                                      01SE8
               QNL(L,N) = - QNL(L,M)                                01SE8
               WNL(L,N) = - WNL(L,M)                                01SE8
 3080    CONTINUE                                                   01SE8
         GO TO 3400                                                 01SE8
C                                                                    1SE8
C * * *  INPUT GENERAL Q-W CURVE                                    01SE8
C                                                                    1SE8
 3100    DO 3200 M = 1, NP                                          01SE8
               QNL(L,M) = QMP * LP(M)                               01SE8
               WNL(L,M) = WMP * MP(M)                               01SE8
         IF ( M  .EQ.  1 )               GO TO 3200                 01SE8
         IF ( WNL(L,M) - WNL(L,M-1) )    3995, 3995, 3200           01SE8
 3200    CONTINUE                                                   01SE8
C                                                                    1SE8
C                                                                    1SE8
C * * *  FORM INITIAL ELASTIC SPRING (SF) FOR NONLINEAR CURVES      01SE8
C                                                                    1SE8
 3400          ISW = 0                                              01SE8
               JSW = 0                                              01SE8
               I11 = IN1(L) + 4                                     30JA9
               I22 = IN2(L) + 4                                     30JA9
               J11 = JN1(L) + 4                                     30JA9
               J22 = JN2(L) + 4                                     30JA9
         IF ( I22  .GT.  I11 )            ISW = 1                   30JA9
         IF ( J22  .GT.  J11 )            JSW = 1                   30JA9
         DO 3500 I = I11, I22                                       30JA9
         DO 3480 J = J11, J22                                       30JA9
               CMX = 1.0                                            01SE8
               CMY = 1.0                                            01SE8
         IF ( ISW  .EQ.  0 )             GO TO 3440                 01SE8
         IF ( JSW  .EQ.  0 )             GO TO 3430                 01SE8
         IF ( J  .EQ.  J11 )             CMY = 0.5                  30JA9
         IF ( J  .EQ.  J22 )             CMY = 0.5                  30JA9
         IF ( I  .EQ.  I11 )             CMX = 0.5                  30JA9
         IF ( I  .EQ.  I22 )             CMX = 0.5                  30JA9
         GO TO 3450                                                 01SE8
C * * *  LINE SPRING IN X DIRECTION                                 01SE8
 3430    IF ( I  .EQ.  I11 )             CMX = 0.5                  30JA9
         IF ( I  .EQ.  I22 )             CMX = 0.5                  30JA9
         GO TO 3450                                                 01SE8
 3440    IF ( JSW  .EQ.  0 )             GO TO 3450                 01SE8
C * * *  LINE SPRING IN Y DIRECTION                                 01SE8
         IF ( J  .EQ.  J11 )             CMY = 0.5                  30JA9
         IF ( J  .EQ.  J22 )             CMY = 0.5                  30JA9
 3450          CON = CMX * CMY                                      01SE8
```

```
                  SFF(I,J) = CON * SFN(L) + SFF(I,J)                190C8
 3480       CONTINUE                                                01SE8
 3500       CONTINUE                                                01SE8
 3600       CONTINUE                                                01SE8
 3680       CONTINUE                                                16MY9
            DO 3690 J = 3, MYP5                                     190C8
       CALL IOBIN(5HWRITE ,18,SFF(3,J),MTK)                         25AG910-
 3690       CONTINUE                                                190C8
       CALL IOBIN (6HWRITER,18)                                     22AG910
            GO TO 4000                                               1SE8
 3700  PRINT 380                                                     1SE8
                  ITMX = 1                                          01SE8
            GO TO 3680                                              01SE8
 3990  PRINT 299                                                    30JA9
            GO TO 9990                                              01SE8
 3995  PRINT 393                                                    01SE8
            GO TO 9990                                              01SE8
C                                                                    1SE8
C * * * * SOLUTION PROCEDURE                                        01SE8
C                                                                    1SE8
 4000       CONTINUE                                                01SE8
       CALL IOBIN(6HREWIND,10)                                      19AG910
       CALL IOBIN(6HREWIND,14)                                      19AG910
       CALL IOBIN(6HREWIND,11)                                      19AG910
            DO 4020 J = 1, MYP7                                     190C8
            DO 4010 I = 1, MXP7                                     190C8
                  WS(I,J) = 0.0                                     190C8
                  WTM1(I,J) = 0.0                                   190C8
                  WTM2(I,J) = 0.0                                   190C8
                  W(I,J) = 0.0                                      02DE8
 4010       CONTINUE                                                190C8
 4020       CONTINUE                                                190C8
            DO 4030 J = 3, MYP5                                     02DE8
       CALL IOBIN(5HWRITE ,10,W(3,J),MTK)                           25AG910-
       CALL IOBIN(5HWRITE ,14,W(3,J),MTK)                           25AG910-
       CALL IOBIN(5HWRITE ,11,W(3,J),MTK)                           25AG910-
 4030       CONTINUE                                                02DE8
       CALL IOBIN (6HWRITER,10)                                     22AG910
       CALL IOBIN (6HWRITER,14)                                     22AG910
       CALL IOBIN (6HWRITER,11)                                     22AG910
 4035       CONTINUE                                                19AG910
            IF ( ITMX  .EQ.  0 )            ITMX = 1                01SE8
                  MTP1  = MT + 1                                    01SE8
            DO 6000 TIM = 1, MTP1                                   01SE8
       CALL DYNLD (NAM, JSFT, MSPD, JSYM, K11, K22, KONT, DQM, NDL,  02JL0
      1          I1D, J1D, I2D, J2D, DQN, QDIF, NCR6,    L1, L2, L3 ) 16MY9
 7778  FORMAT (/10H   TIME = ,I4/)                                  10JL0
 4050       DO 5900 NITR = 1, ITMX                                  01SE8
C                                                                    1SE8
C * * *     SET ML SWITCH                                           01SE8
C                                                                    1SE8
                  NT2SW = 0                                         01SE8
            IF ( NITR  .EQ.  1 )            GO TO 4070              190C8
            IF ( TIM  .EQ.  2 )             GO TO 4080              01SE8
 4060             ML = -1                                           310C9
            GO TO 4090                                              01SE8
```

```
 4070     IF ( TIM  .GT.  3 )              GO TO 4060                    01SE8
 4080         ML = 1                                                     31OC9
 4090     CONTINUE                                                       01SE8
C                                                                         1SE8
C * * *   REWIND FILES FOR FSUB                                          01SE8
C                                                                         1SE8
 5000 CALL IOBIN(6HREWIND,6)                                             19AG9IO
      CALL IOBIN(6HREWIND,7)                                             19AG9IO
      CALL IOBIN(6HREWIND,8)                                             19AG9IO
      CALL IOBIN(6HREWIND,10)                                            19AG9IO
      CALL IOBIN(6HREWIND,11)                                            19AG9IO
      CALL IOBIN(6HREWIND,12)                                            19AG9IO
      CALL IOBIN(6HREWIND,13)                                            19AG9IO
      CALL IOBIN(6HREWIND,14)                                            19AG9IO
      CALL IOBIN(6HREWIND,15)                                            19AG9IO
      CALL IOBIN(6HREWIND,16)                                            19AG9IO
      CALL IOBIN(6HREWIND,18)                                            19AG9IO
C                                                                         1SE8
C * * *   COMPUTE DEFLECTIONS                                            01SE8
C                                                                         1SE8
      CALL FRIP4 ( L1, L2, L3, ML, A, AM1, AM2, ATM, B, BM1, EP1, C,CM1,01SE8
     1             D, E, ET2, DT, CC, ET1, EE, FF, W, N1, N2, N3, QI,    01SE8
     2             QD1, QD2, QD3, QI1, QI2, QI3, SK, RHO, DF, SF )       25MR9
          IF ( TIM  .EQ.  2 )              GO TO 5005                    01SE8
          GO TO 5090                                                     01SE8
 5005     IF ( NT2SW  .EQ.  1 )            GO TO 5090                    27JE9
C                                                                         1SE8
C * * *   SET UP SECOND PASS OF SOLUTION FOR DISPLACEMENT AT TIME STEP 201SE8
C                                                                         1SE8
 5010 CALL MASSAC ( L1, L2, L3, W, RHO, DF, QI              )            01SE8
              NT2SW = 1                                                  01SE8
              ML = 1                                                     30JA9
          GO TO 5000                                                      1SE8
C                                                                         1SE8
C * * *   COMPUTE QI1 - LOAD CORRECTION FOR NONLINEAR FOUNDATION         01SE8
C                                                                         1SE8
 5090     CONTINUE                                                       01SE8
 5100 CALL NONLIN 4 ( NCR7, TOL,                                         19OC8
     1                IM1, IM2, JM1, JM2, QML, WML, SFN, NPC,            01SE8
     2                W, WS, WMAX, L1, L2, L3, 10, 19,                   19OC8
     3                QI1F, IPSW, IPSD, NCOUNT )                         01SE8
          DO 5120 I = 1, MXP7                                            01SE8
          DO 5110 J = 1, MYP7                                            01SE8
              WS(I,J) = W(I,J)                                           01SE8
 5110     CONTINUE                                                       01SE8
 5120     CONTINUE                                                       01SE8
C                                                                         1SE8
      CALL IOBIN(6HREWIND,4)                                             19AG9IO
      CALL IOBIN(6HREWIND,5)                                             19AG9IO
      CALL IOBIN(6HREWIND,14)                                            19AG9IO
      CALL IOBIN(6HREWIND,18)                                            19AG9IO
C * * *   TEST FOR CLOSURE                                               01SE8
C                                                                         1SE8
          IF ( NCOUNT  .EQ.  0 )           GO TO 5200                    01SE8
C                                                                         1SE8
C * * *   MONITOR PRINT                                                  01SE8
```

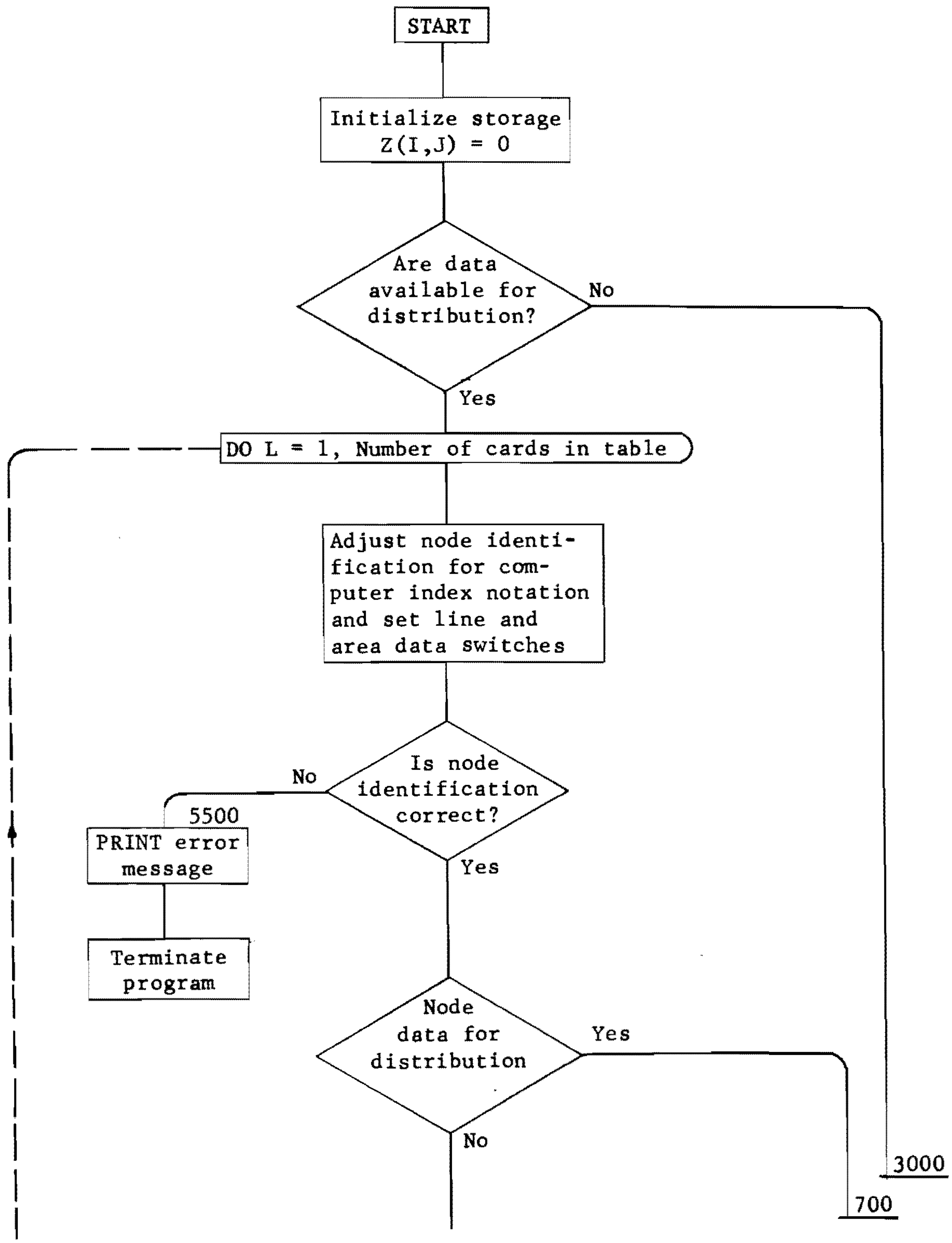
```
C                                                                         1SE8
      PRINT 550, NITR, NCOUNT                                            01SE8
          IF ( IPSW  .EQ.  1 )             PRINT 552                     01SE8
          IF ( IPSD  .EQ.  1 )             PRINT 554                     01SE8
              ILSW = 0                                                   01SE8
              LSW  = 0                                                   01SE8
 5160 CONTINUE                                                           29JA9
          IF ( MON  .EQ.  0 )              GO TO 5900                    01SE8
          GO TO 5300                                                     01SE8
C                                                                         1SE8
C * * *   SET PRINT SWITCHES FOR CLOSED SOLUTION                         01SE8
C                                                                         1SE8
 5200         ILSW = 1                                                   01SE8
          IF ( TIM  .EQ.  1 )              GO TO 5240                    01SE8
          IF ( OP  .EQ.  1 )               GO TO 5240                    17AP9
              ICT = TIM - 1                                              19JL9
 5210         ICT = ICT - OP                                             17AP9
          IF ( ICT ) 5220, 5240, 5210                                    01SE8
 5220         LSW = 0                                                    01SE8
          GO TO 5260                                                     01SE8
 5240         LSW = 1                                                    01SE8
 5260     CONTINUE                                                       01SE8
              ICT = TIM - 1                                              01SE8
C                                                                         1SE8
C * * *   PRINT DATA OUTPUT HEADINGS                                     01SE8
C                                                                         1SE8
 5300     IF ( LSW   .EQ.  0 )             GO TO 5350                    01SE8
      PRINT 30                                                           30JA9
      PRINT 1                                                            01SE8
      PRINT 150, ( AN1(N), N = 1, 32 )                                   01SE8
      PRINT 155, NPROB, ( AN2(N), N = 1, 14 )                            01SE8
      PRINT 560                                                           1SE8
          IF ( TIM  .EQ.  1 )              GO TO 5310                    01SE8
      PRINT 562, ICT                                                     01SE8
          GO TO 5320                                                     01SE8
 5310 PRINT 561                                                          01SE8
 5320     IF ( NCR7  .EQ.  0 )             GO TO 5390                    30JA9
      PRINT 565, NITR                                                    01SE8
          GO TO 5390                                                     01SE8
C                                                                         1SE8
C * * *   PRINT ONLY MONITOR STATION DATA                                01SE8
C                                                                         1SE8
 5350     IF ( ILSW  .EQ.  0 )             GO TO 5380                    01SE8
      PRINT 562, ICT                                                     01SE8
          IF ( NCR7  .EQ.  0 )             GO TO 5370                    17AP9
      PRINT 565, NITR                                                    01SE8
 5370     IF ( MON  .EQ.  0  )             GO TO 5920                    01SE8
          GO TO 5390                                                     01SE8
 5380     IF ( MON  .EQ.  0  )             GO TO 5900                    01SE8
 5390     CONTINUE                                                       21MY9
C                                                                         1SE8
C * * *   OBTAIN STIFFNESS VALUES FOR MOMENT AND REACTION CALCULATIONS  01SE8
C                                                                         1SE8
 5400     CONTINUE                                                       16MY9
C * * *   READ 5 FOR CT - REQD AT J STATIONS  J AND J+1                  01SE8
C                                                                         1SE8
```

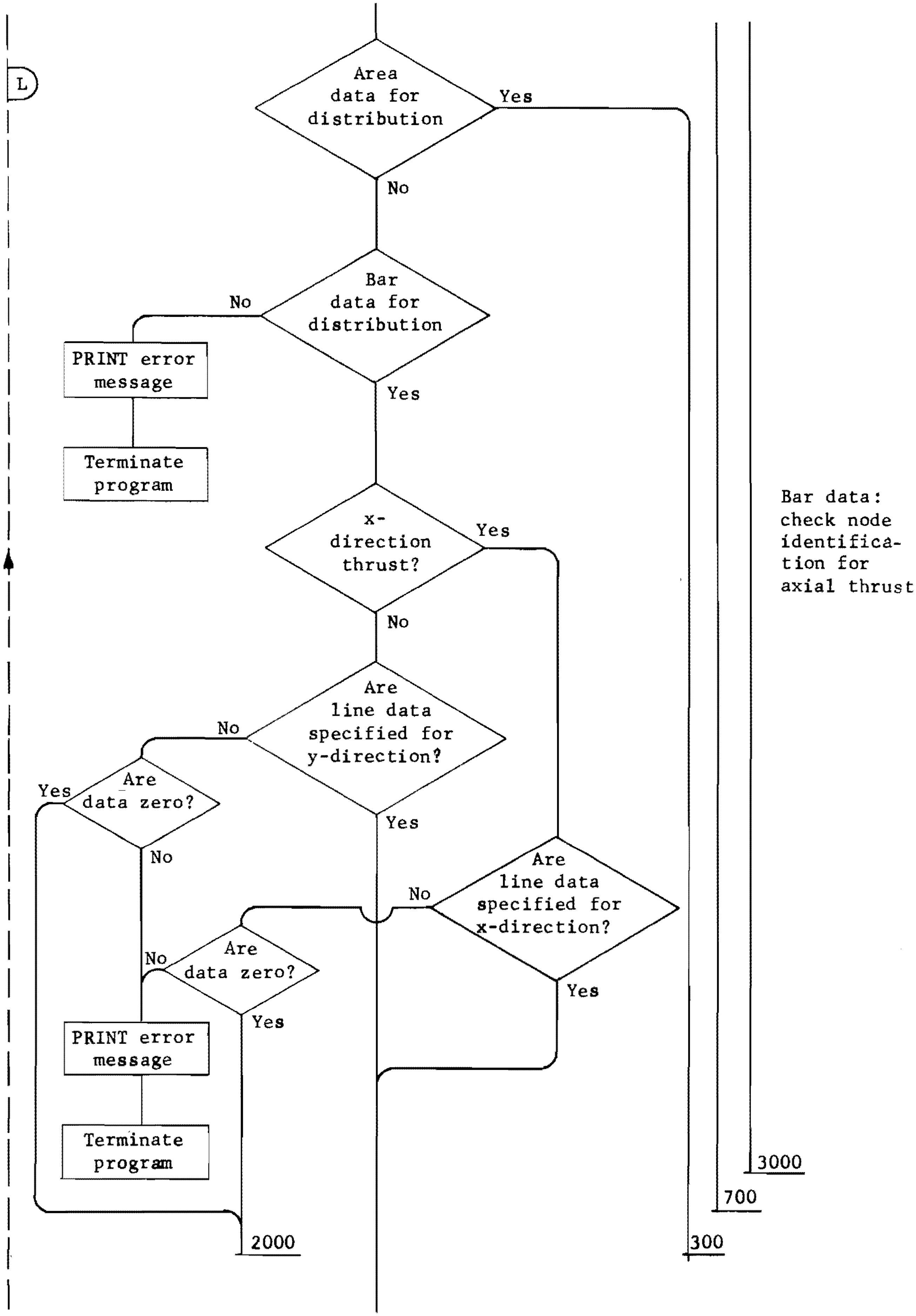
```
      CALL IOBIN(4HREAD    ,4,DX,MTK)                                   25AG910-
      CALL IOBIN(4HREAD    ,4,DX,MTK)                                   25AG910-
      CALL IOBIN(4HREAD    ,4,DX,MTK)                                   25AG910-
      CALL IOBIN(4HREAD    ,5,DX,MTK)                                   25AG910-
      CALL IOBIN(4HREAD    ,5,DX,MTK)                                   25AG910-
      CALL IOBIN(4HREAD    ,5,DX,MTK)                                   25AG910-
      CALL IOBIN(4HREAD    ,5,DX,MTK)                                   25AG910-
      CALL IOBIN(4HREAD    ,5,DX,MTK)                                   25AG910-
      CALL IOBIN(4HREAD    ,5,CT,MXP3)                                  25AG910-
      CALL IOBIN(4HREAD    ,14,DX,MTK)                                   6SE910-
      CALL IOBIN(4HREAD    ,18,DX,MTK)                                   6SE910-
         DO 5800 J = 4, MYP4                                            01SE8
         DO 5420 K = 1, MXP3
             CT(K,2) = CT(K,1)                                          01SE8
 5420    CONTINUE                                                       01SE8
      CALL IOBIN(4HREAD    ,5,DX,MXP3)                                  25AG910-
      CALL IOBIN(4HREAD    ,5,DX,MXP3)                                  25AG910-
      CALL IOBIN(4HREAD    ,5,CT,MXP3)                                  25AG910-
      CALL IOBIN(4HREAD    ,4,DX,MXP3)                                  25AG910-
      CALL IOBIN(4HREAD    ,4,DY,MXP3)                                  25AG910-
      CALL IOBIN(4HREAD    ,4, S,MXP3)                                  25AG910-
      CALL IOBIN(4HREAD    ,14,QI1,MXP3)                                25AG910-
      CALL IOBIN(4HREAD    ,18, SF,MXP3)                                25AG910-
 5430    CONTINUE                                                       19JU9
             JSTA = J - 4                                               01SE8
         IF ( LSW  .EQ.  1 )            PRINT 10                        30JA9
         DO 5750 I = 4, MXP4                                            17DE9
             ISTA = I - 4                                               16MY9
             K = I - 2                                                  16MY9
         IF ( LSW  .EQ.  1 )            GO TO 5500                      01SE8
         DO 5460 MS = 1, MON                                            01SE8
         IF ( ISTA  .EQ.  MSX(MS) )     GO TO 5450                      01SE8
         GO TO 5460                                                     01SE8
 5450    IF ( JSTA  .EQ.  MSY(MS) )     GO TO 5500                      01SE8
 5460    CONTINUE                                                       01SE8
         GO TO 5750                                                     01SE8
C                                                                        1SE8
C * * *  COMPUTE MOMENTS AND REACTIONS                                  01SE8
C                                                                        1SE8
 5500    CONTINUE                                                       01SE8
             DDWX  = ODHX2  * ( W(I-1,J) - 2.0 * W(I,J) + W(I+1,J) )    01SE8
             DDWY  = ODHY2  * ( W(I,J-1) - 2.0 * W(I,J) + W(I,J+1) )    01SE8
             DDWXY = ODHXHY * ( W(I-1,J-1) - W(I-1,J+1) - W(I+1,J-1)    01SE8
     1                        + W(I+1,J+1) ) / 4.0                      01SE8
         IF ( DX( K )  .GT.  DY( K ) ) GO TO 5520                       19MY9
              DPR = DX(K) * PR                                          21MY9
         GO TO 5550                                                     01SE8
 5520        DPR = DY( K ) * PR                                         19MY9
 5550    CONTINUE                                                       01SE8
             BMX = DX( K ) * DDWX + DPR * DDWY                          19MY9
             BMY = DPR * DDWX + DY( K ) * DDWY                          19MY9
C                                                                        1SE8
             DXY = ( CT( K ,1) + CT(K+1,1)  + CT( K ,2) + CT(K+1,2) )   19MY9
     1                 / 4.0                                            01SE8
             BMXY = - DXY * DDWXY                                       13JE9
C                                                                        1SE8
C * * *  COMPUTE MAXIMUM BENDING MOMENT AND ANGLE FROM X-AXIS TO MAX    01SE8
C                                                                        1SE8
             ABMX = ABS( BMX )                                          13JE9
             ABMY = ABS( BMY )                                          13JE9
             ABMXY = ABS( BMXY )                                        13JE9
C                                                                       17DE9
C * * *  THIS PROGRAM WILL ZERO THE MOMENT IF LESS THAN .000001         17DE9
C                                                                       17DE9
         IF ( ABMX  .LT.  10E-06 )      BMX = 0.0                       17DE9
         IF ( ABMY  .LT.  10E-06 )      BMY = 0.0                       17DE9
         IF ( ABMXY .LT.  10E-06 )      BMXY= 0.0                       17DE9
         IF ( ABMX  .GT.  ABMY )        GO TO 5552                      13JE9
         IF ( ABMY  .GT.  ABMXY )       GO TO 5554                      13JE9
 5551        BMOM = ABMXY                                               13JE9
         GO TO 5558                                                     13JE9
 5552    IF ( ABMX  .GT.  ABMXY )       GO TO 5556                      13JE9
         GO TO 5551                                                     13JE9
 5554        BMOM = ABMY                                                13JE9
         GO TO 5558                                                     13JE9
 5556        BMOM = ABMX                                                13JE9
 5558    CONTINUE                                                       13JE9
             MCK = BMOM * 10E-06                                        13JU9
         IF ( ABMX  .LT.  MCK )         BMX = 0.0                       13JU9
         IF ( ABMY  .LT.  MCK )         BMY = 0.0                       13JU9
         IF ( ABMXY .LT.  MCK )         BMXY = 0.0                      13JU9
C                                                                       13JE9
C * * *  CALCULATE MAXIMUM MOMENT - POSITIVE OR NEGATIVE                13JE9
C                                                                       13JE9
             CNTR = ( BMX + BMY ) * 0.5                                 30JA9
             SIDE = ( BMX - BMY ) * 0.5                                 01SE8
             RAD2 = SIDE * SIDE + BMXY * BMXY                           01SE8
             RAD  = SQRT ( RAD2 )                                       01SE8
         IF ( CNTR  .GT.  0.0 )         GO TO 5560                      13JE9
         IF ( CNTR  .EQ.  0.0 )         GO TO 5560                      13JE9
             BMAX = CNTR - RAD                                          13JE9
         GO TO 5562                                                     13JE9
 5560        BMAX = CNTR + RAD                                          13JE9
 5562    CONTINUE                                                       13JE9
C                                                                       13JE9
C * * *  SPECIAL CASE - WHEN MX = MY AND THE CASE WHEN MXY = 0          13JE9
C                                                                       13JE9
         IF ( SIDE  .EQ.  0.0 )         GO TO 5566                      13JE9
         IF ( BMXY  .EQ.  0.0 )         GO TO 5568                      13JE9
         GO TO 5570                                                     13JE9
 5566    IF ( BMXY  .EQ.  0.0 )         GO TO 5569                      13JE9
         IF ( BMXY  .GT.  0.0 )         GO TO 5567                      13JE9
             BETA = 0.7853982                                           13JE9
         GO TO 5580                                                     13JE9
 5567        BETA = - 0.7853982                                         13JE9
         GO TO 5580                                                     13JE9
 5568    IF ( BMX  .GT.  BMY )          GO TO 5569                      13JE9
             BETA = 1.5707963                                           13JE9
         GO TO 5580                                                     13JE9
 5569        BETA = 0.0                                                 13JE9
         GO TO 5580                                                     13JE9
C                                                                       13JE9
```

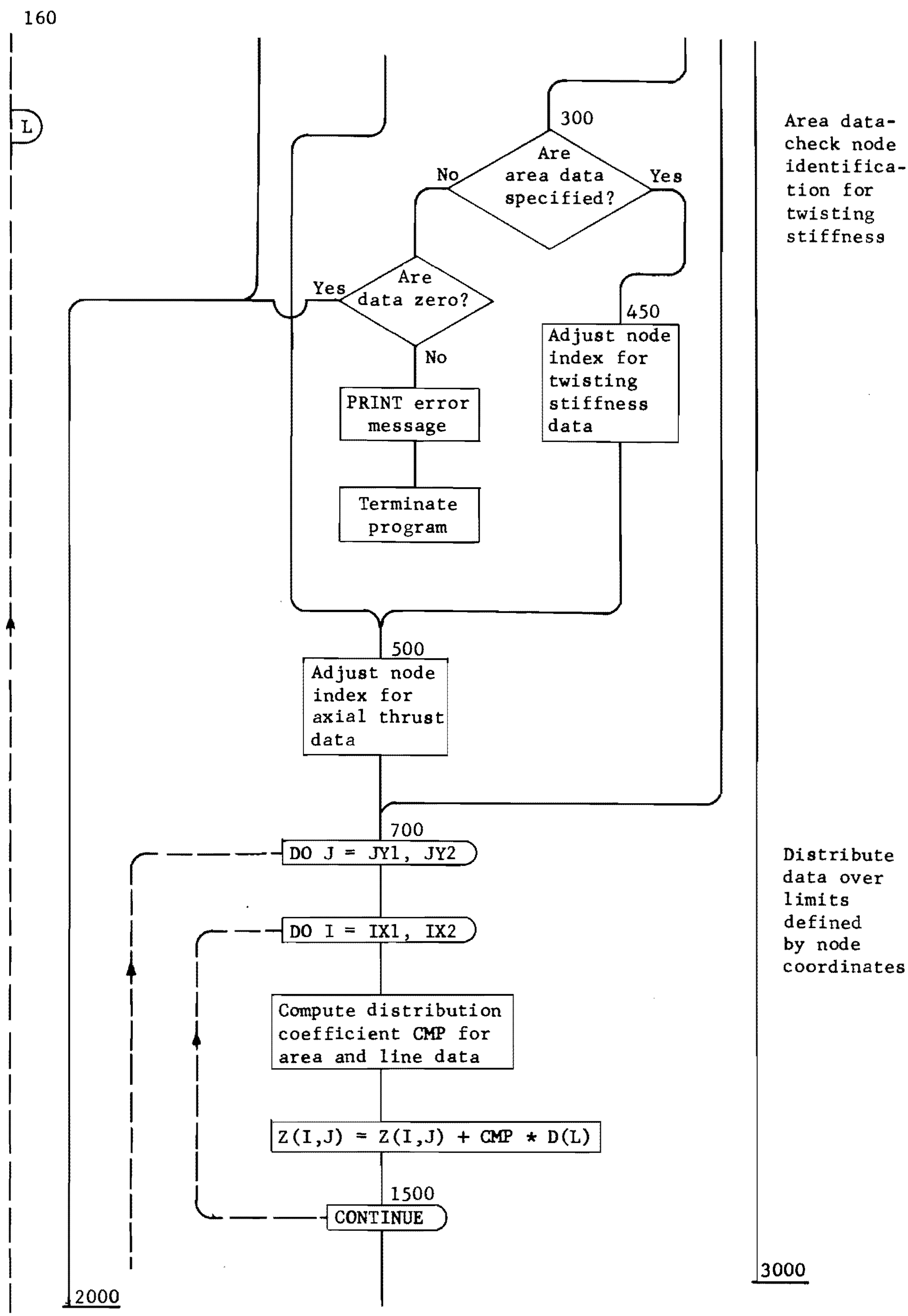
```
C * * *   GENERAL CALCULATION OF BETA                                      13JE9
C                                                                          13JE9
 5570          MOM3 = CNTR - RAD                                           13JE9
               MX3  = MOM3 - BMX                                           13JE9
               BETA = ATAN ( BMXY / MX3 )                                  13JE9
C                                                                          13JE9
C * * *   CHECK WHETHER MAX PRINCIPAL MOMENT IS MAXIMUM ABSOLUTE MOMENT    13JE9
C                                                                          13JE9
 5580     IF ( CNTR )  5582, 5590, 5590                                    13JE9
 5582     IF ( BETA )  5584, 5584, 5586                                    13JE9
 5584          BETA = BETA + 1.5707963                                     13419
          GO TO 5590                                                       13JE9
 5586          BETA = BETA - 1.5707963                                     13JE9
 5590     CONTINUE                                                         13JE9
               BETA = BETA * 57.29578                                      13JE9
 5600     CONTINUE                                                         13JE9
C                                                                           1SE8
C * * *   COMPUTE REACTION                                                 01SE8
C                                                                           1SE8
               REACT = ( SF( K ) + S( K ) ) * ( - W(I,J) ) + OI1( K )      19MY9
      PRINT 580, ISTA, JSTA, W(I,J), BMX, BMY, BMXY,REACT,BMAX,BETA        01SE8
C                                                                           1SE8
 5750     CONTINUE                                                         01SE8
 5800     CONTINUE                                                         01SE8
C                                                                           1SE8
C * * *   CHECK FOR CLOSURE - IF CLOSED PREPARE FOR NEXT TIME STEP         01SE8
C                                                                           1SE8
          IF ( NCOUNT  .EQ.  0 )        GO TO 5920                         01SE8
 5900     CONTINUE                                                         01SE8
C                                                                           1SE8
      PRINT 591                                                            29JA9
C * * *   SET DEFLECTIONS FOR NEXT TIME STEP                               01SE8
C                                                                           1SE8
 5920     CONTINUE                                                         01SE8
      CALL IOBIN(6HREWIND,11)                                              19AG9IO
      CALL IOBIN(6HREWIND,12)                                              19AG9IO
      CALL IOBIN(6HREWIND,13)                                              19AG9IO
      CALL IOBIN(6HREWIND,14)                                              19AG9IO
      CALL IOBIN(6HREWIND,15)                                              19AG9IO
      CALL IOBIN(6HREWIND,16)                                              19AG9IO
          DO 5960 J = 3, MYP5                                              01SE8
          IF ( TIM  .EQ.  1 )           GO TO 5930                         15AP9
      CALL IOBIN(4HREAD   ,12,QD2,MXP3)                                    25AG9IO-
      CALL IOBIN(5HWRITE ,13,QD2,MXP3)                                     30SE9IO-
      CALL IOBIN (4HREAD   ,15,OI2,MXP3)                                   25AG9IO-
      CALL IOBIN(5HWRITE ,16,OI2,MXP3)                                     25AG9IO-
 5930     DO 5940 I = 3, MXP5                                              15AP9
               WTM2(I,J) = WTM1(I,J)                                       01SE8
               WTM1(I,J) = W(I,J)                                          01SE8
 5940     CONTINUE                                                         01SE8
 5960     CONTINUE                                                         01SE8
          IF ( TIM  .EQ.  1 )           GO TO 5970                         15AP9
      CALL IOBIN (6HWRITER,13)                                             22AG9IO
      CALL IOBIN (6HWRITER,16)                                             22AG9IO
      CALL IOBIN(6HREWIND,12)                                              19AG9IO
      CALL IOBIN(6HREWIND,15)                                              19AG9IO
 5970     DO 5980 J = 3, MYP5                                              15AP9
      CALL IOBIN(4HREAD   ,14,OI1,MXP3)                                    25AG9IO-
      CALL IOBIN(5HWRITE ,15,OI1,MXP3)                                     25AG9IO-
      CALL IOBIN(4HREAD   ,11,QD1,MXP3)                                    25AG9IO-
      CALL IOBIN(5HWRITE ,12,QD1,MXP3)                                     25AG9IO-
 5980     CONTINUE                                                         01SE8
      CALL IOBIN (6HWRITER,12)                                             22AG9IO
      CALL IOBIN (6HWRITER,15)                                             22AG9IO
 5983     CONTINUE                                                         19AG9IO
          IF ( TIM  .EQ.  1 )           GO TO 5990                         01SE8
      CALL INERTIA ( L1, L2, L3, W, WTM1, WTM2, AA, BB, CC, DD,            01SE8
     1  ET1, EEM1, EEM2, DDM1, SK, N1, N2, N3, SF )                        22AG9
 5990     CONTINUE                                                         30JA9
 6000     CONTINUE                                                         22JA9
      CALL TIC TOC (4)                                                     22JA9
          GO TO 1000                                                       22JA9
 9950 PRINT 393                                                            22JA9
 9990     CONTINUE                                                         22JA9
      PRINT 30                                                             22JA9
      PRINT 1                                                              22JA9
      PRINT 150                                                            22JA9
      CALL TIC TOC (2)                                                     22JA9
      PRINT 600                                                            22JA9
      END                                                                  30JA9
C
```

SUBROUTINE INTERP9

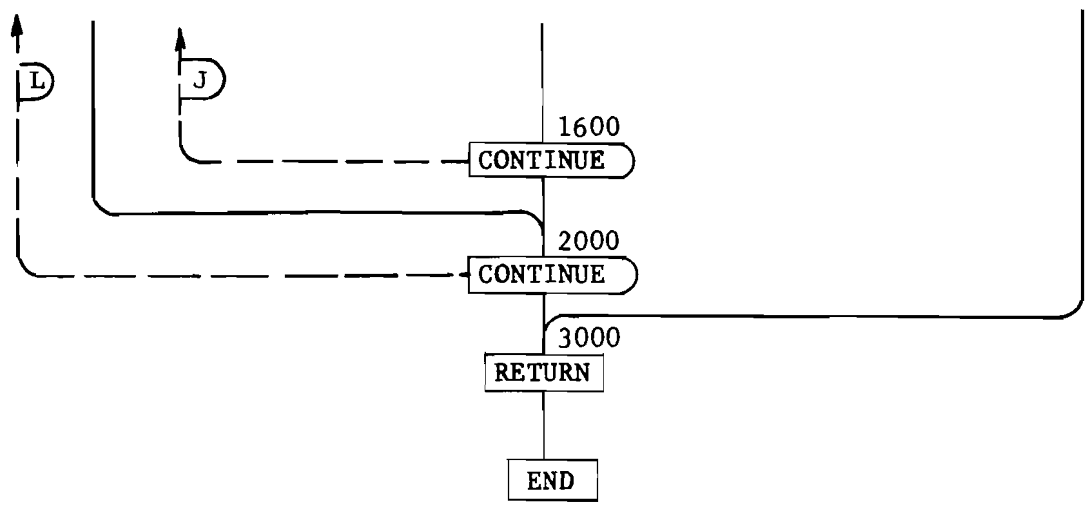
This subroutine performs the distribution
for data from Tables 2 through 5.




L
Area data for distribution
Yes
No
Bar data for distribution
No
PRINT error message
Terminate program
Yes
x-direction thrust?
Yes
No
Are line data specified for y-direction?
No
Yes
Are data zero?
Yes
No
Are line data specified for x-direction?
No
Yes
Are data zero?
No
Yes
PRINT error message
Terminate program
2000
300
700
3000
Bar data: check node identification for axial thrust



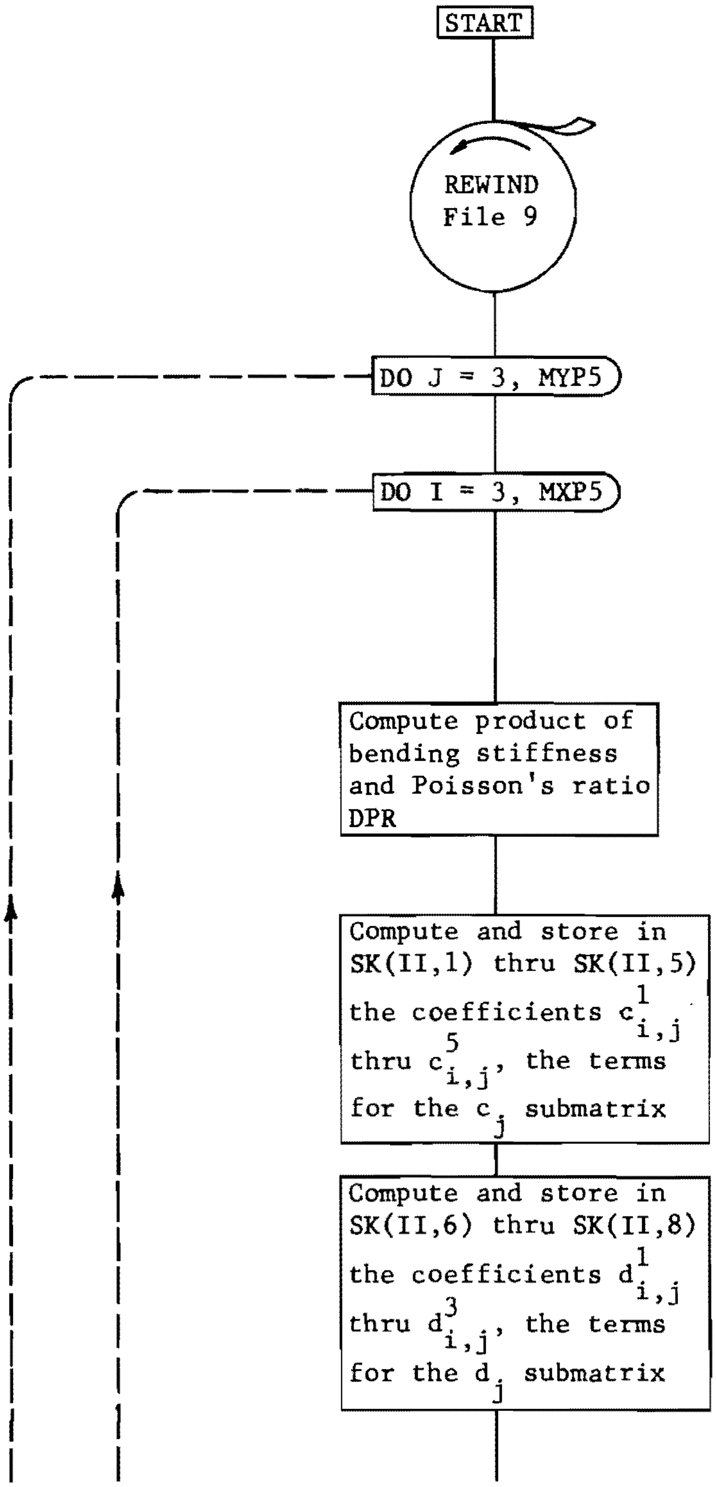
160
L
300
Are area data specified?
No
Yes
Are data zero?
Yes
No
450
Adjust node index for twisting stiffness data
PRINT error message
Terminate program
500
Adjust node index for axial thrust data
700
DO J = JY1, JY2
DO I = IX1, IX2
Compute distribution coefficient CMP for area and line data
Z(I,J) = Z(I,J) + CMP * D(L)
1500
CONTINUE
2000
3000
Area data-check node identification for twisting stiffness
Distribute data over limits defined by node coordinates


L
J
1600
CONTINUE
2000
CONTINUE
3000
RETURN
END

```
      SUBROUTINE INTERP9 ( I1, J1, I2, J2, D, ICARD, Z,                 29JA9
     1                     IS, IB, IG, L2, L3, ICX, ICY                )05JU9
      DIMENSION  I1 ( 50 ),  I2( 50 ),  J1( 50),  J2( 50 ), D( 50 ),     23AG9
     2           Z( L2,L3 )                                              01SE8
      COMMON/INCR/ MX, MY, MXP2, MXP3, MXP4, MXP5, MXP7,                 11NO8
     1           MYP2, MYP3, MYP4, MYP5, MYP7, MT                        11NO8
   10 FORMAT ( / 50H     DATA TYPE NOT PROPERLY DEFINED FOR INTERP 9    )01SE8
   20 FORMAT ( / 50H     ERROR IN INPUT OF DATA FOR DISTRIBUTION        )01SE8
   30 FORMAT ( / 51H     STATIONS NOT IN PROPER ORDER FOR INTERPOLATION)01SE8
C * * *    ZERO STORAGE BLOCK                                            01SE8
           DO 100 J = 1, MYP7                                            01SE8
           DO 50 I = 1, MXP7                                             01SE8
                Z(I,J) = 0.0                                             01SE8
   50      CONTINUE                                                      01SE8
  100      CONTINUE                                                      01SE8
           IF ( ICARD  .EQ.  0 )     GO TO 3000                          01SE8
                IPL = 4                                                  29JA9
           DO 2000 L = 1, ICARD                                          01SE8
                IX1 = I1(L) + IPL                                        01SE8
                IX2 = I2(L) + IPL                                        01SE8
                JY1 = J1(L) + IPL                                        01SE8
                JY2 = J2(L) + IPL                                        01SE8
           IF ( IX2  .LT.  IX1 )     GO TO 5500                          01SE8
           IF ( JY2  .LT.  JY1 )     GO TO 5500                          01SE8
                ISW = 0                                                  01SE8
                JSW = 0                                                  01SE8
           IF ( IX2  .GT.  IX1 )     ISW = 1                             01SE8
           IF ( JY2  .GT.  JY1 )     JSW = 1                             01SE8
C * * *    DISTRIBUTE DATA OVER AREA DEFINED BY IX1, JX1, IX2, JX2       01SE8
C                                                                         1SE8
C * * *    CHECK FOR TYPE OF DATA                                        01SE8
           IF ( IS  .GT.  0 )        GO TO  700                          01SE8
           IF ( IB  .GT.  0 )             GO TO 200                      05JU9
           IF ( IG  .GT.  0 )        GO TO  300                          01SE8
C * * *    TYPE OF DATA NOT DEFINED - ERROR                              01SE8
      PRINT 10                                                           01SE8
           GO TO 6000                                                    01SE8
C * * *    SET UP INTERPOLATION FOR GRID AND BAR TYPE DATA               01SE8
  200      IF ( ICX  .EQ.  1 )            GO TO 250                      05JU9
           IF ( JSW  .GT.  0 )            GO TO 500                      05JU9
           IF ( D(L)  .EQ.  0.0 )         GO TO 2000                     12JU9
           GO TO 275                                                     05JU9
  250      IF ( ISW  .GT.  0 )            GO TO 500                      05JU9
           IF ( D(L)  .EQ.  0.0 )         GO TO 2000                     12JU9
  275 PRINT 10                                                           05JU9
           GO TO 6000                                                    05JU9
  300      IF ( ISW  .EQ.  1 )       GO TO 400                           01SE8
           IF ( D(L)  .EQ.  0.0 )         GO TO 2000                     02JL9
      PRINT 20                                                           01SE8
           GO TO 6000                                                    01SE8
  400      IF ( JSW  .EQ.  1 )            GO TO 450                      23JU9
           IF ( D(L)  .EQ.  0.0 )         GO TO 2000                     02JL9
      PRINT 20                                                           01SE8
           GO TO 6000                                                    01SE8
  450           IX1 = IX1 + 1                                            23JU9
                JY1 = JY1 + 1                                            23JU9
  500      IF ( ICX  .EQ.  1 )            IX1 = IX1 + 1                  23JU9
           IF ( ICY  .EQ.  1 )            JY1 = JY1 + 1                  23JU9
                ISW = 0                                                  01SE8
                JSW = 0                                                  01SE8
  700      DO 1600 J = JY1, JY2                                          01SE8
           DO 1500 I = IX1, IX2                                          01SE8
                CMX = 1.0                                                01SE8
                CMY = 1.0                                                01SE8
           IF ( ISW  .EQ.  0 )       GO TO 900                           01SE8
           IF ( JSW  .EQ.  0 )       GO TO 800                           01SE8
           IF ( J  .EQ.  JY1 )       CMY = 0.5                           01SE8
           IF ( J  .EQ.  JY2 )       CMY = 0.5                           01SE8
           IF ( I  .EQ.  IX1 )       CMX = 0.5                           01SE8
           IF ( I  .EQ.  IX2 )       CMX = 0.5                           01SE8
           GO TO 1000                                                    01SE8
  800      IF ( I  .EQ.  IX1 )       CMX = 0.5                           01SE8
           IF ( I  .EQ.  IX2 )       CMX = 0.5                           01SE8
           GO TO 1000                                                    01SE8
  900      IF ( JSW  .EQ.  0 )       GO TO 1000                          01SE8
           IF ( J  .EQ.  JY1 )       CMY = 0.5                           01SE8
           IF ( J  .EQ.  JY2 )       CMY = 0.5                           01SE8
 1000           CMP = CMX * CMY                                          01SE8
                Z(I,J) = Z(I,J) + CMP * D(L)                             01SE8
 1500      CONTINUE                                                      01SE8
 1600      CONTINUE                                                      01SE8
 2000      CONTINUE                                                      01SE8
 3000 RETURN                                                             01SE8
 5500 PRINT 30                                                           01SE8
 6000      CONTINUE                                                      29JA9
      END
C
```

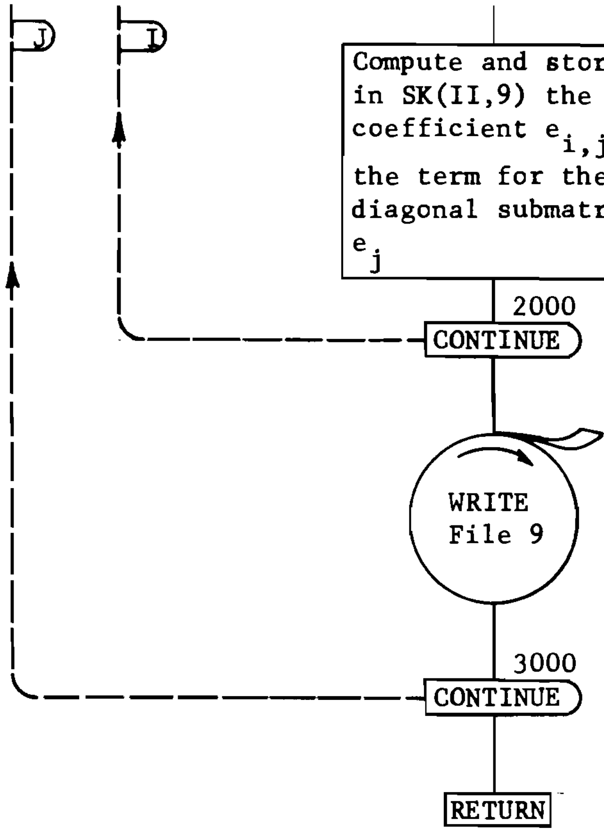
SUBROUTINE STIF1

This subroutine forms the stiffness matrix coefficients for bending stiffness and linear support springs.



The horizontal partition of the stiffness matrix is identified by J and the row in the partition by I

The minimum value of the product of Poisson's ratio and bending stiffness is determined for each node point as required

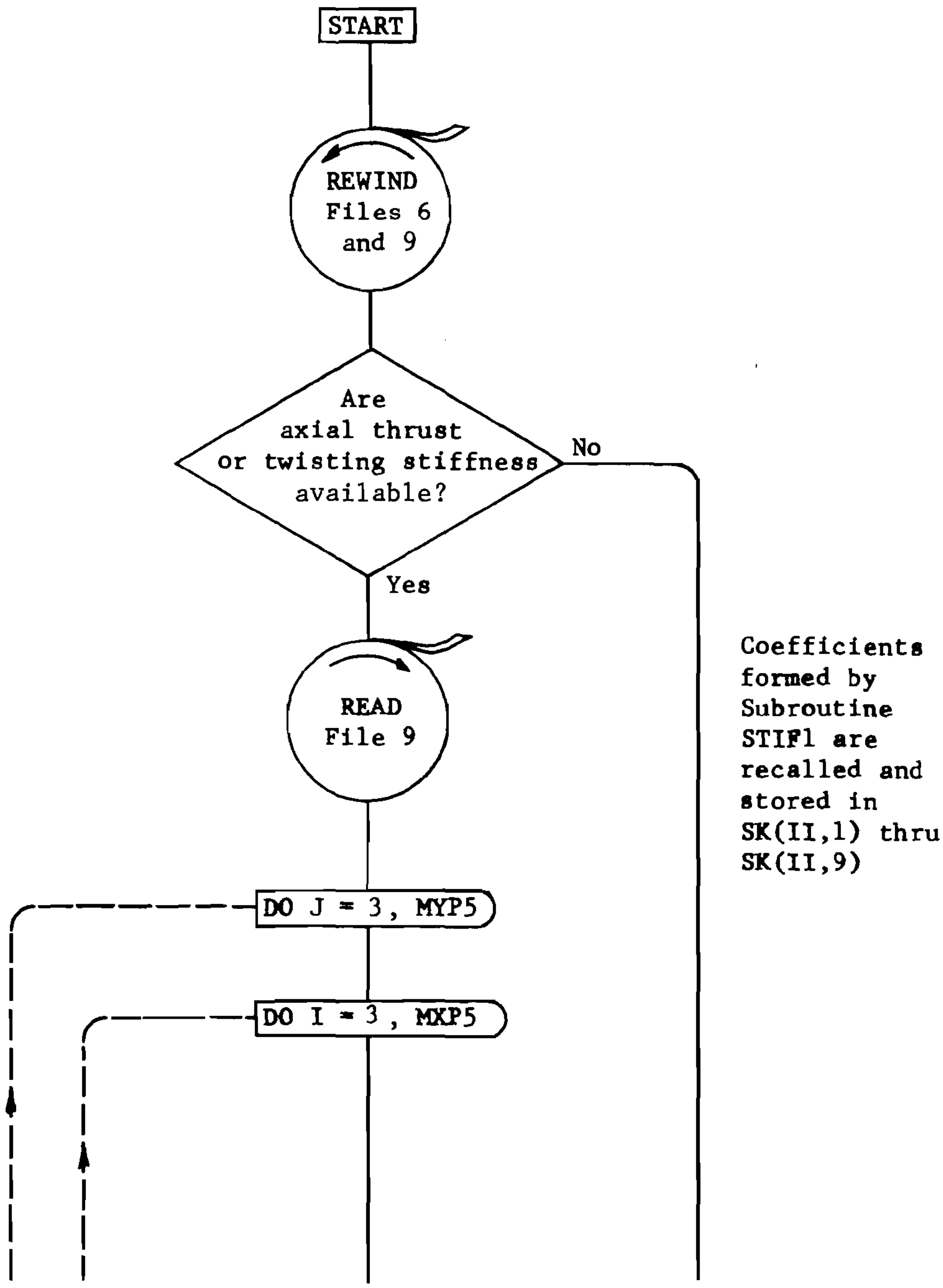


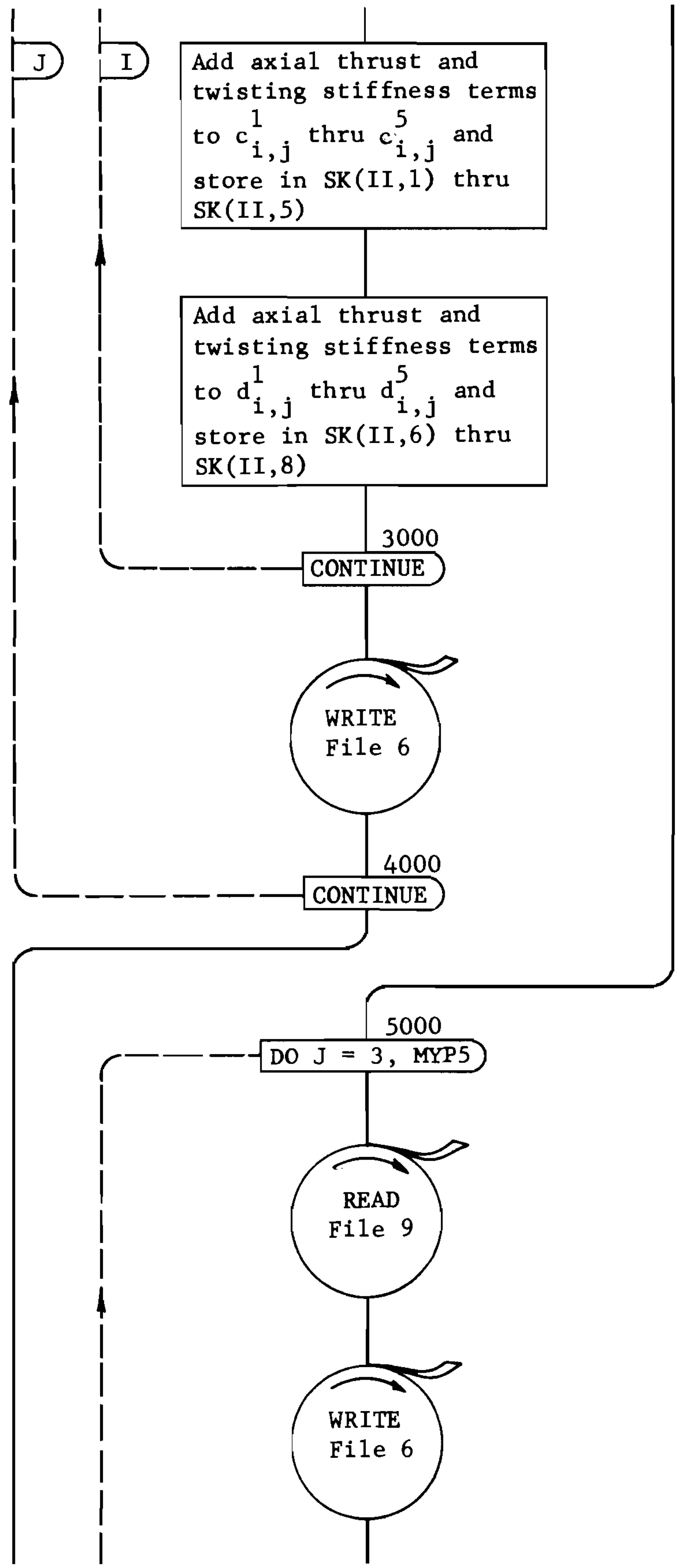
Each horizontal partition of the incomplete stiffness matrix is written on File 9

```
      SUBROUTINE STIF1 ( DX, DY, S, ST1, L1, L2, L3 )                  01SE8
      DIMENSION   DX( L2,L3 ),    DY( L2,L3 ),    S( L2,L3 ),          01SE8
     1                 ST1( L1,  9)                                     30JA9
      COMMON/INCR/ MX, MY, MXP2, MXP3, MXP4, MXP5, MXP7,               11NO8
     1                MYP2, MYP3, MYP4, MYP5, MYP7, MT                  11NO8
      COMMON/CON/  HXDHY3, HYDHX3, ODHXHY, ODHX, ODHY, PR, ODHT2, OD2HT,06JU9
     1                HXDHY, HYDHX                                      06JU9
      CALL IOBIN(6HREWIND,9)                                            19AG910
C * * *   THIS SUBROUTINE FORMS THE STIFFNESS MATRIX TERMS ASSOCIATED   01SE8
C * * *   WITH DX, DY, AND LINEAR FOUNDATION SPRINGS                    01SE8
C                                                                        1SE8
          DO 3000 J = 3, MYP5                                           01SE8
          DO 2000 I = 3, MXP5                                           17MR9
              II = I - 2                                                01SE8
C                                                                        1SE8
C * * *   COMPUTE PRODUCTS OF POISSINS RATIO AND PLATE STIFFNESS        01SE8
C                                                                        1SE8
          IF ( I  .GT.  3 )               GO TO 500                     01SE8
              LS = I - 1                                                01SE8
              LE = I + 1                                                01SE8
          DO 400 L = LS, LE                                             01SE8
              TDPX = DX(L,J) * PR                                       01SE8
              TDPY = DY(L,J) * PR                                       01SE8
          IF ( TDPX  .GT.  TDPY )         TDPX = TDPY                   01SE8
          IF ( L  .EQ.  LS )              GO TO 200                     01SE8
          IF ( L  .EQ.  LE )              GO TO 100                     01SE8
              DPR2   = TDPX                                             30JA9
          GO TO 400                                                     01SE8
  100         DPR3   = TDPX                                             30JA9
          GO TO 400                                                     01SE8
  200         DPR1   = TDPX                                             30JA9
  400     CONTINUE                                                      01SE8
          GO TO 1000                                                    01SE8
  500         DPR1 = DPR2                                               30JA9
              DPR2 = DPR3                                               30JA9
C                                                                        1SE8
C * * *   COMPUTE PRODUCTS OF POISSONS RATIO AND STIFFNESS FOR NEW STA  01SE8
C                                                                        1SE8
              TDPX = DX(I+1,J) * PR                                     01SE8
              TDPY = DY(I+1,J) * PR                                     01SE8
          IF ( TDPX  .GT.  TDPY )         TDPX = TDPY                   01SE8
              DPR3   = TDPX                                             30JA9
C                                                                        1SE8
 1000     CONTINUE                                                      17MR9
              TDPX = DX(I,J+1) * PR                                     17MR9
              TDPY = DY(I,J+1) * PR                                     01SE8
          IF ( TDPX  .GT.  TDPY )         TDPX = TDPY                   01SE8
              DPR4   = TDPX                                             30JA9
C                                                                        1SE8
C * * *   FORM MATRIX COEFFICIENTS AT ROW I OF SUB MATRIX J             01SE8
C         ST1(II,1) THRU ST1(II,5) ARE LITTLE CC TERMS                  01SE8
C         ST1(II,6) THRU ST1(II,8) ARE LITTLE DD TERMS                  01SE8
C         ST1(II,9) IS THE LITTLE EE TERM                               01SE8
C                                                                        1SE8
              ST1(II,1) = HYDHX3 * DX(I-1,J)                            01SE8
              ST1(II,2) = - HYDHX3 * 2.0 * ( DX(I-1,J) + DX(I,J) )      01SE8
     1                  - ODHXHY * 2.0 * ( DPR1    + DPR2    )          30JA9
              ST1(II,3) = HYDHX3 * ( DX(I-1,J) + 4.0 * DX(I,J) +        01SE8
     1                           DX(I+1,J) ) +                          01SE8
     2                  HXDHY3 * ( DY(I,J-1) + 4.0 * DY(I,J) +          01SE8
     3                           DY(I,J+1) ) +                          01SE8
     4                  ODHXHY * 8.0 * DPR2    + S(I,J)                 30JA9
              ST1(II,4) = - HYDHX3 * 2.0 * ( DX(I,J) + DX(I+1,J) )      01SE8
     1                  - ODHXHY * 2.0 * ( DPR2    + DPR3    )          30JA9
              ST1(II,5) = HYDHX3 * DX(I+1,J)                            01SE8
              ST1(II,6) = ODHXHY * ( DPR1    + DPR4    )                30JA9
              ST1(II,7) = - HXDHY3 * 2.0 * ( DY(I,J) + DY(I,J+1) )      01SE8
     1                  - ODHXHY * 2.0 * ( DPR2    + DPR4    )          30JA9
              ST1(II,8) = ODHXHY * ( DPR3    + DPR4    )                30JA9
              ST1(II,9) = HXDHY3 * DY(I,J+1)                            01SE8
C                                                                        1SE8
C * * *   WRITE STIFFNESS MATRIX ON TAPE 9 BY ROWS                      30JA9
C                                                                        1SE8
 2000     CONTINUE                                                      17MR9
              MTK =  9 * MXP3                                           19AG910
      CALL IOBIN(5HWRITE ,9,ST1,MTK)                                    25AG910-
C                                                                        1SE8
 3000     CONTINUE                                                      01SE8
      CALL IO3IN (6HWRITER,9 )                                          22AG910
      RETURN                                                            ,1SE8
      END                                                               ,1SE8
C
```

SUBROUTINE STFMX

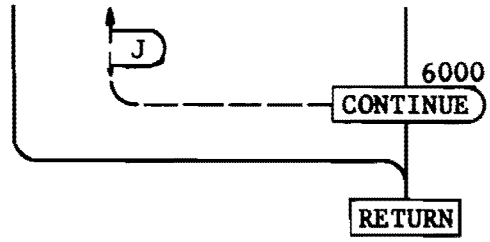
The formation of the static stiffness is completed with the addition of terms related to axial thrust and twisting stiffness to the coefficients computed by Subroutine STIF1.





Each horizontal partition of the completed stiffness matrix is stored on File 6

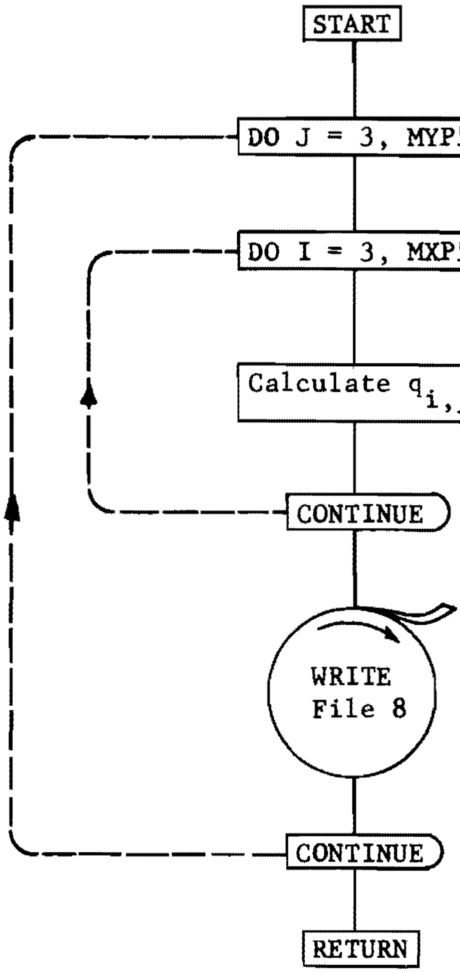
In the absence of axial thrust and twisting stiffness, the stiffness coefficients are transferred from File 9 to File 6.


J
6000
CONTINUE
RETURN

```
      SUBROUTINE STFMX ( PX, PY, CT,       SK, L1, L2, L3, NCT3 )         29JA9
      DIMENSION   PX( L2,L3 ),   PY( L2,L3 ),   CT( L2,L3 ),              01SE8
     1                                SK( L1,  9)                         29JA9
      COMMON/INCR/ MX, MY, MXP2, MXP3, MXP4, MXP5, MXP7,                  11NO8
     1             MYP2, MYP3, MYP4, MYP5, MYP7, MT                       11NO8
      COMMON/CON/  HXDHY3, HYDHX3, ODHXHY, ODHX, ODHY, PR, ODHT2, OD2HT,06JU9
     1             HXDHY, HYDHX                                           06JU9
C                                                                          1SE8
C * * *   COMPLETE FORMATION OF STIFFNESS MATRIX                          01SE8
C                                                                          1SE8
      CALL IOBIN(6HREWIND,6)                                              19AG9IO
      CALL IOBIN(6HREWIND,9)                                              19AG9IO
          IF ( NCT3  .EQ.  0 )           GO TO 5000                       01SE8
              MTK =  9 * MXP3                                             19AG9IO
          DO 4000 J = 3, MYP5                                             01SE8
      CALL IOBIN(4HREAD   ,9,SK,MTK)                                      25AG9IO-
          DO 3000 I = 3, MXP5                                             01SE8
              II = I - 2                                                  01SE8
C                                                                          1SE8
C * * *   COMPUTE MATRIX COEFFICIENTS BASED ON PX, PY, AND CT TERMS       01SE8
C                                                                          1SE8
              SK(II,1) =  SK(II,1)                                        29JA9
              SK(II,2) =  SK(II,2) - ODHXHY * 2.0 * ( CT(I,J) +           29JA9
     1                        CT(I,J+1) ) - ODHX * PX(I,J)                07JU9
              SK(II,3) =  SK(II,3) + ODHXHY * 2.0 * ( CT(I,J) +           29JA9
     1                        CT(I,J+1) + CT(I+1,J) + CT(I+1,J+1) ) +     01SE8
     2                        ODHX * ( PX(I,J) + PX(I+1,J) ) +            07JU9
     3                        ODHY * ( PY(I,J) + PY(I,J+1) )              07JU9
              SK(II,4) =  SK(II,4) - ODHXHY * 2.0 * ( CT(I+1,J) +         29JA9
     1                        CT(I+1,J+1) ) - ODHX * PX(I+1,J)            07JU9
              SK(II,5) =  SK(II,5)                                        29JA9
              SK(II,6) =  SK(II,6) + ODHXHY * 2.0 * CT(I,J+1)             29JA9
              SK(II,7) =  SK(II,7) - ODHXHY * 2.0 * ( CT(I,J+1) +         29JA9
     1                        CT(I+1,J+1) ) - ODHY * PY(I,J+1)            07JU9
              SK(II,8) =  SK(II,8) + ODHXHY * 2.0 * CT(I+1,J+1)           29JA9
              SK(II,9) =  SK(II,9)                                        29JA9
 3000     CONTINUE                                                        01SE8
      CALL IOBIN(5HWRITE ,6,SK,MTK)                                       19AG9IO
 4000     CONTINUE                                                        01SE8
          GO TO 7000                                                      01SE8
 5000     DO 6000 J = 3, MYP5                                             01SE8
      CALL IOBIN(4HREAD   ,9,SK,MTK)                                      25AG9IO-
      CALL IOBIN(5HWRITE ,6,SK,MTK)                                       25AG9IO-
 6000     CONTINUE                                                        01SE8
 7000 CALL IOBIN (6HWRITER,6 )                                            30SE9IO
      RETURN                                                              30SE9
      END                                                                  1SE8
C
```

SUBROUTINE STALD

This subroutine forms the static load vector and writes it on File 8.

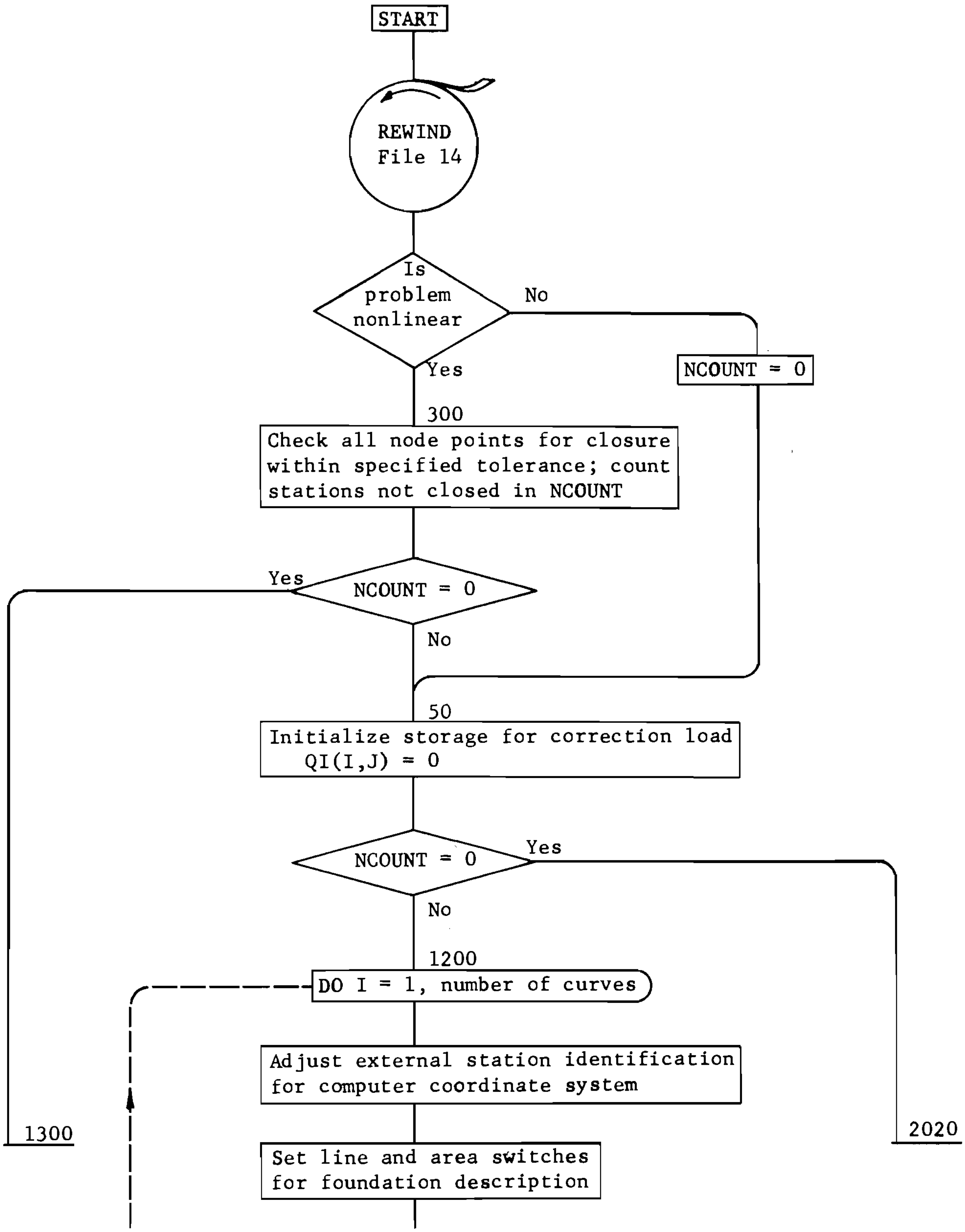


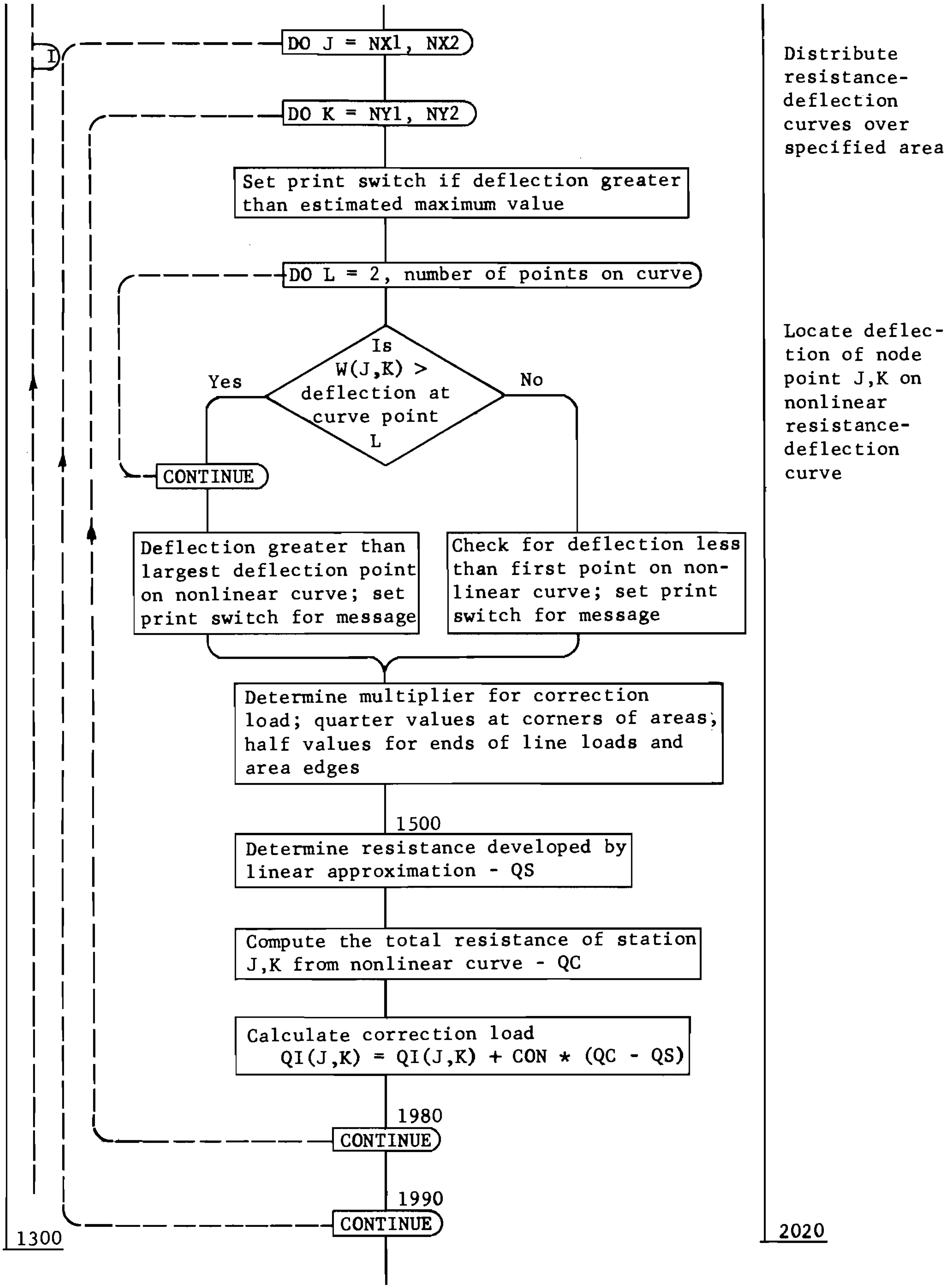
Each horizontal partition of the static load vector is stored on File 8

```
      SUBROUTINE STALD ( Q, TX, TY, FF, L1, L2, L3 )                    29JA9
      DIMENSION    Q( L2,L3 ),   TX( L2,L3 ),   TY( L2,L3 ),   FF( L1 ) 01SE8
      COMMON/INCR/ MX, MY, MXP2, MXP3, MXP4, MXP5, MXP7,                11NO8
     1             MYP2, MYP3, MYP4, MYP5, MYP7, MT                     11NO8
      COMMON/CON/  HXDHY3, HYDHX3, ODHXHY, ODHX, ODHY, PR, ODHT2, OD2HT,06JU9
     1             HXDHY, HYDHX                                         06JU9
C                                                                        1SE8
C * * *   THIS SUBROUTINE FORMS THE STATIC LOAD VECTOR                  01SE8
C                                                                        1SE8
      CALL IOBIN(6HREWIND,8)                                            19AG9IO
          DO 3000 J = 3, MYP5                                           01SE8
          DO 2000 I = 3, MXP5                                           01SE8
               II = I - 2                                               01SE8
               FF(II) = Q(I,J) - ODHX * ( TX(I,J) - TX(I+1,J) ) -       01SE8
     1                           ODHY * ( TY(I,J) - TY(I,J+1) )         01SE8
 2000     CONTINUE                                                      01SE8
      CALL IOBIN(5HWRITE ,8,FF,MXP3)                                    25AG9IO-
 3000     CONTINUE                                                      01SE8
      CALL IOBIN (6HWRITER,8 )                                          22AG9IO
      RETURN                                                            01SE8
      END
C
```

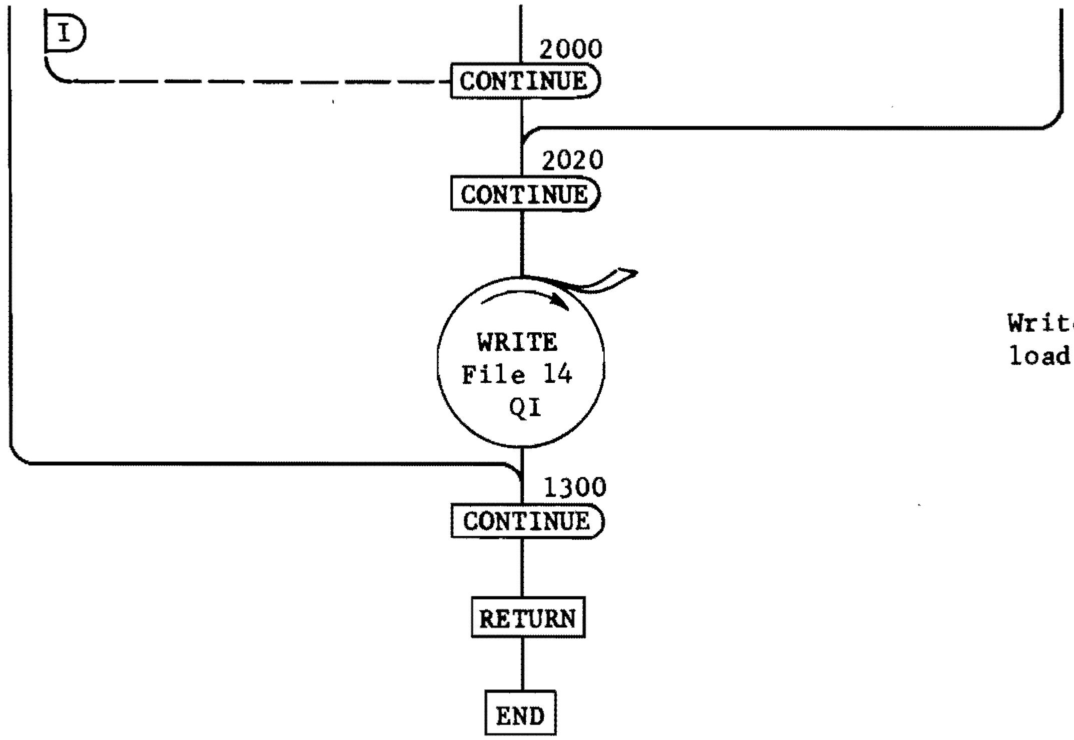
SUBROUTINE NONLIN4

The correction load for the load iteration procedure is computed by this subroutine.





DO J = NX1, NX2
DO K = NY1, NY2
Set print switch if deflection greater than estimated maximum value
DO L = 2, number of points on curve
Is W(J,K) > deflection at curve point L
Yes
No
CONTINUE
Deflection greater than largest deflection point on nonlinear curve; set print switch for message
Check for deflection less than first point on nonlinear curve; set print switch for message
Determine multiplier for correction load; quarter values at corners of areas, half values for ends of line loads and area edges
1500
Determine resistance developed by linear approximation - QS
Compute the total resistance of station J,K from nonlinear curve - QC
Calculate correction load QI(J,K) = QI(J,K) + CON * (QC - QS)
1980
CONTINUE
1990
CONTINUE
1300
2020
I
Distribute resistance-deflection curves over specified area
Locate deflection of node point J,K on nonlinear resistance-deflection curve

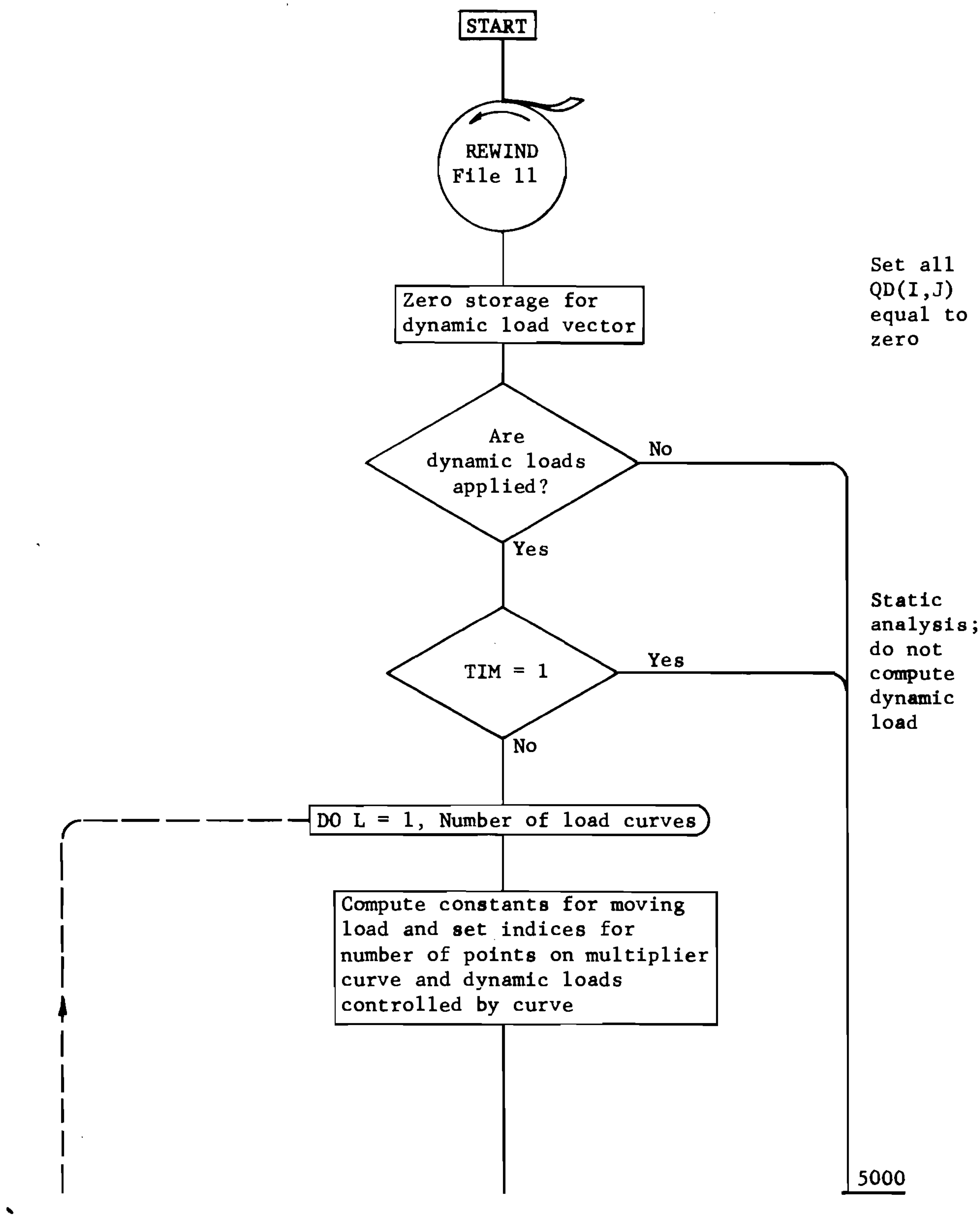


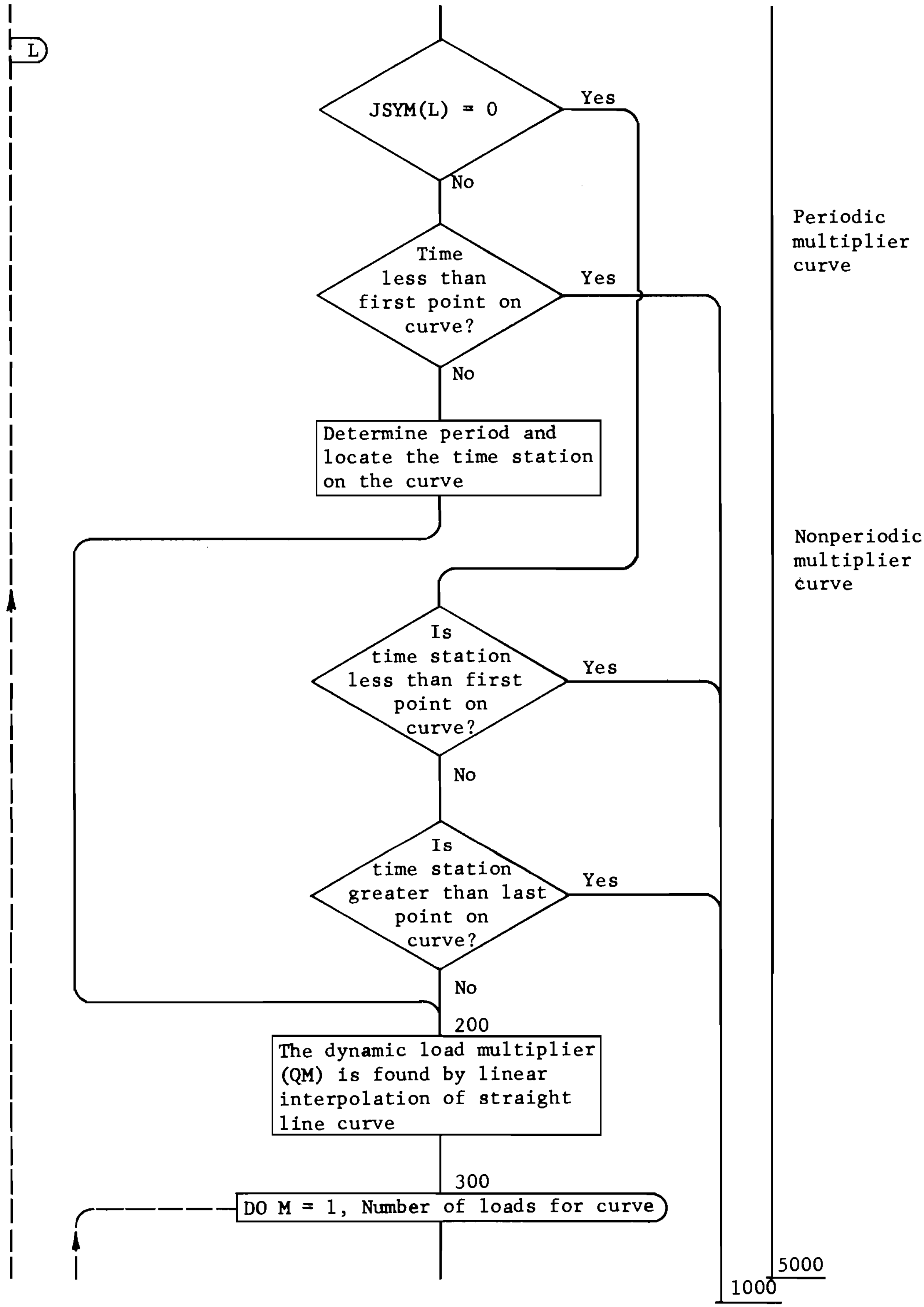
Write correction
load on File 14

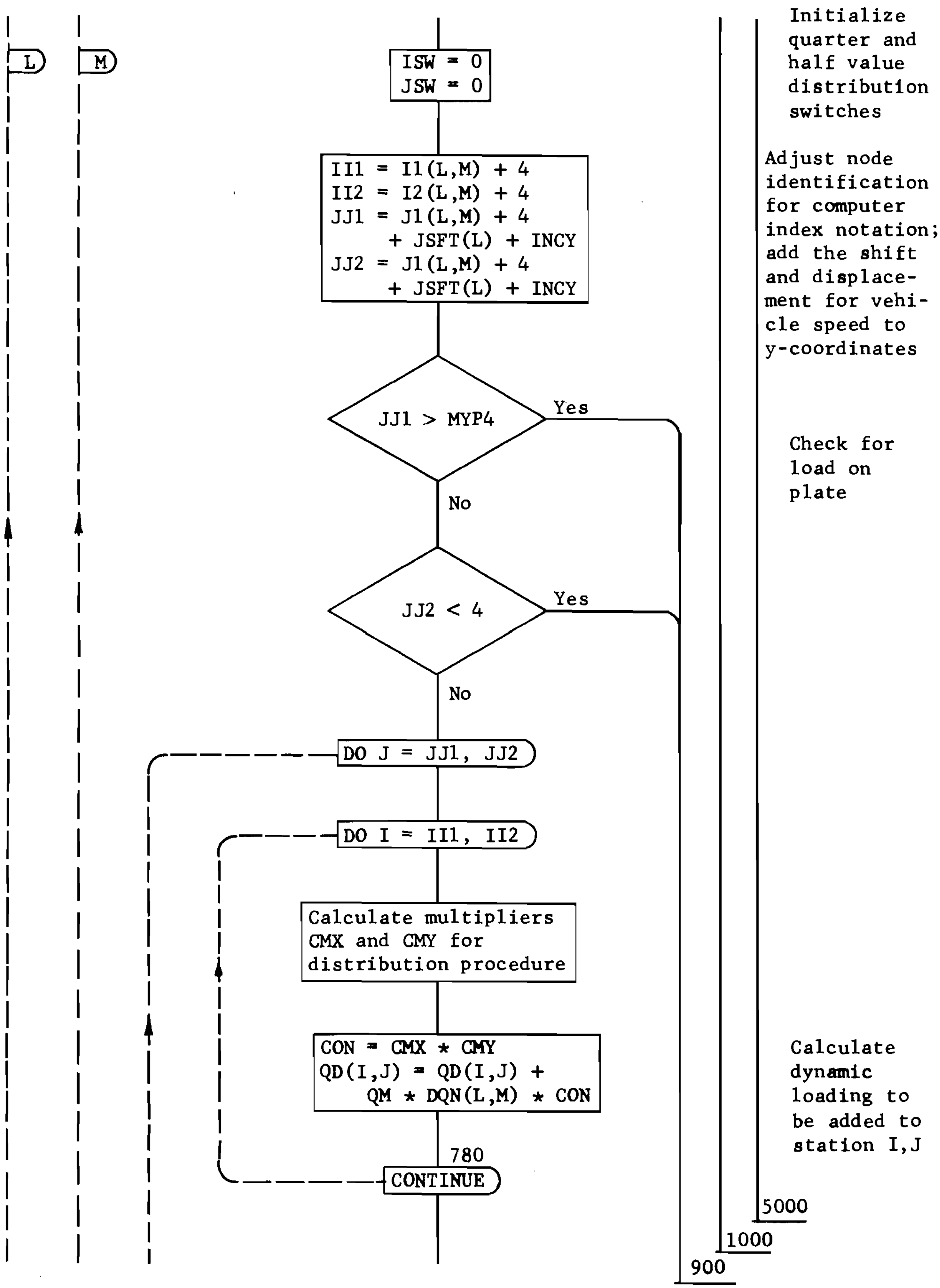
```
      SUBROUTINE NONLIN 4 ( NCRV, TOL,                                  02DE8
     1                      IS1, IS2, JS1, JS2, QN, WN, SFN, NPC,       02DE8
     2                      W, WW, WMAX, L1, L2, L3, L8, L9,            11NO8
     3                      QI, IPSW, IPSD, NCOUNT  )                   11NO8
      DIMENSION  IS1( L8),      IS2( L8),      JS1( L8),      JS2( L8), 11NO8
     1           NPC( L8),      SFN( L8),                               11NO8
     2            QN( L8, L9),   WN( L8, L9),                           11NO8
     3             W( L2, L3),   WW( L2, L3),   QI( L2, L3)             11NO8
      COMMON/INCR/ MX, MY, MXP2, MXP3, MXP4, MXP5, MXP7,                11NO8
     1             MYP2, MYP3, MYP4, MYP5, MYP7, MT                     11NO8
          IPSW  = 0                                                     20JE8
          IPSWD = 0                                                     20JE8
      CALL IOBIN(6HREWIND,14)                                           19AG9IO
C * * * * CHECK FOR NONLINEAR DATA                                      30JU9
C                                                                       30JU9
          IF ( NCRV  .GT.  0 )           GO TO 300                      30JU9
              NCOUNT = 0                                                30JU9
   50     DO 200 J = 1, MYP7                                            30JU9
          DO 100 I = 1, MXP7                                            30JU9
              QI(I,J) = 0.0                                             30JU9
  100     CONTINUE                                                      30JU9
  200     CONTINUE                                                      30JU9
          IF ( NCOUNT  .EQ.  0 )         GO TO 2020                     30JU9
          GO TO 1200                                                    30JU9
  300         NCOUNT = 0                                                30JU9
          DO 900 J = 4, MYP4                                            30JU9
          DO 800 I = 4, MXP4                                            30JU9
              DIF = ABS( W(I,J) - WW(I,J) )                             30JU9
          IF ( DIF  .GT.  TOL )          NCOUNT = NCOUNT + 1            30JU9
  800     CONTINUE                                                      30JU9
  900     CONTINUE                                                      30JU9
          IF ( NCOUNT  .GT.  0 )         GO TO 50                       30JU9
          GO TO 1300                                                    30JU9
C * * *   START INTERPOLATION FOR NEW VALUES OF LOAD OR SPRING OR BOTH  17DE9
 1200     DO 2000 I = 1, NCRV                                           11NO8
              NP = NPC(I)                                               20JE8
C * * *   LOCATE COMPUTED DEFLECTION WITH RESPECT TO INPUT O - W CURVE  17DE9
              NX1 = IS1(I) + 4                                          11NO8
              NX2 = IS2(I) + 4                                          11NO8
              NY1 = JS1(I) + 4                                          11NO8
              NY2 = JS2(I) + 4                                          11NO8
              ISW = 0                                                   20JE8
              JSW = 0                                                   20JE8
          IF ( NX2  .GT.  NX1 )    ISW = 1                              20JE8
          IF ( NY2  .GT.  NY1 )    JSW = 1                              20JE8
          DO 1990 J = NX1, NX2                                          20JE8
          DO 1980 K = NY1, NY2                                          20JE8
          WCK = ABS( W(J,K) )                                           20JE8
          IF (  WCK    .GT.  WMAX )      IPSWD = 1                      20JE8
              KSW = 0                                                   20JE8
          DO 1970 L = 2, NP                                             20JE8
          IF ( W(J,K)  .GT.  WN(I,L) )  GO TO 1970                      20JE8
          IF ( L  .GT.  2 )          GO TO 1205                         20JE8
              KSW = 1                                                   20JE8
C * * *   CHECK FOR POINT OFF LEFT END OF CURVE                         17DE9
          IF ( W(J,K)  .LT.  WN(I,1) )  IPSW = 1                        20JE8
 1205         WC = W(J,K) - WN(I,L)                                     20JE8
              S1 = ( QN(I,L-1) - QN(I,L) ) / ( WN(I,L-1) - WN(I,L) )    20JE8
 1210         CMX = 1.0                                                 20JE8
 1211         CMY = 1.0                                                 20JE8
C * * *   COMPUTE Q FOR NEXT SOLUTION - HALF VALUE AT END STATIONS      17DE9
          IF ( ISW  .EQ.  0 )        GO TO 1240                         20JE8
          IF ( JSW  .EQ.  0 )        GO TO 1220                         20JE8
C * * *   AN AREA LOAD IS CALLED FOR - HALF VALUES READ ALONG EDGES
          IF ( J  .EQ.  NX1 )        CMX = 0.5                          20JE8
          IF ( J  .EQ.  NX2 )        CMX = 0.5                          20JE8
          IF ( K  .EQ.  NY1 )        CMY = 0.5                          20JE8
          IF ( K  .EQ.  NY2 )        CMY = 0.5                          20JE8
          GO TO 1250                                                    20JE8
C * * *   LINE LOAD IN X DIRECTION - USE HALF VALUES AT END STATIONS    17DE9
 1220     IF ( J  .EQ.  NX1 )        CMX = 0.5                          20JE8
          IF ( J  .EQ.  NX2 )        CMX = 0.5                          20JE8
          GO TO 1250                                                    20JE8
C * * *   CHECK FOR LINE LOAD IN Y DIRECTION - HALF VALUES AT END STA.  17DE9
 1240     IF ( JSW  .EQ.  0 )        GO TO 1250                         20JE8
          IF ( K  .EQ.  NY1 )        CMY = 0.5                          20JE8
          IF ( K  .EQ.  NY2 )        CMY = 0.5                          20JE8
 1250     CONTINUE                                                      20JE8
              CON = CMX * CMY                                           28MY9
C * * *   LOAD ITERATION METHOD, NO PARENT PROBLEM IS REQUIRED
 1500         QS = - W(J,K) * SFN(I)                                    20JE8
              QC =  QN(I,L) + WC * S1                                   20JE8
              QI(J,K) = QI(J,K) + CON * ( QC - QS )                     20JE8
          GO TO 1980                                                    20JE8
 1970     CONTINUE                                                      20JE8
          IF ( KSW  .EQ.  1 )        GO TO 1980                         20JE8
              IPSW = 1                                                  20JE8
              WC = W(J,K) - WN(I,NP)                                    20JE8
              S1 = ( QN(I,NP-1) - QN(I,NP) ) / ( WN(I,NP-1) - WN(I,NP))20JE8
              L = NP                                                    20JE8
          GO TO 1210                                                    20JE8
 1980     CONTINUE                                                      20JE8
 1990     CONTINUE                                                      20JE8
 2000     CONTINUE                                                      20JE8
 2020     CONTINUE                                                      02AP9
          DO 1255 J = 3, MYP5                                           03AP9
      CALL IOBIN(5HWRITE ,14,QI(3,J),MXP3)                              19AG9IO
 1255     CONTINUE                                                      02DE8
      CALL IOBIN (6HWRITER,14)                                          22AG9IO
 1300     CONTINUE                                                      11NO8
      RETURN                                                            20JE8
      END                                                               20JE8
C
```

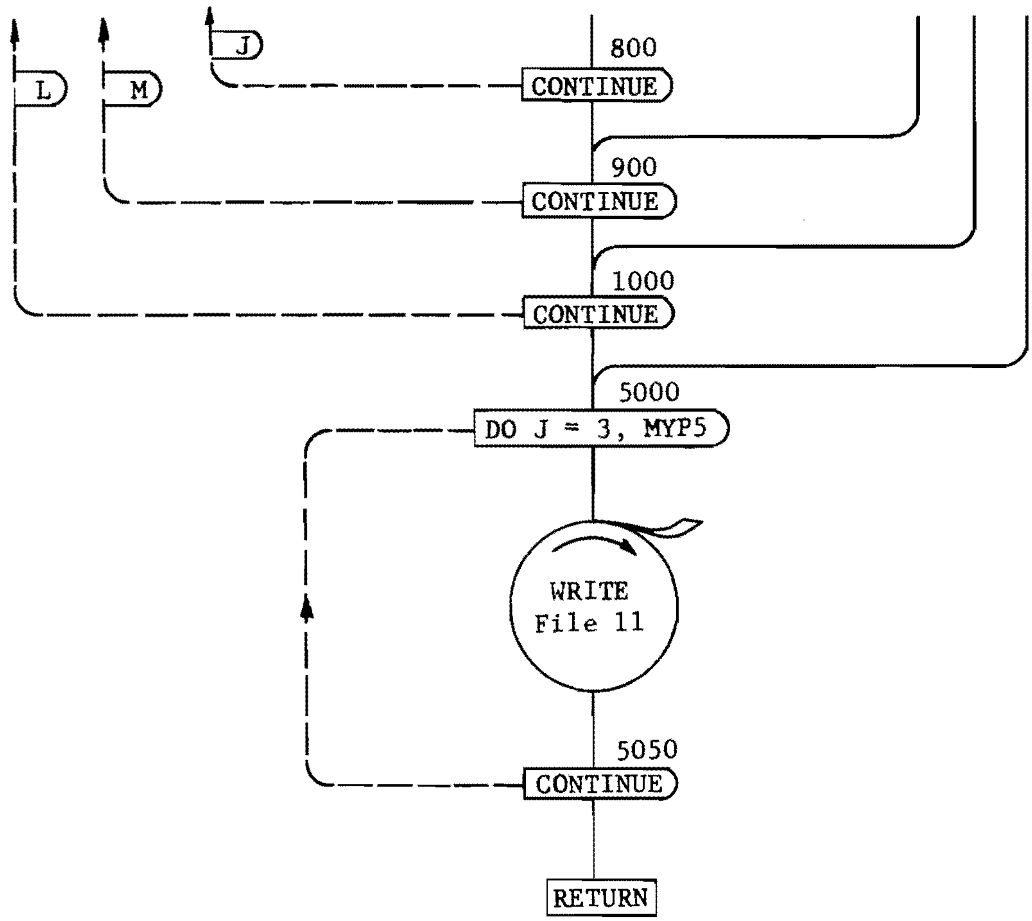
SUBROUTINE DYNLD

This subroutine computes the dynamic loading for each time station.




L
JSYM(L) = 0
Yes
No
Time less than first point on curve?
Yes
No
Determine period and locate the time station on the curve
Is time station less than first point on curve?
Yes
No
Is time station greater than last point on curve?
Yes
No
200
The dynamic load multiplier (QM) is found by linear interpolation of straight line curve
300
DO M = 1, Number of loads for curve
Periodic multiplier curve
Nonperiodic multiplier curve
5000
1000


L
M
ISW = 0
JSW = 0
Initialize quarter and half value distribution switches
II1 = I1(L,M) + 4
II2 = I2(L,M) + 4
JJ1 = J1(L,M) + 4 + JSFT(L) + INCY
JJ2 = J1(L,M) + 4 + JSFT(L) + INCY
Adjust node identification for computer index notation; add the shift and displacement for vehicle speed to y-coordinates
JJ1 > MYP4
Yes
No
Check for load on plate
JJ2 < 4
Yes
No
DO J = JJ1, JJ2
DO I = II1, II2
Calculate multipliers CMX and CMY for distribution procedure
CON = CMX * CMY
QD(I,J) = QD(I,J) + QM * DQN(L,M) * CON
Calculate dynamic loading to be added to station I,J
780
CONTINUE
5000
1000
900



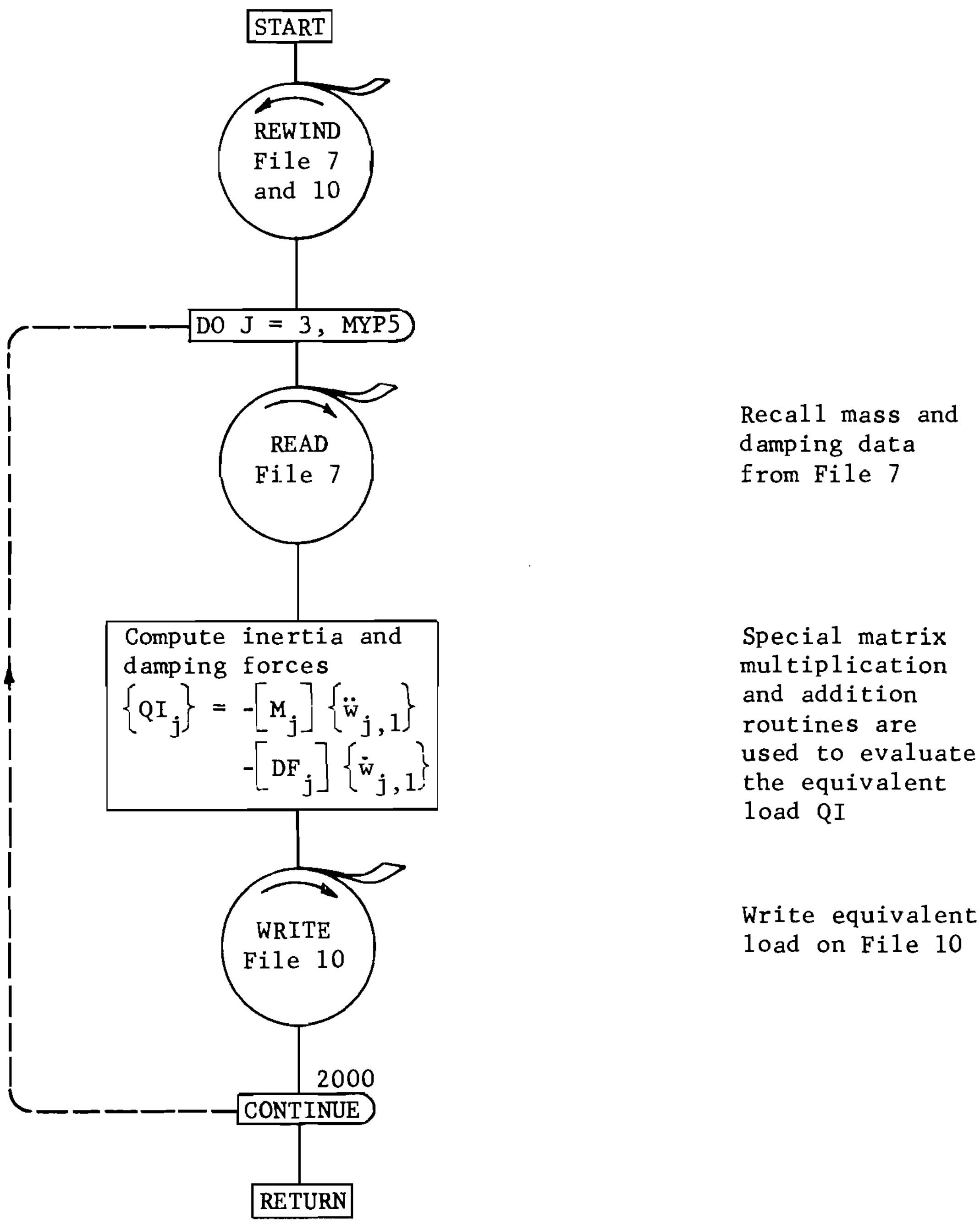
Write dynamic loading for new time step on File 11

```
      SUBROUTINE DYNLD ( NAM, JSFT, MSPD, JSYM, K1, K2, KONT, DQM, NDL, 19MY9
     1                   I1, J1, I2, J2, DQN, QD, NCR6, L1, L2, L3)     01SE8
      DIMENSION  NAM( 20 ),     JSFT( 20 ),     MSPD( 20 ),     JSYM( 20 ),01SE8
     1            K1( 20, 20),    K2( 20, 20), KONT( 20, 20), NDL( 20),  19MY9
     2           DQM( 20, 20),    I1( 20,20),     J1( 20,20), I2( 20,20), 19MY9
     3            J2( 20,20),    DQN( 20,20),                            19MY9
     4            QD( L2,L3 )                                           01SE8
      COMMON/INCR/ MX, MY, MXP2, MXP3, MXP4, MXP5, MXP7,                11NO8
     I             MYP2, MYP3, MYP4, MYP5, MYP7, HT, HT, HY             20JU9
      COMMON/RI/    NK, NL, NF, NT2SW, TIM                              22JA9
      TYPE INTEGER TIM                                                  11NO8
      TYPE REAL MSPD, KNTR, K1, K2                                      29JU0
C                                                                        1SE8
C * * *   THIS SUBROUTINE FORMS THE DYNAMIC LOAD VECTOR FOR TIME ICT     01SE8
C                    INITIALIZE DYNAMIC LOAD VECTOR                     01SE8
C                                                                        1SE8
      CALL IOBIN(6HREWIND,11)                                           19AG910
          DO 100 I = 1, MXP7                                            01SE8
          DO  50 J = 1, MYP7                                            01SE8
              QD(I,J) = 0.0                                             01SE8
   50     CONTINUE                                                      01SE8
  100     CONTINUE                                                      01SE8
C                                                                        1SE8
C * * *   CHECK FOR DYNAMIC LOAD                                        01SE8
C                                                                        1SE8
          IF ( NCR6  .EQ.  0 )            GO TO 5000                    01SE8
          IF ( TIM .EQ. 1 )               GO TO 5000                    29JA9
C                                                                        1SE8
C * * *   COMPUTE DYNAMIC LOAD MULTIPLIERS                              01SE8
C                                                                        1SE8
              ICT = TIM - 1                                             29JA9
          DO 1000 L = 1, NCR6                                           01SE8
              NM = NAM(L)                                               01SE8
              QM = 0.0                                                  01SE8
              DTIM  =  ICT * HT                                         29JU0
              KNTR  =  DTIM                                             29JU0
              ND = NDL(L)                                               19MY9
              TIMP = 5280. / 300.                                       13JE9
              SEC = ICT * HT                                            03JL9
              SPED = MSPD(L) * TIMP                                     13JE9
              DIST = SEC * SPED                                         13JE9
              YSTA = DIST / HY                                          13JE9
              INCY = YSTA                                               13JE9
              YPRT = YSTA - INCY                                        13JE9
C                                                                        1SE8
C * * *   CHECK FOR PERIODIC LOADING                                    01SE8
C                                                                        1SE8
          IF ( JSYM(L)  .EQ.  0 )         GO TO 120                     01SE8
          IF ( DTIM  .LT.  K1(L,1))       GO TO 1000                    29JU0
C                                                                        1SE8
C * * *   PERIOD OF DYNAMIC LOADING IN TIME STATIONS                    01SE8
C                                                                        1SE8
              KPER = K2(L,NM) - K1(L,1)                                 01SE8
  110     IF ( KNTR  .LT.  K2(L,NM) )     GO TO 200                     01SE8
              KNTR = KNTR - KPER                                        01SE8
          GO TO 110                                                     01SE8
C                                                                        1SE8
C * * *   CHECK TIME STATION IN RANGE OF Q-T CURVE                      01SE8
C                                                                        1SE8
  120     IF ( K1(L,1)  - KNTR ) 140, 200, 1000                         01SE8
  140     IF ( K2(L,NM) - KNTR ) 1000, 200, 200                         01SE8
C                                                                        1SE8
C * * *   COMPUTE AMPLITUDE MULTIPLIER, QM                              01SE8
C                                                                        1SE8
  200         KS1 = 0                                                   01SE8
          DO 250 M = 1, NM                                              01SE8
              KSW = 2 * KS1 + 1 + KONT(L,M)                             01SE8
              KS1 = KONT(L,M)                                           30JA9
          GO TO ( 210, 220, 250, 230 ), KSW                             01SE8
  210     IF ( KNTR  .GT.  K2(L,M) )      GO TO 250                     01SE8
              QM = DQM(L,M)                                             01SE8
          GO TO 300                                                     01SE8
  220     IF ( KNTR  .GT.  K2(L,M+1) )  GO TO 250                       01SE8
              QM1 = DQM(L,M)                                            01OC9
              QM2 = DQM(L,M+1)                                          01OC9
              DIF1 = K2(L,M+1) - K1(L,M)                                01OC9
              DIF2 = KNTR - K1(L,M)                                     01SE8
              QM = QM1 + ( QM2 - QM1 ) * DIF2 / DIF1                    01SE8
          GO TO 300                                                     01SE8
  230     IF ( KNTR  .GT.  K2(L,M+1) )  GO TO 250                       01SE8
              QM1 = DQM(L,M)                                            01SE8
              QM2 = DQM(L,M+1)                                          01SE8
              DIF1 = K2(L,M+1) - K2(L,M)                                01SE8
              DIF2 = KNTR - K2(L,M)                                     01SE8
              QM = QM1 + ( QM2 - QM1 ) * DIF2 / DIF1                    01SE8
          GO TO 300                                                     01SE8
  250     CONTINUE                                                      01SE8
C                                                                        1SE8
C * * *   COMPUTE POSITION AND VALUE OF DYNAMIC LOAD                    01SE8
C                                                                        1SE8
  300     CONTINUE                                                      19MY9
          DO 900 M=1,ND                                                 19MY9
              ISW = 0                                                   19MY9
              JSW = 0                                                   01SE8
              II1 = I1(L,M) + 4                                         02JU9
              II2 = I2(L,M) + 4                                         02JU9
              JJ1 = J1(L,M) + 4 + JSFT(L) + INCY                        13JU9
              JJ2 = J2(L,M) + 4 + JSFT(L) + INCY                        13JU9
          IF ( II2  .GT.  II1 )           ISW = 1                       01SE8
          IF ( JJ2  .GT.  JJ1 )           JSW = 1                       01SE8
              JJSW  =  0                                                29JU0
          IF (YPRT )  305, 315, 310                                     29JU0
C * * *   YPRT NEGATIVE, LOAD MOVING IN NEGATIVE Y, JJ1 = JJ1 - 1       29JU0
  305         JJSW  =  1                                                29JU0
              JJ1   =  JJ1 - 1                                          29JU0
          GO TO 315                                                     29JU0
C * * *   YPRT POSITIVE, LOAD MOVING IN POSITIVE Y, JJ2 = JJ2 + 1       29JU0
  310         JJSW  =  1                                                29JU0
              JJ2   =  JJ2 + 1                                          29JU0
  315     CONTINUE                                                      29JU0
          IF ( JJ1  .GT. MYP4 )           GO TO 900                     06JU9
          IF ( JJ2  .LT.  4 )             GO TO 900                     06JU9
```

```
C                                                                        1SE8
C * * *   SET INDICES FOR LOAD DISTRIBUTION                             01SE8
C                                                                        1SE8
                DIF    =  ABS ( YPRT )                                  29JU0
C                                                                        1SE8
          DO 800 I = III, II2                                           01SE8
          DO 780 J = JJ1, JJ2                                           01SE8
                CPRT   =  1.0                                           29JU0
                CMX = 1.0                                               01SE8
                CMY = 1.0                                               01SE8
          IF ( J - 4 )    780, 330, 325                                 29JL0
  325     IF ( J- MYP4 ) 330, 330, 780                                  29JL0
  330     CONTINUE                                                      29JU0
          IF ( ISW  .EQ.  0 )                GO TO 400                  29JU0
          IF ( JSW  .EQ.  0 )                GO TO 500                  29JU0
C * * *   AREA LOAD                                                     29JU0
          IF ( JJSW  .EQ.  1 )               GO TO 350                  29JU0
          IF ( J  .EQ.  JJ1 )                CMY  =  0.5                08JL0
          IF ( J  .EQ.  JJ2 )                CMY  =  0.5                29JU0
  340     IF ( I  .EQ.  III )                CMX  =  0.5                29JU0
          IF ( I  .EQ.  II2 )                CMX  =  0.5                29JU0
          GO TO 600                                                     29JU0
C * * *   DISTRIBUTION FOR MOVING AREA LOAD                             29JU0
  350     IF ( MSPD  .GT.  0.0 )             GO TO 380                  29JU0
          GO TO 390                                                     29JU0
  360           CMY    =  0.5                                           29JU0
                CPRT   =  1.0 - ( DIF / HY )                            29JU0
          GO TO 340                                                     29JU0
  370           CPRT   =  1.0 - 0.5*(DIF/HY)                            10JL0
          GO TO 340                                                     29JU0
  375           CMY    =  0.5                                           10JL0
                CPRT   =  DIF/HY                                        10JL0
          GO TO 340                                                     10JL0
  365           CPRT   =  0.5 + 0.5*(DIF/HY)                            10JL0
          GO TO 340                                                     10JL0
  380     IF ( J  .EQ.  JJ1 )                GO TO 360                  29JU0
          IF ( J  .EQ.  JJ1 + 1 )            GO TO 370                  29JU0
          IF ( J  .EQ.  JJ2 )                GO TO 375                  10JL0
          IF ( J  .EQ.  JJ2 - 1 )            GO TO 365                  10JL0
          GO TO 340                                                     29JU0
  390     IF ( J  .EQ.  JJ1 )                GO TO 375                  10JL0
          IF ( J  .EQ.  JJ1 + 1 )            GO TO 365                  10JL0
          IF ( J  .EQ.  JJ2 )                GO TO 360                  29JU0
          IF ( J  .EQ.  JJ2 - 1 )            GO TO 370                  29JU0
          GO TO 340                                                     29JU0
C * * *   CHECK FOR DISTRIBUTION IN Y DIRECTION                         29JU0
  400     IF ( JSW  .EQ.  0 )                GO TO 550                  29JU0
C * * *   LINE LOADING IN Y DIRECTION                                   29JU0
          IF ( JJSW  .EQ.  1 )               GO TO 430                  08JL0
          IF ( J  .EQ.  JJ1 )                CMY  =  0.5                29JU0
          IF ( J  .EQ.  JJ2 )                CMY  =  0.5                29JU0
          GO TO 600                                                     29JU0
C * * *   DISTRIBUTION FOR MOVING Y - LINE LOAD                         29JU0
  430     IF ( MSPD  .GT.  0.0 )             GO TO 460                  29JU0
          GO TO 480                                                     29JU0
  440           CMY    =  0.5                                           29JU0
                CPRT   =  1.0 - ( DIF / HY )                            29JU0
          GO TO 600                                                     29JU0
  450           CMY    =  0.5                                           29JU0
                CPRT   =  DIF / HY                                      02JL0
          GO TO 600                                                     02JL0
  460     IF ( J  .EQ.  JJ1 )                GO TO 440                  02JL0
          IF ( J  .EQ.  JJ1 + 1 )            GO TO 450                  02JL0
          IF ( J  .EQ.  JJ2 )                GO TO 450                  02JL0
          IF ( J  .EQ.  JJ2 - 1 )            GO TO 440                  02JL0
          GO TO 600                                                     02JL0
  480     IF ( J  .EQ.  JJ1 )                GO TO 450                  02JL0
          IF ( J  .EQ.  JJ1 + 1 )            GO TO 440                  02JL0
          IF ( J  .EQ.  JJ2 )                GO TO 440                  02JL0
          IF ( J  .EQ.  JJ2 - 1 )            GO TO 450                  02JL0
          GO TO 600                                                     02JL0
C * * *   DISTRIBUTION FOR LINE LOAD IN X DIRECTION                     02JL0
  500     IF ( JJSW  .EQ.  1 )               GO TO 505                  02JL0
  502     IF ( I       .EQ.  III )          CMX  =  0.5                 08JL0
          IF ( I       .EQ.  II2 )          CMX  =  0.5                 08JL0
          GO TO 600                                                     02JL0
  505     IF ( MSPD  .GT.  0.0 )             GO TO 515                  02JL0
          GO TO 525                                                     02JL0
  508           CPRT   =  DIF / HY                                      02JL0
          GO TO 502                                                     02JL0
  510     CPRT = 1.0 - (DIF/HY)                                         08JL0
          GO TO 502                                                     08JL0
  515     IF ( J     .EQ.  JJ1 )             GO TO 510                  08JL0
          IF ( J     .EQ.  JJ2 )             GO TO 508                  08JL0
  525     IF ( J     .EQ.  JJ1 )             GO TO 508                  08JL0
          IF ( J     .EQ.  JJ2 )             GO TO 510                  08JL0
C * * *   DISTRIBUTION FOR CONCENTRATED LOAD                            02JL0
  550     IF ( JJSW  .EQ.  1 )               GO TO 560                  02JL0
          GO TO 600                                                     02JL0
  560     IF (MSPD  .GT.  0.0 )               GO TO 570                 08JL0
          GO TO 575                                                     02JL0
  562           CPRT   =  1.0 - ( DIF / HY )                            02JL0
          GO TO 600                                                     02JL0
  565           CPRT   =  DIF / HY                                      02JL0
          GO TO 600                                                     02JL0
  570     IF ( J    .EQ.  JJ1 )              GO TO 562                  02JL0
          GO TO 565                                                     02JL0
  575     IF ( J    .EQ.  JJ1 )              GO TO 565                  02JL0
          GO TO 562                                                     02JL0
C * * *   CALCULATE CONTRIBUTION OF DYNAMIC LOAD TO STATION I,J         02JL0
  600           CON   =  CMX * CMY * CPRT                               02JL0
                QD(I,J) = QD(I,J) + QM * DQN(L,M) * CON                 19MY9
  780     CONTINUE                                                      01SE8
  800     CONTINUE                                                      01SE8
  900     CONTINUE                                                      19MY9
 1000     CONTINUE                                                      01SE8
 5000     DO 5050  J = 3,MYP5                                           29JA9
      CALL IOBIN(5HWRITE ,11,QD(3,J),MXP3)                              25AG910-
 5050     CONTINUE                                                      29JA9
      CALL IOBIN (6HWRITER,11)                                          22AG910
 5060 RETURN                                                            29JA9
      END                                                               29JA9
```

SUBROUTINE MASSAC

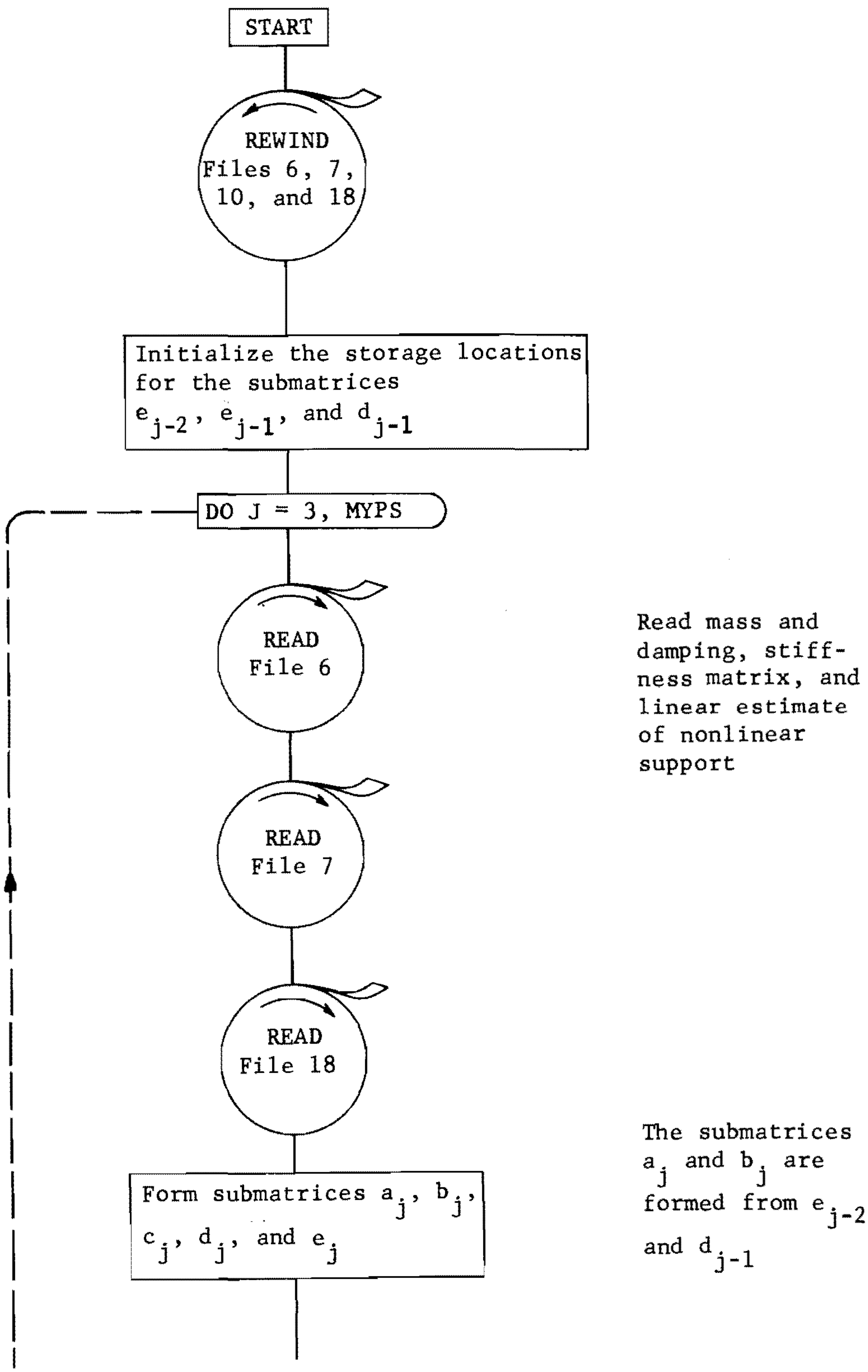
The product of mass and acceleration, is added to the product of viscous damping times velocity, to calculate the equivalent load vector for deflection analysis at the first-time step.

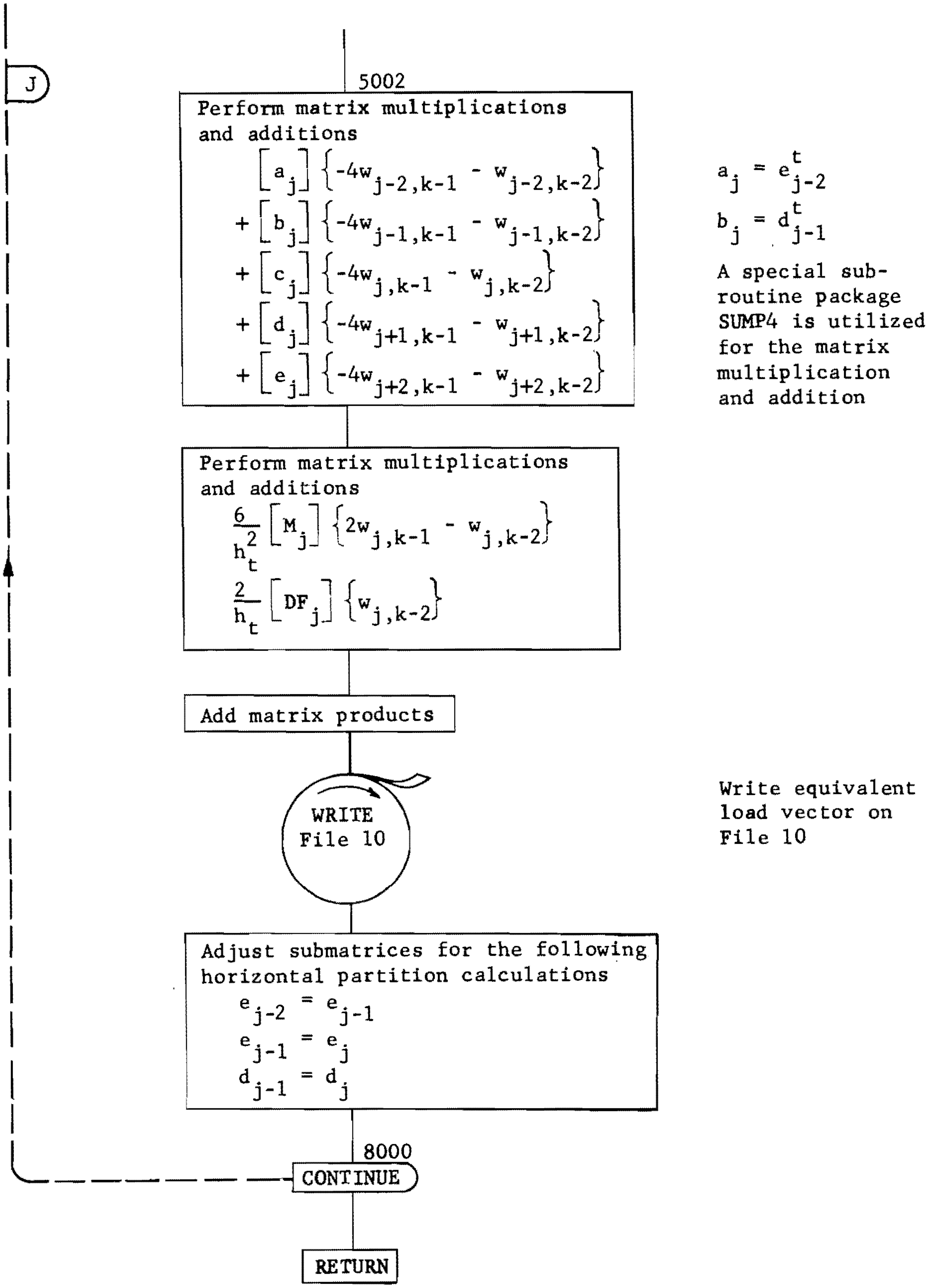


```
      SUBROUTINE MASSAC ( L1, L2, L3, W, RHO, DF, QI                    )      01SE8
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     THIS SUBROUTINE FORM THE PRODUCT OF MASS TIMES ACCELERATION AND          *
C         VISCOUS DAMPING TIMES VELOCITY TO ADD TO THE RHS OF THE              *
C         EQUILIBRIUM EQUATION FOR THE FIRST TIME STEP. THE  PROBLEM           *
C         WHICH IS THEN SOLVED IS.                                             *
C                                                                              *
C                       K*W = Q - RHO*DDW - DF*DW                              *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      DIMENSION   W( L2, L3 ),        RHO(   L1    ),        DF(   L1    ), 01SE8
     1            QI(    L1    )                                            01SE8
      COMMON/INCR/ MX, MY, MXP2, MXP3, MXP4, MXP5, MXP7,                    11NO8
     1             MYP2, MYP3, MYP4, MYP5, MYP7, MT                         11NO8
      COMMON/CON/  HXDHY3, HYDHX3, ODHXHY, ODHX, ODHY, PR, ODHT2, OD2HT,06JU9
     1             HXDHY, HYDHX                                             06JU9
      COMMON/RI/   NK, NL, NF, NT2SW, TIM                                   29JA9
      CALL IOBIN(6HREWIND,7)                                                19AG9IO
      CALL IOBIN(6HREWIND,10)                                               19AG9IO
C                                                                            1SE8
C * * *   COMPUTE MULTIPLIER FOR CONVERTING ACCELERATION TO VFLOCITY        01SE8
C                                                                            1SE8
              VMP = 0.25 / OD2HT                                            29JA9
          DO 2000 J = 3, MYP5                                               01SE8
      CALL IOBIN(4HREAD   ,7,RHO,MXP3)                                      25AG9IO
      CALL IOBIN(4HREAD   ,7,DF,MXP3)                                       25AG9IO-
      CALL MBFV ( RHO, W(3,J), QI, L1,  1,NK, 1 )                           17DE9
      CALL CFV  ( QI, L1, 1, NK, -1.0 )                                     17DE9
      CALL CFV  ( W(3,J), L1, 1, NK, VMP )                                  17DE9
      CALL MBFV ( DF, W(3,J), DF, L1,  1,NK, 1 )                            17DE9
      CALL ASFV ( QI, DF, QI, L1, 1, NK, -1.0 )                             17DE9
      CALL IOBIN(5HWRITE ,10,QI,MXP3)                                       30SE9IO-
 2000     CONTINUE                                                          01SE8
      CALL IOBIN (6HWRITER,10)                                              30SE9IO
      RETURN                                                                 1SE8
      END                                                                    1SE8
C
```

SUBROUTINE INERTIA

The stiffness, damping, and mass matrices are multiplied by the deflection vectors for times k-1 and k-2 to form a portion of the equivalent load vector for the following time step.



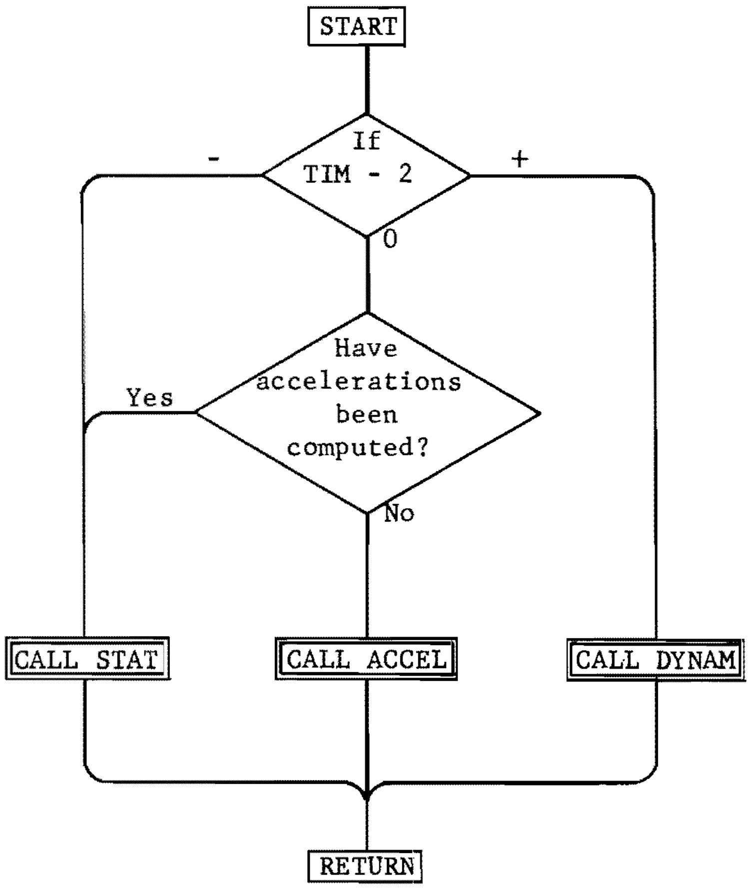

J
5002
Perform matrix multiplications and additions
$[a_j]\{-4w_{j-2,k-1} - w_{j-2,k-2}\}$
$+ [b_j]\{-4w_{j-1,k-1} - w_{j-1,k-2}\}$
$+ [c_j]\{-4w_{j,k-1} - w_{j,k-2}\}$
$+ [d_j]\{-4w_{j+1,k-1} - w_{j+1,k-2}\}$
$+ [e_j]\{-4w_{j+2,k-1} - w_{j+2,k-2}\}$
$a_j = e^t_{j-2}$
$b_j = d^t_{j-1}$
A special subroutine package SUMP4 is utilized for the matrix multiplication and addition
Perform matrix multiplications and additions
$\frac{6}{h_t^2}[M_j]\{2w_{j,k-1} - w_{j,k-2}\}$
$\frac{2}{h_t}[DF_j]\{w_{j,k-2}\}$
Add matrix products
WRITE File 10
Write equivalent load vector on File 10
Adjust submatrices for the following horizontal partition calculations
$e_{j-2} = e_{j-1}$
$e_{j-1} = e_j$
$d_{j-1} = d_j$
8000
CONTINUE
RETURN

```
      SUBROUTINE INERTIA ( L1, L2, L3, W, WTM1, WTM2, AA, BB, CC, DD,     01SE8
     1   EE, EEM1, EEM2, DDM1, SK, N1, N2, N3, SF )                         22AG9
      DIMENSION    W( L2, L3),        WTM1( L2, L3),      WTM2( L2, L3),   01SE8
     1            AA(  L1    ),          BB( L1, N2),        CC(L1, N3),   01SE8
     2            DD( L1, N2),          EE( L1, N1),       EEM1(  L1  ),   30JA9
     3  EEM2(  L1 ), DDM1( L1,N2 ), SK( L1, 9 ), SF( L1 )                  22AG910
      COMMON/INCR/ MX, MY, MXP2, MXP3, MXP4, MXP5, MXP7,                  11NO8
     1            MYP2, MYP3, MYP4, MYP5, MYP7, MT                         11NO8
      COMMON/CON/  HXDHY3, HYDHX3, ODHXHY, ODHX, ODHY, PR, ODHT2, OD2HT,06JU9
     1            HXDHY, HYDHX                                             06JU9
      COMMON/RI/   NK, NL, NF, NT2SW, TIM                                 29JA9
      TYPE REAL TIM                                                        19JL9
      CALL IOBIN(6HREWIND,6)                                               19AG910
      CALL IOBIN(6HREWIND,7)                                               19AG910
      CALL IOBIN(6HREWIND,10)                                              19AG910
      CALL IOBIN(6HREWIND,18)                                              22AG910
          DO 500 I = 1, MXP3                                               01SE8
               EEM2(I) = 0.0                                               01SE8
               EEM1(I) = 0.0                                               01SE8
               DDM1(I,1) = 0.0                                             01SE8
               DDM1(I,2) = 0.0                                             01SE8
               DDM1(I,3) = 0.0                                             01SE8
  500     CONTINUE                                                         01SE8
               RMP = 6.0 * ODHT2                                           30JU9
               DMP = 6.0 * OD2HT                                           30JU9
C                                                                           1SE8
C * * * * FORM PARTITIONED ROW J OF STIFFNESS MATRIX AND MULTIPLY BY       01SE8
C * * * *      APPROPRIATE DEFLECTION VECTOR                                01SE8
C                                                                           1SE8
          DO 8000 J = 3, MYP5                                              01SE8
               MTK = 9 * MXP3                                              19AG910
      CALL IOBIN(4HREAD   ,6,SK,MTK)                                       25AG910-
      CALL IOBIN(4HREAD   ,18,SF,MXP3)                                     25AG910-
          DO 5000 I = 1, MXP3                                              01SE8
               AA(I) = EEM2(I)                                             01SE8
               EE(I) = SK(I,9)                                             01SE8
               SK(I,3) = SK(I,3) + SF(I)                                   22AG910
          IF ( I  .EQ.  1 )              GO TO 1000                        01SE8
          IF ( I  .EQ.  2 )              GO TO 2000                        01SE8
          IF ( I  .EQ.  MXP2 )           GO TO 3000                        01SE8
          IF ( I  .EQ.  MXP3 )           GO TO 4000                        01SE8
C                                                                           1SE8
C * * *   FORM MATRICES AT GENERAL INTERIOR STATION                        01SE8
C                                                                           1SE8
               CC(I,1) = SK(I,1)                                           01SE8
               CC(I,2) = SK(I,2)                                           01SE8
               CC(I,3) = SK(I,3)                                           01SE8
               CC(I,4) = SK(I,4)                                           01SE8
               CC(I,5) = SK(I,5)                                           01SE8
C                                                                           1SE8
               BB(I,1) = DDM1(I-1,3)                                       10JL9
               BB(I,2) = DDM1(I,2)                                         01SE8
               BB(I,3) = DDM1(I+1,1)                                       01SE8
C                                                                           1SE8
               DD(I,1) = SK(I,6)                                           01SE8
               DD(I,2) = SK(I,7)                                           01SE8
               DD(I,3) = SK(I,8)                                           01SE8
          GO TO 5000                                                       01SE8
C                                                                           1SE8
C * * *   FORM MATRIX AT I = 1                                             01SE8
C                                                                           1SE8
 1000          CC(I,1) = SK(I,3)                                           01SE8
               CC(I,2) = SK(I,4)                                           01SE8
               CC(I,3) = SK(I,5)                                           01SE8
               CC(I,4) = 0.0                                               01SE8
               CC(I,5) = 0.0                                               01SE8
C                                                                           1SE8
               BB(I,1) = DDM1(I,1)                                         01SE8
               BB(I,2) = DDM1(I+1,1)                                       01SE8
               BB(I,3) = 0.0                                               01SE8
C                                                                           1SE8
               DD(I,1) = SK(I,7)                                           01SE8
               DD(I,2) = SK(I,8)                                           01SE8
               DD(I,3) = 0.0                                               01SE8
          GO TO 5000                                                       01SE8
C                                                                           1SE8
C * * *   FORM MATRIX AT I = 2                                             01SE8
C                                                                           1SE8
 2000          CC(I,1) = SK(I,2)                                           01SE8
               CC(I,2) = SK(I,3)                                           01SE8
               CC(I,3) = SK(I,4)                                           01SE8
               CC(I,4) = SK(I,5)                                           01SE8
               CC(I,5) = 0.0                                               17DE9
C                                                                           16JL9
               BB(I,1) = DDM1(I-1,2)                                       10JL9
               BB(I,2) = DDM1(I,2)                                         10JL9
               BB(I,3) = DDM1(I+1,1)                                       14JL9
C                                                                           14JL9
               DD(I,1) = SK(I,6)                                           14JL9
               DD(I,2) = SK(I,7)                                           14JL9
               DD(I,3) = SK(I,8)                                           10JL9
          GO TO 5000                                                        1SE8
C                                                                          01SE8
C * * *   FORM MATRIX AT I = MXP2                                           1SE8
C                                                                          01SE8
 3000          CC(I,1) = 0.0                                               01SE8
               CC(I,2) = SK(I,1)                                           01SE8
               CC(I,3) = SK(I,2)                                           01SE8
               CC(I,4) = SK(I,3)                                           01SE8
               CC(I,5) = SK(I,4)                                           17DE9
C                                                                          10JL9
               BB(I,1) = DDM1(I-1,3)                                       10JL9
               BB(I,2) = DDM1(I,2)                                         10JL9
               BB(I,3) = DDM1(I+1,2)                                       14JL9
C                                                                          14JL9
               DD(I,1) = SK(I,6)                                           14JL9
               DD(I,2) = SK(I,7)                                           14JL9
               DD(I,3) = SK(I,8)                                           10JL9
          GO TO 5000                                                        1SE8
C                                                                          01SE8
C * * *   FORM MATRIX AT I = MXP3                                           1SE8
C
```

```
 4000          CC(I,1) = 0.0                                                   01SE8
               CC(I,2) = 0.0                                                   01SE8
               CC(I,3) = SK(I,1)                                               29JA9
               CC(I,4) = SK(I,2)                                               01SE8
               CC(I,5) = SK(I,3)                                               01SE8
C                                                                               1SE8
               BB(I,1) = 0.0                                                   29JA9
               BB(I,2) = DDM1(I-1,3)                                           01SE8
               BB(I,3) = DDM1(I,3)                                             01SE8
C                                                                               1SE8
C                                                                               1SE8
               DD(I,1) = 0.0                                                   01SE8
               DD(I,2) = SK(I,6)                                               01SE8
               DD(I,3) = SK(I,7)                                               01SE8
 5000     CONTINUE                                                             01SE8
C                                                                               1SE8
C * * *   STIFFNESS MATRIX HAS BEEN FORMED, READ RHO INTO 1ST COLUMN            01SE8
C * * *        OF SK AND DF INTO SECOND COLUMN OF SK                            01SE8
C                                                                               1SE8
      CALL IOBIN(4HREAD   ,7,SK,MXP3)                                          25AG9IO-
      CALL IOBIN(4HREAD   ,7,SK(1,2),MXP3)                                     25AG9IO-
C                                                                               1SE8
C * * *   FORM PRODUCT OF K TIMES ( - 4*WTM1 - WTM2 )                          01SE8
C                                                                               1SE8
      CALL RFV  ( SK(1,3), WTM1(3,J-2), L1, 1, NK )                           26JL0
      CALL CFV  ( SK(1,3), L1, 1, NK, -4.0 )                                  26JL0
      CALL ASFV ( SK(1,3), WTM2(3,J-2), SK(1,3), L1, 1, NK, -1.0 )            26JL0
      CALL MBFV ( AA, SK(1,3), W(3,J), L1,  1,NK, 1 )                         26JL0
      CALL RFV  ( SK(1,3), WTM1(3,J-1), L1, 1, NK )                           26JL0
      CALL CFV  ( SK(1,3), L1, 1, NK, -4.0 )                                  26JL0
      CALL ASFV ( SK(1,3), WTM2(3,J-1), SK(1,3), L1, 1, NK, -1.0 )            26JL0
      CALL MBFV ( BB, SK(1,3), SK(1,4), L1,  1,NK, 3 )                        26JL0
      CALL ASFV ( SK(1,4), W(3,J), W(3,J), L1, 1, NK, +1.0 )                  17DE9
      CALL RFV  ( SK(1,3), WTM1(3,J), L1, 1, NK )                             17DE9
      CALL CFV  ( SK(1,3), L1, 1, NK, -4.0 )                                  17DE9
      CALL ASFV ( SK(1,3), WTM2(3,J), SK(1,3), L1, 1, NK, -1.0 )              17DE9
      CALL MBFV ( CC, SK(1,3), SK(1,4), L1,  1,NK, 5 )                        17DE9
      CALL ASFV ( SK(1,4), W(3,J), W(3,J), L1, 1, NK, +1.0 )                  01JL9
      CALL RFV  ( SK(1,3), WTM1(3,J+1), L1, 1, NK )                           17DE9
      CALL CFV  ( SK(1,3), L1, 1, NK, -4.0 )                                  17DE9
      CALL ASFV ( SK(1,3), WTM2(3,J+1), SK(1,3), L1, 1, NK, -1.0 )            17DE9
      CALL MBFV ( DD, SK(1,3), SK(1,4), L1,  1,NK, 3 )                        17DE9
      CALL ASFV ( SK(1,4), W(3,J), W(3,J), L1, 1, NK, +1.0 )                  17DE9
      CALL RFV  ( SK(1,3), WTM1(3,J+2), L1, 1, NK )                           17DE9
      CALL CFV  ( SK(1,3), L1, 1, NK, -4.0 )                                  17DE9
      CALL ASFV ( SK(1,3), WTM2(3,J+2), SK(1,3), L1, 1, NK, -1.0 )            17DE9
      CALL MBFV ( EE, SK(1,3), SK(1,4), L1,  1,NK, 1 )                        17DE9
      CALL ASFV ( SK(1,4), W(3,J), W(3,J), L1, 1, NK, +1.0 )                  17DE9
C                                                                               1SE8
C * * *   FORM PRODUCT OF RHO TIMES ( 2 * WTM1 - WTM2 )                        27JE9
C * * *             AND DF TIMES - WTM2                                        01SE8
C                                                                               1SE8
      CALL RFV  ( SK(1,3), WTM1(3,J), L1, 1, NK )                             17DE9
      CALL CFV  ( SK(1,3), L1, 1, NK, 2.0 )                                   17DE9
      CALL ASFV ( SK(1,3), WTM2(3,J), SK(1,3), L1, 1, NK, -1.0 )              17DE9
      CALL MBFV ( SK(1,1), SK(1,3), SK(1,4), L1,  1,NK, 1 )                   17DE9
      CALL CFV  ( SK(1,4), L1, 1, NK, RMP )                                   30JU9
      CALL ASFV ( SK(1,4), W(3,J), W(3,J), L1, 1, NK, +1.0 )                  17DE9
      CALL MBFV ( SK(1,2), WTM2(3,J), SK(1,3), L1,  1,NK, 1 )                 17DE9
      CALL CFV  ( SK(1,3), L1, 1, NK, DMP )                                   30JU9
      CALL ASFV ( SK(1,3), W(3,J), W(3,J), L1, 1, NK, +1.0 )                  30JU9
C                                                                               1SE8
C * * *   INERTIA LOAD VECTOR IS FORMED AND STORED IN W(1,J)                   01SE8
C * * *        WRITE QI TAPE (10) AND SHIFT EEM1 TO EFM2 AND                    01SE8
C * * *        EE TO EEM1, SHIFT DD TO DDM1 AND REPEAT FOR NEW J                01SE8
C                                                                               1SE8
               W(3,J)     = 0.0                                                01JL9
               W(MXP5,J) = 0.0                                                 01JL9
          IF ( J  .GT.  3 )                GO TO 5800                          30JU9
 5200     DO 5500 L = 3, MXP5                                                  30JU9
               W(L,J) = 0.0                                                    30JU9
 5500     CONTINUE                                                             30JU9
          GO TO 6000                                                           30JU9
 5800     IF ( J .EQ. MYP5 )                         GO TO 5200                22JL9
 6000     CONTINUE                                                             30JU9
      CALL IOBIN(5HWRITE ,10,W(3,J),MXP3)                                      30SE9IO
          DO 7000 I = 1, MXP3                                                  01SE8
               EEM2(I) = EEM1(I)                                               01SE8
               EEM1(I) = EE(I)                                                 01SE8
               DDM1(I,1) = DD(I,1)                                             01SE8
               DDM1(I,2) = DD(I,2)                                             01SE8
               DDM1(I,3) = DD(I,3)                                             01SE8
 7000     CONTINUE                                                             01SE8
 8000     CONTINUE                                                             01SE8
      CALL IOBIN (6HWRITER,10)                                                 22AG9IO
      RETURN                                                                    1SE8
      END                                                                       1SE8
C
```

SUBROUTINE EXCUT

This subroutine selects the appropriate subroutine to form the deflection coefficient matrix and the equivalent load vector.



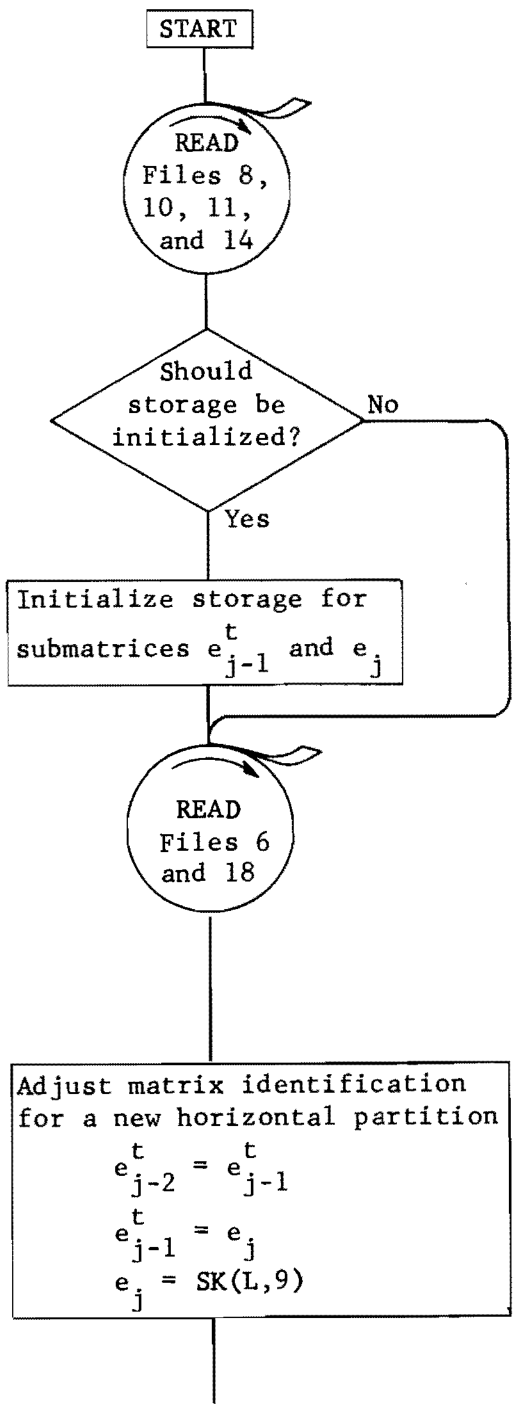
Check time step

Check calculation at first time step

```
      SUBROUTINE EXCUT ( L1, L2, L3, ET2, DT, CC, ET1, EE, FF,         01SE8
     1                   ML, JJ, N1, N2, N3, QI, QD1, QD2, QD3,        01SE8
     2                   QI1, QI2, QI3, SK, RHO, DF, SF        )       25MR9
      COMMON/INCR/ MX, MY, MXP2, MXP3, MXP4, MXP5, MXP7,               11NO8
     1             MYP2, MYP3, MYP4, MYP5, MYP7, MT                    11NO8
      COMMON/CON/  HXDHY3, HYDHX3, ODHXHY, ODHX, ODHY, PR, ODHT2, OD2HT,06JU9
     1             HXDHY, HYDHX                                        06JU9
      COMMON/RI/   NK, NL, NF, NT2SW, TIM                              29JA9
      TYPE INTEGER TIM                                                 11NO8
      DIMENSION  ET2( L1, N1),   DT( L1, N2),   CC( L1, N3),           11NO8
     1           ET1( L1, N1),   EE( N1, L1),   FF(   L1  ),           11NO8
     2           RHO(   L1  ),   DF(   L1  ),                          11NO8
     3            QI(   L1  ),                                         11NO8
     4           QD1(   L1  ),                                         11NO8
     5           QD2(   L1  ),  QD3(   L1  ),                          11NO8
     6           QI1(   L1  ),  QI2(   L1  ),                          11NO8
     7           QI3(   L1  ),                                         11NO8
     8            SK( L1,  9),  SF(  L1  )                             25MR9
          IF ( TIM - 2 ) 100, 50, 200                                  01SE8
   50     IF ( NT2SW  .EQ.  0 )          GO TO 300                     01SE8
  100 CALL STAT ( L1, L2, L3, ET2, DT, CC, ET1, EE, FF,                01SE8
     1           ML, JJ, N1, N2, N3, QI, QI1, SK, SF, QD1      )       27JE9
          GO TO 500                                                    01SE8
  200 CALL DYNAM ( L1, L2, L3, ET2, DT, CC, ET1, EE, FF,               01SE8
     1              ML, JJ, N1, N2, N3, QI, QD1, QD2, QD3,             01SE8
     2             QI1, QI2, QI3, SK, RHO, DF, SF              )       25MR9
          GO TO 500                                                    01SE8
  300 CALL ACCEL ( L1, L2, L3, ET2, DT, CC, ET1, EE, FF,               01SE8
     1              ML, JJ, N1, N2, N3, QD1, QI1, QI2,                 27JE9
     2             SK, RHO, DF, SF                             )       25MR9
  500     CONTINUE                                                     01SE8
      RETURN                                                           01SE8
      END                                                              01SE8
C
```

SUBROUTINE STAT

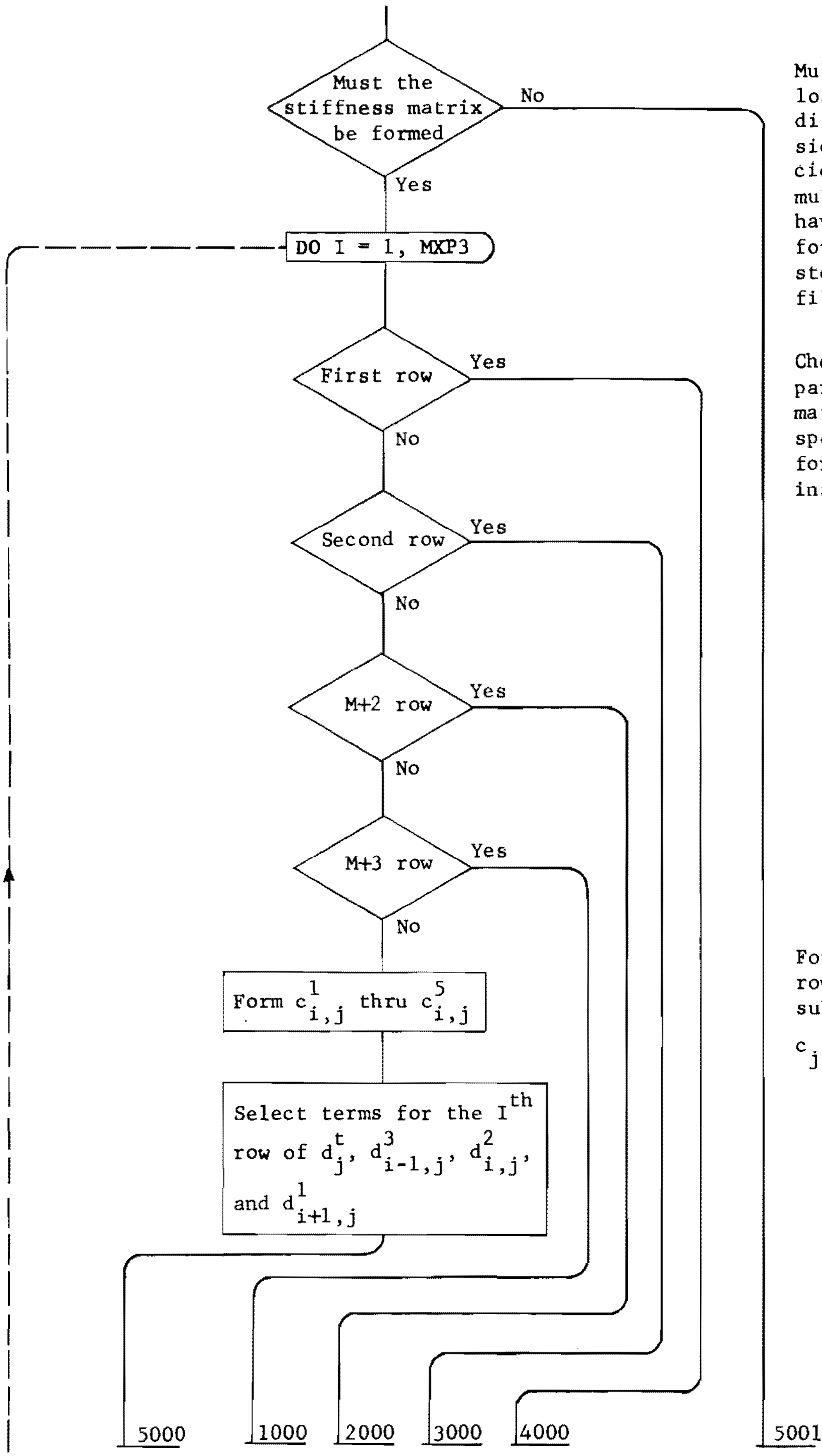
This subroutine forms the stiffness matrix and load vector for the static analysis and the deflection analysis at the end of the first time step.



Read static load, inertia and damping load, dynamic load, and correction load vectors

Recall a horizontal partition of the stiffness matrix and the corresponding linear approximation to the nonlinear foundation

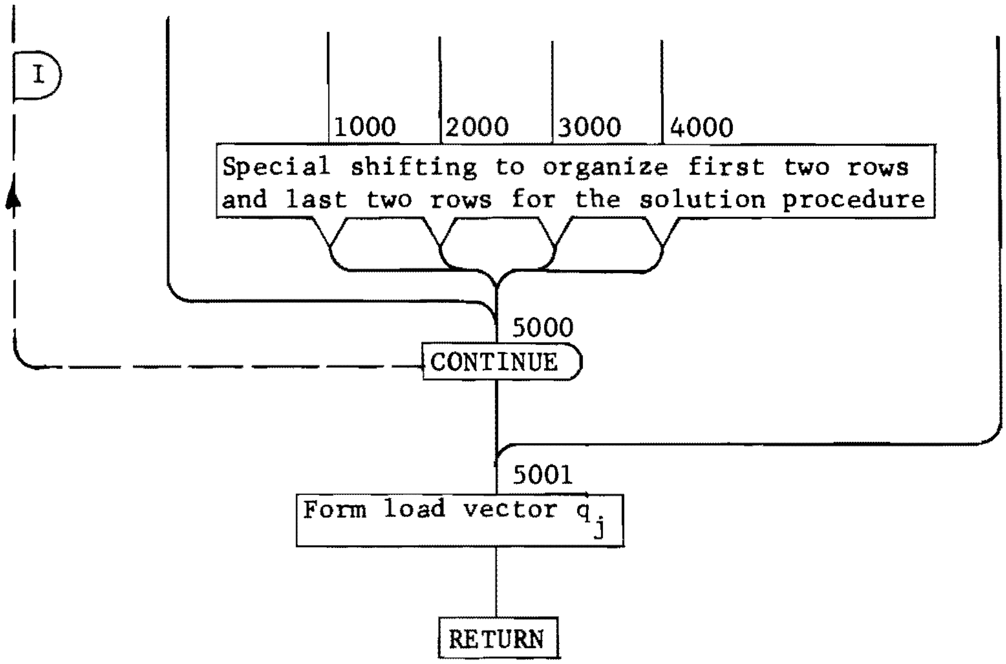
The submatrices e^t_{j-2} and e^t_{j-1} are replaced by what was e^t_{j-1} and e_j for the preceeding horizontal partition



Multiple loading condition, recursion coefficients, and multipliers have been formed and stored on files

Check row of partitioned matrix for special formation instructions

Form the I^{th} row of the submatrices c_j and d^t_j

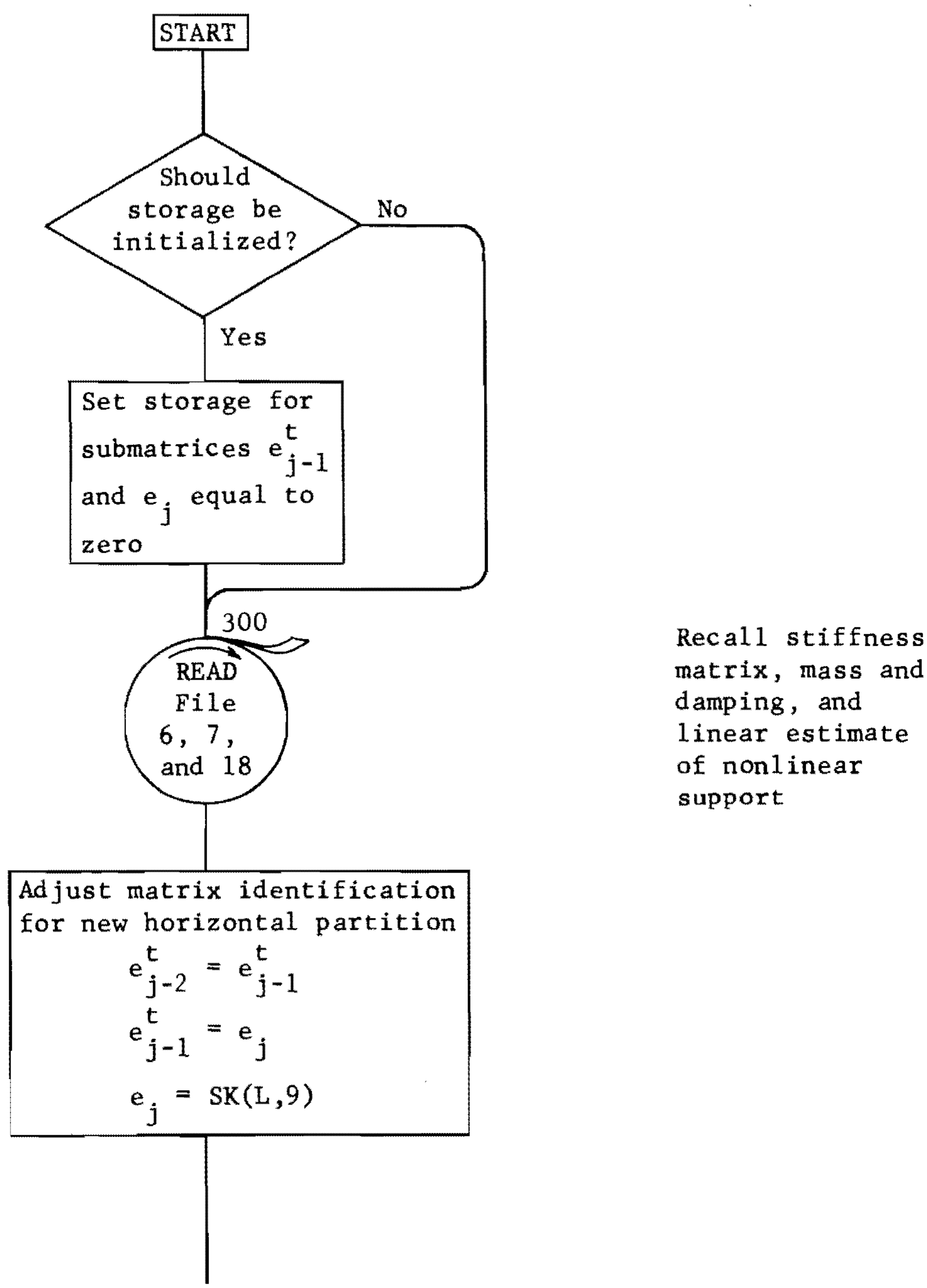

I
1000
2000
3000
4000
Special shifting to organize first two rows
and last two rows for the solution procedure
5000
CONTINUE
5001
Form load vector q_j
RETURN

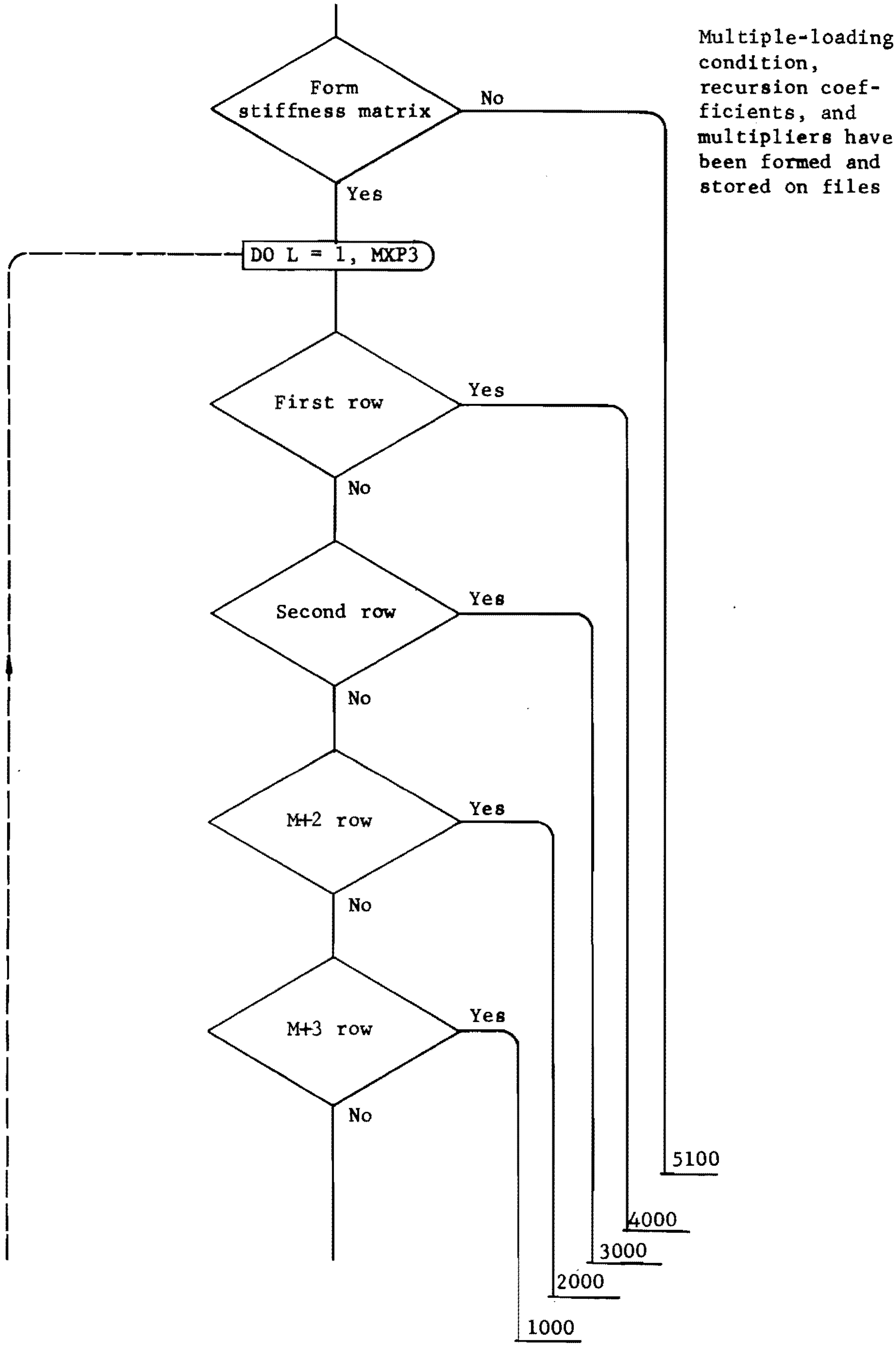
```
      SUBROUTINE STAT ( L1, L2, L3, ET2, DT, CC, ET1, EE, FF,      ML,   30JA9
     1                   JJ, N1, N2, N3, QI, QI1, SK, SF, QD1 )          27JE9
      COMMON/INCR/ MX, MY, MXP2, MXP3, MXP4, MXP5, MXP7,                 11NO8
     1             MYP2, MYP3, MYP4, MYP5, MYP7, MT                      11NO8
      COMMON/CON/  HXDHY3, HYDHX3, ODHXHY, ODHX, ODHY, PR, ODHT2, OD2HT,06JU9
     1             HXDHY, HYDHX                                          06JU9
      COMMON/RI/   NK, NL, NF, NT2SW, TIM                                29JA9
      TYPE INTEGER TIM                                                   11NO8
      DIMENSION  ET2( L1, N1),   DT( L1, N2),   CC( L1, N3),             11NO8
     1           ET1( L1, N1),   EE( N1, L1),   FF(    L1  ),            11NO8
     3             QI(  L1  ),     QI1(  L1  ),                          30JA9
     8             SK( L1,  9),          SF(  L1), QD1(L1)               27JE9
C                                                                         1SE8
C * * *   THIS SUBROUTINE FORMS THE ARRAYS OF MATRIX COEFFICIENTS        01SE8
C * * *          FOR THE R-1 PACKAGE FOR THE SOLUTION OF THE STATIC DEFL 01SE8
C                                                                         1SE8
      CALL IOBIN(4HREAD   ,8,FF,MXP3)                                    25AG910-
      CALL IOBIN(4HREAD   ,10,QI,MXP3)                                   25AG910-
      CALL IOBIN(4HREAD   ,14,QI1,MXP3)                                  30SE910
      CALL IOBIN(4HREAD   ,11,QD1,MXP3)                                  25AG910
         IF ( JJ  .GT.  NF )              GO TO 300                      11NO8
C * * *   INITIALIZE STORAGE                                             01SE8
C                                                                         1SE8
         DO 100 L = 1, MXP3                                              01SE8
               ET1(L,1) = 0.0                                            01SE8
               EE(1,L)  = 0.0                                            01SE8
  100    CONTINUE                                                        01SE8
C * * *   READ JJTH ROW OF STIFFNESS MATRIX SUBMATRICES                  01SE8
  300          MTK = 9 * MXP3                                            19AG910
      CALL IOBIN(4HREAD   ,6,SK,MTK)                                     25AG910
      CALL IOBIN(4HREAD   ,18,SF,MXP3)                                   25AG910
C                                                                         1SE8
C * * *   FORM ET2 AND ET1                                               01SE8
C                                                                         1SE8
         DO 700 L = 1, MXP3                                              01SE8
               ET2(L,1) = ET1(L,1)                                       01SE8
               ET1(L,1) = EE(1,L)                                        01SE8
               EE(1,L) = SK(L,9)                                         01SE8
  700    CONTINUE                                                        01SE8
         IF ( ML   .EQ.  - 1 )            GO TO 5001                     30JA9
C                                                                         1SE8
C * * *   FORM DT, CC, AND EE                                            01SE8
C                                                                         1SE8
         DO 5000 I = 1, MXP3                                             01SE8
               SK(I,3) = SK(I,3) + SF(I)                                 25MR9
         IF ( I   .EQ.   1 )              GO TO 1000                     22MR9
         IF ( I   .EQ.   2 )              GO TO 2000                     22MR9
         IF ( I   .EQ.   MXP2 )           GO TO 3000                     22MR9
         IF ( I   .EQ.   MXP3 )           GO TO 4000                     22MR9
C                                                                         1SE8
C * * *   FORM SUBMATRICES AT A GENERAL INTERIOR STATION                 01SE8
C                                                                         1SE8
               CC(I,1) = SK(I,1)                                         01SE8
               CC(I,2) = SK(I,2)                                         01SE8
               CC(I,3) = SK(I,3)                                         01SE8
         IF ( CC(I,3)  .LT.  1.E-20 )  CC(I,3 ) = 1.0                    19MR9
               CC(I,4) = SK(I,4)                                         01SE8
               CC(I,5) = SK(I,5)                                         01SE8
C                                                                         1SE8
               DT(I,1) = SK(I-1,8)                                       01SE8
               DT(I,2) = SK(I,7)                                         01SE8
               DT(I,3) = SK(I+1,6)                                       01SE8
         GO TO 5000                                                      01SE8
C                                                                         1SE8
C * * *   SHIFT MATRIX COEFF FOR CC AND DT TERMS TO LT EDGE - I = 1      01SE8
C                                                                         1SE8
 1000          CC(I,1) = SK(I,3)                                         01SE8
         IF ( CC(I,1)  .LT.  1.E-20 )  CC(I,1) = 1.0                     19MR9
               CC(I,2) = SK(I,4)                                         01SE8
               CC(I,3) = SK(I,5)                                         01SE8
               CC(I,4) = 0.0                                             01SE8
               CC(I,5) = 0.0                                             01SE8
C                                                                         1SE8
               DT(I,1) = SK(I,7)                                         01SE8
               DT(I,2) = SK(I+1,6)                                       01SE8
               DT(I,3) = 0.0                                             01SE8
         GO TO 5000                                                      01SE8
C                                                                         1SE8
C * * *   SHIFT MATRIX COEFF FOR CC TO LT EDGE - I = 2                   01SE8
C                                                                         1SE8
 2000          CC(I,1) = SK(I,2)                                         01SE8
               CC(I,2) = SK(I,3)                                         01SE8
         IF ( CC(I,2)  .LT.  1.E-20 )  CC(I,2) = 1.0                     19MR9
               CC(I,3) = SK(I,4)                                         01SE8
               CC(I,4) = SK(I,5)                                         01SE8
               CC(I,5) =  0.0                                            01SE8
C                                                                         1SE8
               DT(I,1) = SK(I-1,8)                                       01SE8
               DT(I,2) = SK(I,7)                                         01SE8
               DT(I,3) = SK(I+1,6)                                       01SE8
         GO TO 5000                                                      01SE8
C                                                                         1SE8
C * * *   SHIFT MATRIX COEFF FOR CC TO RT EDGE - I = MXP2                01SE8
C                                                                         1SE8
 3000          CC(I,1) = 0.0                                             01SE8
               CC(I,2) = SK(I,1)                                         01SE8
               CC(I,3) = SK(I,2)                                         01SE8
               CC(I,4) = SK(I,3)                                         01SE8
         IF ( CC(I,4)  .LT.  1.E-20 )  CC(I,4) = 1.0                     19MR9
               CC(I,5) = SK(I,4)                                         01SE8
C                                                                         1SE8
               DT(I,1) = SK(I-1,8)                                       01SE8
               DT(I,2) = SK(I,7)                                         01SE8
               DT(I,3) = SK(I+1,6)                                       01SE8
         GO TO 5000                                                      01SE8
C                                                                         1SE8
C * * *   SHIFT MATRIX COEFF FOR CC AND DT TO RT EEGE - I = MxP3         01SE8
C                                                                         1SE8
 4000          CC(I,1) = 0.0                                             01SE8
               CC(I,2) = 0.0                                             01SE8
               CC(I,3) = SK(I,1)                                         01SE8
               CC(I,4) = SK(I,2)                                         01SE8
```

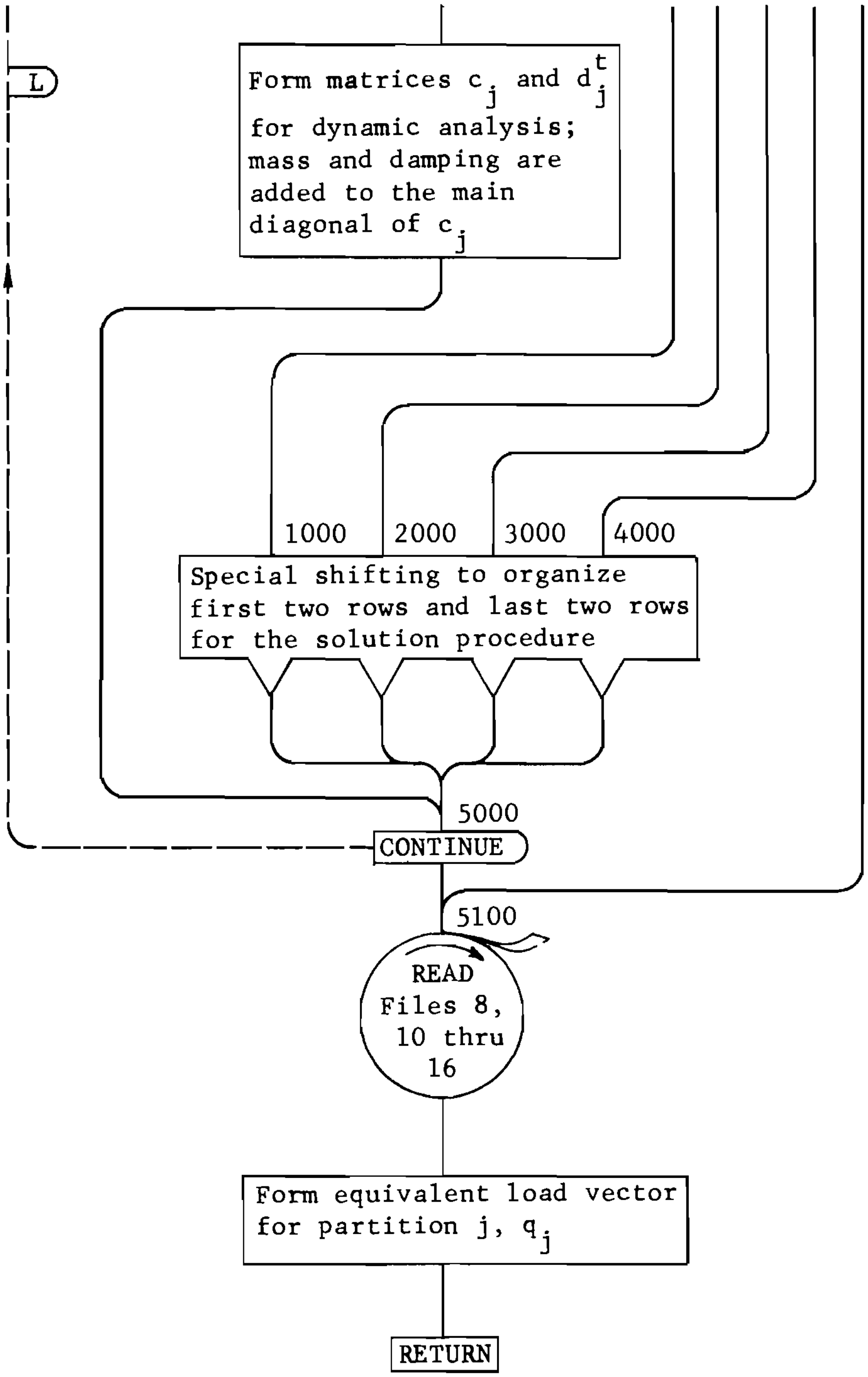
```
            CC(I,5) = SK(I,3)                                            01SE8
      IF ( CC(I,5)  .LT.  1.E-20 )  CC(I,5) = 1.0                        19MR9
C                                                                         1SE8
            DT(I,1) = 0.0                                                01SE8
            DT(I,2) = SK(I-1,8)                                          01SE8
            DT(I,3) = SK(I,7)                                            01SE8
 5000    CONTINUE                                                        01SE8
 5001    CONTINUE                                                        16OC9
         DO 6000  I = 1, MXP3                                            11NO8
            FF(I) = FF(I) + QI(I) + QI1(I) + QD1(I)                      27JE9
 6000    CONTINUE                                                        01SE8
      RETURN                                                              1SE8
      END                                                                 1SE8
C
```

SUBROUTINE DYNAM

This subroutine generates the equivalent load vector and matrix coefficients for the deflection analysis for the general time step.




Form stiffness matrix
No
Yes
Multiple-loading condition, recursion coefficients, and multipliers have been formed and stored on files
DO L = 1, MXP3
First row
Yes
No
Second row
Yes
No
M+2 row
Yes
No
M+3 row
Yes
No
5100
4000
3000
2000
1000



Form L^{th} row of submatrices c_j and d_j^t

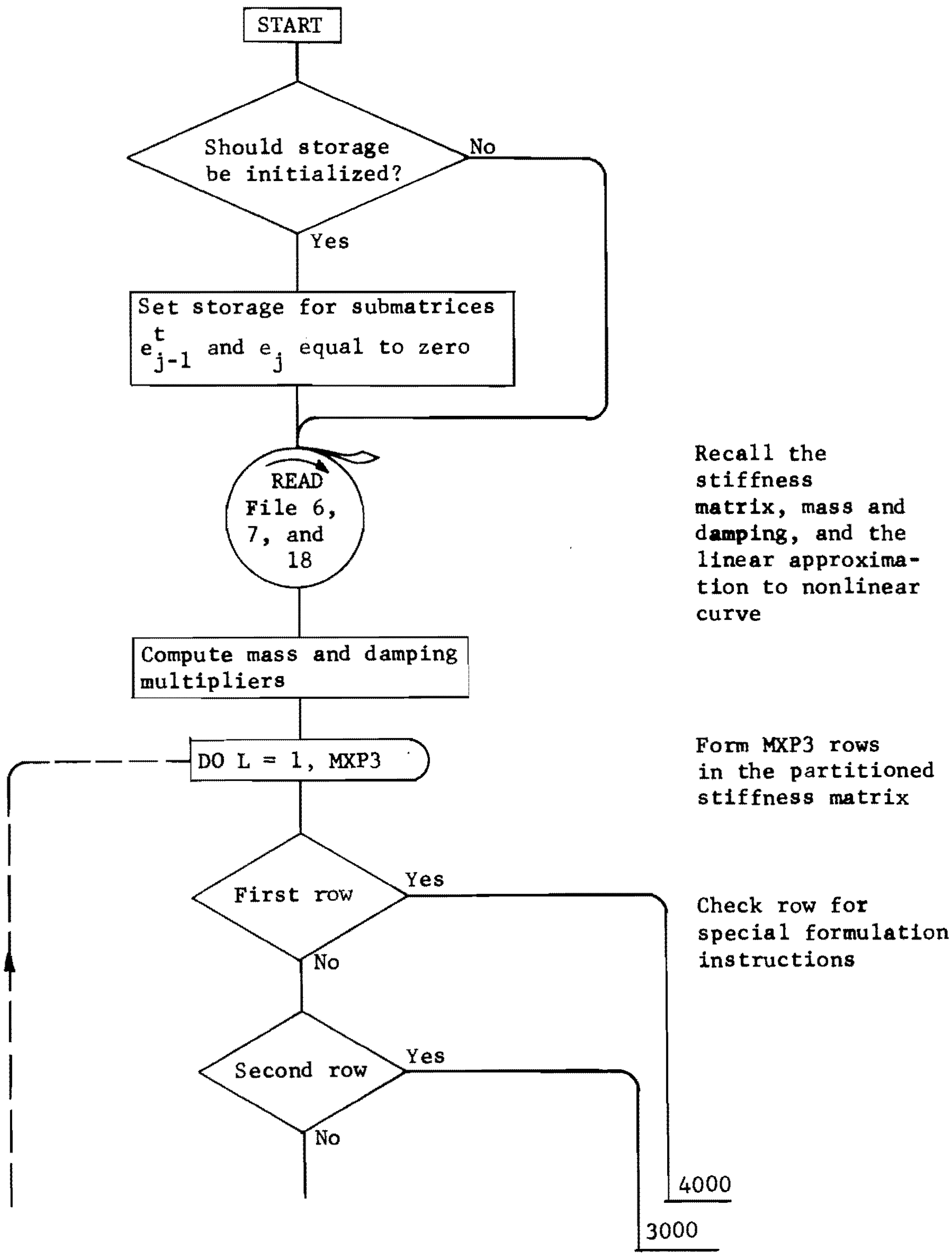
Read all load vectors for partition j

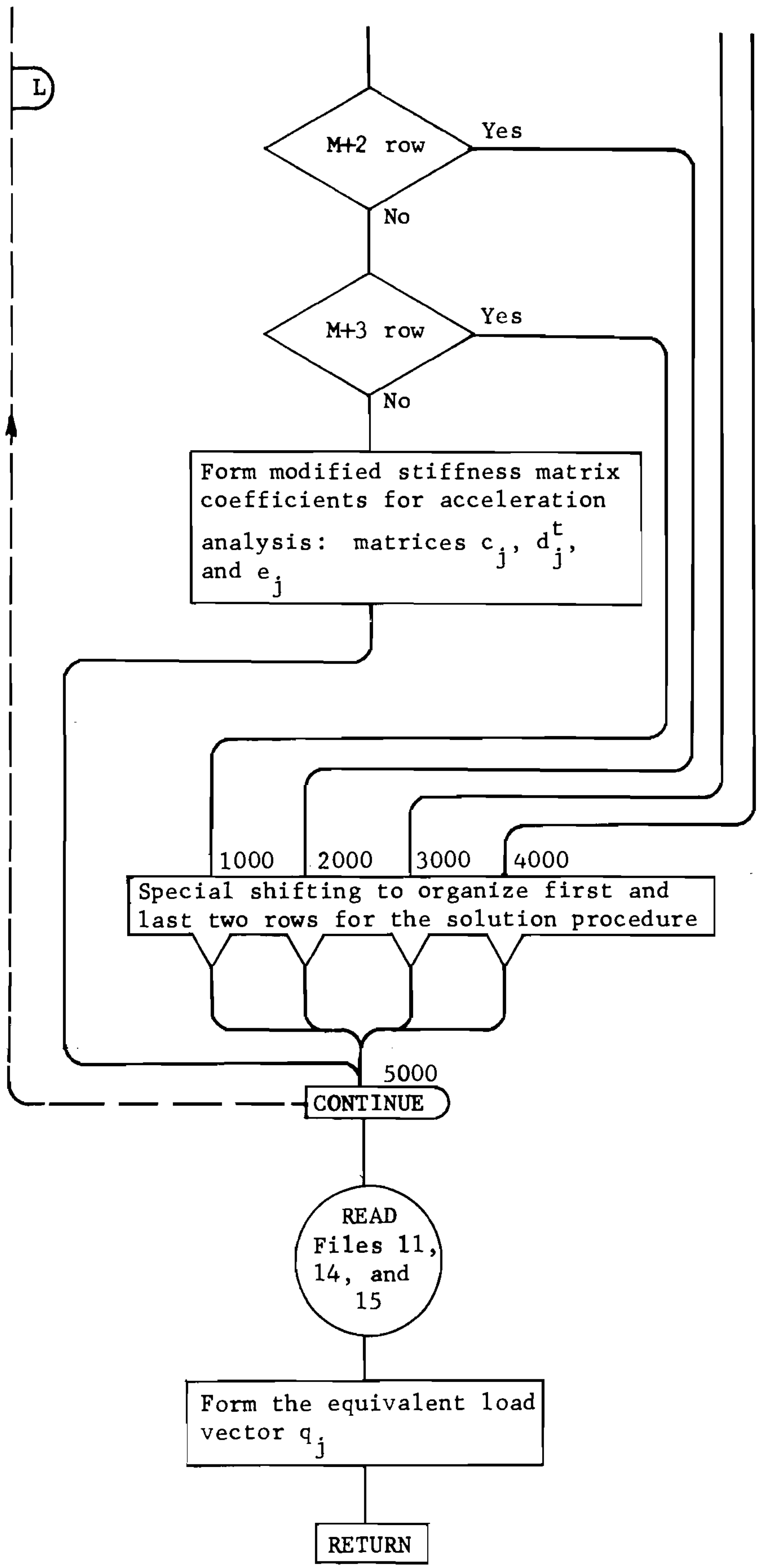
```
      SUBROUTINE DYNAM ( L1, L2, L3, ET2, DT, CC, ET1, EE, FF,          01SE8
     1                   ML, JJ, N1, N2, N3, QI, QD1, QD2, QD3,         01SE8
     2                   QI1, QI2, QI3, SK, RHO, DF, SF        )        25MR9
      COMMON/INCR/ MX, MY, MXP2, MXP3, MXP4, MXP5, MXP7,                11NO8
     1             MYP2, MYP3, MYP4, MYP5, MYP7, MT                     11NO8
      COMMON/CON/  HXDHY3, HYDHX3, ODHXHY, ODHX, ODHY, PR, ODHT2, OD2HT,06JU9
     1             HXDHY, HYDHX                                         06JU9
      COMMON/RI/    NK, NL, NF, NT2SW, TIM                              22JA9
      DIMENSION  ET2( L1, N1 ),        DT( L1, N2 ),       CC( L1, N3 ), 01SE8
     1               ET1( L1 , N1 ),        EE( N1 , L1 ),       FF( L1 ),30JU9
     2             QI( L1 ),            QD1( L1 ),          QD2( L1 ),  01SE8
     3            QD3( L1 ),            QI1( L1 ),          QI2( L1 ),  01SE8
     4            QI3( L1 ),             SK( L1, 9  ),      RHO( L1 ),  01SE8
     5             DF( L1 ),             SF(  L1  )                     25MR9
C                                                                        1SE8
C * * * * THIS SUBROUTINE FORMS THE STIFFNESS MATRIX AND LOAD VECTOR    01SE8
C * * * *       FOR THE 2ND TO LAST TIME STEP                           01SE8
C                                                                        1SE8
         IF ( JJ  .GT.  NF )              GO TO 300                     01SE8
         DO 100 L = 1, MXP3                                             01SE8
              ET1(L,1) = 0.0                                            01SE8
              EE(1,L)  = 0.0                                            01SE8
  100    CONTINUE                                                       01SE8
C                                                                        1SE8
C * * *  BEGIN FORMULATION OF SUBMATRICES FOR FRIP4 SOLUTION            01SE8
C * * *  READ JJ TH ROW OF STIFFNESS MATRIX, MASS AND DAMPING           01SE8
C                                                                        1SE8
  300         MKT = 9 * MXP3                                            19AG9IO
      CALL IOBIN(4HREAD   ,6,SK,MKT)                                    25AG9IO-
      CALL IOBIN(4HREAD   ,7,RHO,MXP3)                                  25AG9IO-
      CALL IOBIN(4HREAD   ,7, DF,MXP3)                                  25AG9IO-
      CALL IOBIN(4HREAD   ,18,SF,MXP3)                                  25AG9IO-
         DO 400 L = 1, MXP3                                             01SE8
              ET2(L,1) = ET1(L,1)                                       01SE8
              ET1(L,1) = EE(1,L)                                        01SE8
              EE(1,L)  = SK(L,9)                                        01SE8
  400    CONTINUE                                                       01SE8
         IF ( ML  .EQ.  -1 )              GO TO 5100                    01SE8
C                                                                        1SE8
C * * *  WHEN ML = 1, FORM ENTIRE STIFFNESS MATRIX                      01SE8
C                                                                        1SE8
         DO 5000 L = 1, MXP3                                            01SE8
              SK(L,3) = SK(L,3) + SF(L)                                 25MR9
         IF ( L  .EQ.  1 )                GO TO 1000                    01SE8
         IF ( L  .EQ.  2 )                GO TO 2000                    01SE8
         IF ( L  .EQ.  MXP2 )             GO TO 3000                    01SE8
         IF ( L  .EQ.  MXP3 )             GO TO 4000                    01SE8
C                                                                        1SE8
C * * *  FORM CC AND DT AT GENERAL INTERIOR STATION                     01SE8
C                                                                        1SE8
              CC(L,1) = SK(L,1)                                         01SE8
              CC(L,2) = SK(L,2)                                         01SE8
              CC(L,3) = SK(L,3) + 6.0 * ( DF(L) * OD2HT + RHO(L) *      01SE8
     1                                ODHT2 )                           01SE8
         IF ( CC(L,3)  .LT.  1.E-20 )  CC(L,3 ) = 1.0                   19MR9
              CC(L,4) = SK(L,4)                                         01SE8
              CC(L,5) = SK(L,5)                                         01SE8
C                                                                        1SE8
  700         DT(L,1) = SK(L-1,8)                                       01SE8
              DT(L,2) = SK(L,7)                                         01SE8
              DT(L,3) = SK(L+1,6)                                       01SE8
         GO TO 5000                                                     01SE8
C                                                                        1SE8
C * * *  SHIFT MATRIX COEFF FOR CC AND DT TERMS TO LT EDGE, L = 1       01SE8
C                                                                        1SE8
 1000         CC(L,1) = SK(L,3)                                         01SE8
         IF ( CC(L,1)  .LT.  1.E-20 )  CC(L,1) = 1.0                    19MR9
              CC(L,2) = SK(L,4)                                         01SE8
              CC(L,3) = SK(L,5)                                         01SE8
              CC(L,4) = 0.0                                             01SE8
              CC(L,5) = 0.0                                             01SE8
C                                                                        1SE8
              DT(L,1) = SK(L,7)                                         01SE8
              DT(L,2) = SK(L+1,6)                                       01SE8
              DT(L,3) = 0.0                                             01SE8
         GO TO 5000                                                     01SE8
C                                                                        1SE8
C * * *  SHIFT MATRIX COEFF FOR CC TO LT EDGE, L = 2                    01SE8
C                                                                        1SE8
 2000         CC(L,1) = SK(L,2)                                         01SE8
              CC(L,2) = SK(L,3) + 6.0 * ( DF(L) * OD2HT + RHO(L) *      01SE8
     1                                ODHT2 )                           01SE8
         IF ( CC(L,2)  .LT.  1.E-20 )  CC(L,2) = 1.0                    19MR9
              CC(L,3) = SK(L,4)                                         01SE8
              CC(L,4) = SK(L,5)                                         01SE8
              CC(L,5) = 0.0                                             01SE8
         GO TO 700                                                      01SE8
C                                                                        1SE8
C * * *  SHIFT MATRIX COEFF FOR CC TO RT EDGE, L = MXP2                 01SE8
C                                                                        1SE8
 3000         CC(L,1) = 0.0                                             01SE8
              CC(L,2) = SK(L,1)                                         01SE8
              CC(L,3) = SK(L,2)                                         01SE8
              CC(L,4) = SK(L,3) + 6.0 * ( DF(L) * OD2HT + RHO(L) *      01SE8
     1                                ODHT2 )                           01SE8
         IF ( CC(L,4)  .LT.  1.E-20 )  CC(L,4) = 1.0                    19MR9
              CC(L,5) = SK(L,4)                                         09MR9
         GO TO 700                                                      01SE8
C                                                                        1SE8
C * * *  SHIFT MATRIX COEFF FOR CC AND DT TO RT EDGE, L = MXP3          01SE8
C                                                                        1SE8
 4000         CC(L,1) = 0.0                                             01SE8
              CC(L,2) = 0.0                                             01SE8
              CC(L,3) = SK(L,1)                                         01SE8
              CC(L,4) = SK(L,2)                                         01SE8
              CC(L,5) = SK(L,3)                                         30JA9
         IF ( CC(L,5)  .LT.  1.E-20 )  CC(L,5) = 1.0                    19MR9
C                                                                        1SE8
              DT(L,1) = 0.0                                             01SE8
              DT(L,2) = SK(L-1,8)                                       01SE8
              DT(L,3) = SK(L,7)                                         01SE8
 5000    CONTINUE                                                       01SE8
```

```
 5100     CONTINUE                                                            01SE8
C                                                                              1SE8
C * * *   FORM LOAD VECTOR - FF                                               01SE8
C                                                                              1SE8
      CALL IOBIN(4HREAD   ,10, QI,MXP3)                                       25AG910-
      CALL IOBIN(4HREAD   ,11,QD1,MXP3)                                       25AG910-
      CALL IOBIN(4HREAD   ,12,QD2,MXP3)                                       25AG910-
      CALL IOBIN(4HREAD   ,13,QD3,MXP3)                                       25AG910-
      CALL IOBIN(4HREAD   ,14,QI1,MXP3)                                       25AG910-
      CALL IOBIN(4HREAD   ,15,QI2,MXP3)                                       25AG910-
      CALL IOBIN(4HREAD   ,16,QI3,MXP3)                                       25AG910-
      CALL IOBIN(4HREAD   ,8,FF,MXP3)                                         25AG910-
 5999     CONTINUE                                                            17DE9
          DO 6000 L = 1, MXP3                                                 01SE8
              FF(L) = 6.0 * FF(L) + QI(L) + QD1(L) + QI1(L) +                 01SE8
     1                4.0 * ( QD2(L) + QI2(L) ) + QD3(L) + QI3(L)             01SE8
 6000     CONTINUE                                                            01SE8
      RETURN                                                                  01SE8
      END                                                                      1SE8
C
```

SUBROUTINE ACCEL

This subroutine organizes the matrix coefficients and right-hand side for acceleration analysis at the first time step.





Recall dynamic loading for first time step and correction loads for static condition and the first time step

```
      SUBROUTINE ACCEL ( L1, L2, L3, ET2, DT, CC, ET1, EE, FF,          01SE8
     1                 ML, JJ, N1, N2, N3, QD1, Q11, Q12,               27JE9
     2                      SK, RHO, DF, SF                    )        25MR9
      COMMON/INCR/ MX, MY, MXP2, MXP3, MXP4, MXP5, MXP7,                11NO8
     1               MYP2, MYP3, MYP4, MYP5, MYP7, MT                   11NO8
      COMMON/CON/  HXDHY3, HYDHX3, ODHXHY, ODHX, ODHY, PR, ODHT2, OD2HT,06JU9
     1               HXDHY, HYDHX                                       06JU9
      COMMON/RI/   NK, NL, NF, NT2SW, TIM                               29JA9
      TYPE INTEGER TIM                                                  11NO8
      TYPE REAL KMULT                                                   11NO8
      DIMENSION  ET2( L1, N1 ),       DT( L1, N2 ),       CC( L1, N3 ), 01SE8
     1             ET1( L1, N1 ),       EE( N1, L1 ),       FF( L1 ),   01SE8
     2             QD1( L1 ),           Q11( L1 ),           Q12( L1 ), 27JE9
     3              SK( L1, 9 ),       RHO(    L1  ),       DF(    L1  ),18NO8
     4                           SF(   L1   )                           27JE9
C                                                                        1SE8
C * * * * THIS SUBROUTINE FORMS THE DYNAMIC STIFFNESS MATRIX FOR THE    01SE8
C * * * *         SOLUTION OF THE INITIAL ACCELERATION                  01SE8
                  JJJ = JJ - 2                                          01SE8
          IF ( JJ  .GT.  NF )              GO TO 300                    01SE8
          DO 100 L = 1,MXP3                                             01SE8
                  ET1(L,1) = 0.0                                        01SE8
                  EE(1,L)  = 0.0                                        01SE8
  100     CONTINUE                                                      01SE8
C                                                                        1SE8
C * * *   BEGIN FORMULATION OF SUBMATRICES FOR FRIP4 SOLUTION           01SE8
C * * *   READ JJ TH ROW OF STIFFNESS MATRIX, MASS AND DAMPING          01SE8
C                                                                        1SE8
  300             MTK = 9 * MXP3                                        19AG910
      CALL IOBIN(4HREAD    ,6,SK,MTK)                                   25AG910
      CALL IOBIN(4HREAD    ,7,RHO,MXP3)                                 25AG910
      CALL IOBIN(4HREAD    ,7, DF,MXP3)                                 25AG910
      CALL IOBIN(4HREAD    ,18,SF,MXP3)                                 25AG910
          DO 400 L = 1, MXP3                                            01SE8
                  ET2(L,1) = ET1(L,1)                                   01SE8
                  ET1(L,1) = EE(1,L)                                    01SE8
  400     CONTINUE                                                      01SE8
C                                                                        1SE8
C * * *   FORM CC, DT, AND EE                                           01SE8
C                                                                        1SE8
                  KMULT = 6.0 * ODHT2                                   16JU9
                  KMULT = 1.0 / KMULT                                   16JU9
                  OD2HT = 4.0 * DD2HT                                   16JU9
                  OD2HT = 1.0 / OD2HT                                   16JU9
          DO 5000 L = 1, MXP3                                           01SE8
                  SK(L,3) = SK(L,3) + SF(L)                             25MR9
          IF ( L  .EQ.  1 )                GO TO 1000                   01SE8
          IF ( L  .EQ.  2 )                GO TO 2000                   01SE8
          IF ( L  .EQ.  MXP2 )             GO TO 3000                   01SE8
          IF ( L  .EQ.  MXP3 )             GO TO 4000                   01SE8
C                                                                        1SE8
C * * *   FORM SUBMATRICES AT A GENERAL INTERIOR STATION                01SE8
C                                                                        1SE8
                  CC(L,1) = SK(L,1) * KMULT                             01SE8
                  CC(L,2) = SK(L,2) * KMULT                             01SE8
                  CC(L,3) = SK(L,3) * KMULT + DF(L) * OD2HT + RHO(L)    01SE8
          IF ( CC(L,3)  .LT.  1.E-20 )  CC(L,3 ) = 1.0                  19MR9
                  CC(L,4) = SK(L,4) * KMULT                             01SE8
                  CC(L,5) = SK(L,5) * KMULT                             01SE8
C                                                                        1SE8
  700             DT(L,1) = SK(L-1,8) * KMULT                           11NO8
                  DT(L,2) = SK(L,7)    * KMULT                          11NO8
                  DT(L,3) = SK(L+1,6) * KMULT                           11NO8
C                                                                       11NO8
                  EE(1,L) = SK(L,9)    * KMULT                          11NO8
          GO TO 5000                                                    01SE8
C                                                                        1SE8
C * * *   SHIFT MATRIX COEFF FOR CC AND DT TERMS TO LT EDGE, L = 1      01SE8
C                                                                        1SE8
 1000             CC(L,1) = SK(L,3) * KMULT                             01SE8
          IF ( CC(L,1)  .LT.  1.E-20 )  CC(L,1) = 1.0                   19MR9
                  CC(L,2) = SK(L,4) * KMULT                             01SE8
                  CC(L,3) = SK(L,5) * KMULT                             01SE8
                  CC(L,4) = 0.0                                         01SE8
                  CC(L,5) = 0.0                                         01SE8
C                                                                        1SE8
                  DT(L,1) = SK(L,7)    * KMULT                          01SE8
                  DT(L,2) = SK(L+1,6) * KMULT                           01SE8
                  DT(L,3) = 0.0                                         01SE8
C                                                                       11NO8
                  EE(1,L) = SK(L,9)    * KMULT                          11NO8
          GO TO 5000                                                    01SE8
C                                                                        1SE8
C * * *   SHIFT MATRIX COEFF FOR CC TO LT EDGE, L = 2                   01SE8
C                                                                        1SE8
 2000             CC(L,1) = SK(L,2) * KMULT                             01SE8
                  CC(L,2) = SK(L,3) * KMULT + DF(L) * OD2HT + RHO(L)    01SE8
          IF ( CC(L,2)  .LT.  1.E-20 )  CC(L,2) = 1.0                   19MR9
                  CC(L,3) = SK(L,4) * KMULT                             01SE8
                  CC(L,4) = SK(L,5) * KMULT                             01SE8
                  CC(L,5) = 0.0                                         01SE8
          GO TO 700                                                     01SE8
C                                                                        1SE8
C * * *   SHIFT MATRIX COEFF FOR CC TO RT EDGE, L = MXP2                01SE8
C                                                                        1SE8
 3000             CC(L,1) = 0.0                                         01SE8
                  CC(L,2) = SK(L,1) * KMULT                             01SE8
                  CC(L,3) = SK(L,2) * KMULT                             01SE8
                  CC(L,4) = SK(L,3) * KMULT + DF(L) * OD2HT + RHO(L)    01SE8
          IF ( CC(L,4)  .LT.  1.E-20 )  CC(L,4) = 1.0                   19MR9
                  CC(L,5) = SK(L,4) * KMULT                             01SE8
          GO TO 700                                                     01SE8
C                                                                        1SE8
C * * *   SHIFT MATRIX COEFF FOR CC AND DT TO RT EDGE, L = MXP3         01SE8
C                                                                        1SE8
 4000             CC(L,1) = 0.0                                         01SE8
                  CC(L,2) = 0.0                                         01SE8
                  CC(L,3) = SK(L,1) * KMULT                             01SE8
                  CC(L,4) = SK(L,2) * KMULT                             01SE8
                  CC(L,5) = SK(L,3) * KMULT                             01SE8
          IF ( CC(L,5)  .LT.  1.E-20 )  CC(L,5) = 1.0                   19MR9
C                                                                        1SE8
```

```
               DT(L,1) = 0.0                                                    01SE8
               DT(L,2) = SK(L-1,8) * KMULT                                      25JU9
               DT(L,3) = SK(L,7)   * KMULT                                      25JU9
               EE(1,L) = 0.0                                                    18NO8
 5000     CONTINUE                                                              01SE8
C                                                                                1SE8
C * * *   FORM THE LOAD VECTOR - FF                                             01SE8
C                                                                                1SE8
 5200     CONTINUE                                                              17DE9
      CALL IOBIN(4HREAD    ,11,QD1,MXP3)                                        25AG910
      CALL IOBIN(4HREAD    ,14,QI1,MXP3)                                        25AG910
      CALL IOBIN(4HREAD    ,15,QI2,MXP3)                                        25AG910-
          DO 6000 L = 1, MXP3                                                   01SE8
               FF(L) = QD1(L) + QI1(L) - QI2(L)                                 27JE9
 6000     CONTINUE                                                              01SE8
               OD2HT = 4.0 * OD2HT                                              16JU9
               OD2HT = 1.0 / OD2HT                                              16JU9
      RETURN                                                                     1SE8
      END                                                                       01SE8
C
```

SUBROUTINE FRIP4

This subroutine is an IOBIN version of the FRIP4 Five-Wide Recursion-Inversion Solution Process which is documented in Research Report 56-19, "An Alegebraic Equation Solution Process Formulated in Anticipation of Banded Linear Equations," by Frank L. Endres and Hudson Matlock (Ref 6). IOBIN is a CDC system routine which is used for efficient file manipulations.

These routines are called by FRIP4 and some by subroutines MASSAC and INERTIA and are completely documented in Ref 6:

INVR5		-	Takes inverse of general positive definite matrix
INVR6	DCOM1 INVLT1 MLTXL	-	Takes inverse of symmetric positive definite matrix
MFFV		-	Multiplies full (square) matrix times a full (square) matrix or a vector
SMFF		-	Symmetric multiplication of a full times a full matrix
MFFT		-	Multiplies a full times the transpose of a full matrix
MBFV		-	Multiplies a banded (packed) matrix times a full matrix or a vector
MFB		-	Multiplies a full matrix times a banded (packed) matrix
ABF		-	Adds a banded matrix to a full matrix
ASFV		-	Adds or subtracts two full matrices or two vectors
RFV		-	Replaces a full matrix or a vector by another
CFV		-	Multiples a full matrix or a vector by a constant

```
      SUBROUTINE FRIP 4 ( L1, L2, L3, ML, A, AM1, AM2, ATM, B, BM1, EP1,11NO8
     1                C, CM1, D, E, ET2, DT, CC, ET1, EE, FF, W, N1, N2,  11NO8
     2                N3, QI, QD1, QD2, QD3, QI1, QI2, QI3, SK, RHO,DF,SF)25MR9
C * * * *       FRIP 4A - REVISION DATE 16JU9          ( SLAB 33 )      16JU9
C******* THIS GROUP OF 15 SUBROUTINES PROVIDES THE USER WITH AN          20MY8
C        EFFICIENT GENERAL SPARSELY BANDED EQUATION SOLVER              04JA8
C        (THE MATRIX IS ASSUMED TO BE SYMMETRIC AND POSITIVE DEFINITE)  12MR8
C        WHICH CAN HANDLE UP TO 5 GROUPS OF BANDS , EACH                04JA8
C        OF ARBITRARY WIDTH                                             04JA8
C******* THIS ROUTINE SUPERVISES 14 SUBROUTINES , 13 OF WHICH            20MY8
C        ARE SELF-SUFFICIENT AND COME AS A PACKAGE , THE                04JA8
C        REMAINING ONE GENERATES AND PACKS THE STIFFNESS                04JA8
C******* MATRIX AS OUTLINED IN IN THE APPENDIX OF THE RELATED REPORT    23MR8
C        THIS ROUTINE MUST BE SUPPLIED BY THE USER SINCE                04JA8
C        IT DESCRIBES HIS PARTICULAR PROBLEM                            04JA8
C******* IN THE MAIN PROGRAM THE FOLLOWING PAIR  SHOULD BE EQUIVALENCED 20MY8
C        ( ATM , DT )                                                   20MY8
C******* SCRATCH TAPES SHOULD BE REQUESTED FOR TAPES 1 AND 2            05MR8
      DIMENSION  A(L1    )  , AM1(L1    )  , AM2(L1    )  ,             20MY8
     1            B(L1,L1)  , BM1(L1,L1)  , EP1(L1,L1)  , ATM(L1    )  , 20MY8
     2            C(L1,L1)  , CM1(L1,L1)  ,    D(L1,L1)  ,              20MY8
     3            E(L1,L1)  , W(L2,L3)    , ET2(L1,N1)  ,               20MY8
     4           DT(L1,N2)  ,   CC(L1,N3)  , ET1(L1,N1)  ,   EE(N1,L1)  , 23MR8
     5                          FF(L1)                                  23MR8
      DIMENSION   QI( L1),       QD1( L1),       QD2( L1),      QD3( L1), 11NO8
     1            QI1( L1),      QI2( L1),       QI3( L1),      RHO( L1), 11NO8
     2             DF( L1),       SK(L1,  9),  SF( L1 )                  25MR9
      COMMON/INCR/ MX, MY, MXP2, MXP3, MXP4, MXP5, MXP7,                 11NO8
     1             MYP2, MYP3, MYP4, MYP5, MYP7, MT                     11NO8
      COMMON/CON/  HXDHY3, HYDHX3, ODHXHY, ODHX, ODHY, PR, ODHT2, OD2HT,06JU9
     1             HXDHY, HYDHX                                         06JU9
      COMMON/RI/   NK, NL, NF, NT2SW, TIM                               29JA9
      TYPE INTEGER TIM                                                  11NO8
      CALL IOBIN(6HREWIND,1)                                            19AG9IO
      CALL IOBIN(6HREWIND,2)                                            19AG9IO
      CALL IOBIN(6HREWIND,3)                                            19AG9IO
                K2 = NK * NK                                            19AG9IO
         IF( ML ) 140, 100, 100                                         04JA8
C        SET INITIAL CONDITIONS                                         04JA8
  100    DO 135  J = 1 , NK                                             01FE8
         DO 130  I = 1 , NK                                             01FE8
              B(I,J)  = 0.0                                             04JA8
              C(I,J)  = 0.0                                             04JA8
              CM1(I,J) = 0.0                                            04JA8
              EP1(I,J) = 0.0                                            23MR8
              D(I,J) = 0.0                                              16JU9
  130    CONTINUE                                                       04JA8
  135    CONTINUE                                                       04JA8
  140    DO 150  I = 1 , NK                                             01FE8
              A(I) = 0.0                                                20MY8
              AM1(I) = 0.0                                              20MY8
  150    CONTINUE                                                       04JA8
C*************************************************************************
C        BEGIN FORWARD PASS  --  SOLVE FOR RECURSION COEFFICIENTS       04JA8
C*************************************************************************
C                           . . . . . . . . . . . . . . . . . . . . . . . . ..
```

```
         DO 1000  J = NF , NL                                           01FE8  .
              JJ = J                                                    04JA8  .
C        FORM SUB-MATRICES                                              04JA8  .
      CALL EXCUT ( L1, L2, L3, ET2, DT, CC, ET1, EE, FF, ML, JJ,         11NO8
     1             N1, N2, N3, QI, QD1, QD2, QD3, QI1, QI2, QI3,          11NO8
     2             SK, RHO, DF, SF )                                    11MR9
      CALL RFV    ( AM2, AM1, L1 , 1   , NK )                           20MY8  .
      CALL RFV    ( AM1, A   , L1 , 1   , NK )                          20MY8  .
         IF( ML ) 210, 180, 180                                         04JA8  .
  180 CALL RFV    ( BM1, B   , L1 , L1 , NK )                           20MY8  .
         GO TO 220                                                      04JA8  .
C        READ D AND E MULTIPLIERS FROM TAPE 3                           17JA8  .
  210 CALL IOBIN(4HREAD   ,3,D,K2)                                      25AG9IO
      CALL IOBIN ( 7HREADSKP,3,E,K2 )                                   31OC9
         GO TO 280                                                      04JA8  .
C        CALCULATE RECURSION MULTIPLIER E                               04JA8  .
  220 CALL RFV    ( E   , EP1, L1 , L1 , NK )                           20MY8  .
C        CALCULATE RECURSION MULTIPLIER EP1                             23MR8  .
      CALL MBFV   ( ET1, BM1, EP1, L1 , L1 , NK , N1 )                  20MY8  .
      CALL ABF    ( DT , EP1, EP1, L1 , NK , N2 )                       23MR8  .
C        CALCULATE RECURSION MULTIPLIER D                               04JA8  .
      CALL SMFF   ( E   , BM1, D   , L1 , NK )                          05MR8  .
      CALL RFV    ( BM1, CM1, L1 , L1 , NK )                            20MY8  .
      CALL RFV    ( CM1, C   , L1 , L1 , NK )                           20MY8  .
      CALL MBFV   ( ET2, BM1, C   , L1 , L1 , NK , N1 )                 20MY8  .
      CALL ASFV   ( D   , C   , D   , L1 , L1 , NK , +1 )               20MY8  .
      CALL ABF    ( CC , D   , D   , L1 , NK , N3 )                     01FE8  .
      CALL INVR6 ( D   , L1 , NK )                                      15MR8  .
      CALL CFV    ( D   , L1 , L1 , NK , -1.)                           20MY8  .
C        CALCULATE RECURSION COEFFIECENT C                              04JA8  .
      CALL MFB    ( D   , EE , C   , L1 , NK , N1 )                     20MR8  .
C        CALCULATE RECURSION COEFFIECENT B                              04JA8  .
      CALL MFFT   ( D   , EP1, B   , L1 , NK )                          23MR8  .
C        CALCULATE RECURSION COEFFIECENT A                              04JA8  .
  280 CALL MFFV   ( E   , AM1, A   , L1 , 1    , NK )                   20MY8  .
      CALL MBFV   ( ET2, AM2, ATM, L1 , 1    , NK , N1 )                20MY8  .
      CALL ASFV   ( A   , ATM, AM2, L1 , 1    , NK , +1 )               20MY8  .
      CALL ASFV   ( AM2, FF , ATM, L1 , 1    , NK , -1 )                20MY8  .
      CALL MFFV   ( D   , ATM, A   , L1 , 1    , NK )                   20MY8  .
C        SAVE A COEFFICIENT ON TAPE 1                                   04JA8  .
      CALL IOBIN(6HWRITER,1,A,NK)                                       19AG9IO
  290    IF (IOBIN(4HTEST,1))  290, 300, 300                            18OC9
  300    IF ( ML )  400, 600, 500                                       18OC9
  400 CALL IOBIN(7HREADSKP,2,W,K2 )                                     25AG9IO-
      CALL IOBIN(7HREADSKP,2,W,K2 )                                     16OC9
  450    IF  (IOBIN(4HTEST,2))  450, 1000, 1000                         18OC9
C        SAVE D AND E MULTIPLIERS ON TAPE 3                             17JA8  .
  500 CALL IOBIN(5HWRITE ,3,D,K2)                                       25AG9IO-
      CALL IOBIN(5HWRITE ,3,E,K2)                                       25A
      CALL IOBIN (6HWRITER,3 )                                          22AG9IO
C        SAVE B AND C COEFFICIENTS ON TAPE 2                            17JA8  .
  600 CALL IOBIN(6HWRITER,2,B,K2)                                       19AG9IO
      CALL IOBIN(6HWRITER,2,C,K2)                                       19AG9IO
 1000    CONTINUE                                                       04JA8  .
C                   . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ..
C*************************************************************************
```

```
C         BEGIN BACKWARD PASS  --  COMPUTE RECURSION EQUATION                  04JA8
C*******************************************************************************
      CALL IOBIN(4HBKSP,1)                                                     19AG9IO
      CALL IOBIN(4HBKSP,2)                                                     19AG9IO
      CALL IOBIN(4HBKSP,2)                                                     19AG9IO
      CALL RFV ( W(NF,NL), A , L1, 1, NK )                                     10AP9
      CALL IOBIN(4HBKSP,1)                                                     19AG9IO
      CALL IOBIN(4HBKSP,2)                                                     19AG9IO
      CALL IOBIN(4HBKSP,2)                                                     19AG9IO
      CALL IOBIN(7HREADSKP,1,A,NK)                                             25AG9IO-
      CALL IOBIN(7HREADSKP,2,B,K2)                                             25AG9IO-
      CALL IOBIN(7HREADSKP,2,C,K2)                                             25AG9IO-
      CALL IOBIN(4HBKSP,1)                                                     19AG9IO
      CALL IOBIN(4HBKSP,2)                                                     19AG9IO
      CALL IOBIN(4HBKSP,2)                                                     19AG9IO
      CALL MFFV      ( B,  W(NF, NL),  AM1,  L1,  1,   NK )                    10AP9
      CALL ASFV      (  A,  AM1,  W(NF,NL-1),  L1,  1,  NK,  +1  )             10AP9
              NLM2 = NL - 2                                                    20MY8
C                          . . . . . . . . . . . . . . . . . . . . . . . . . . . . ..
          DO 2000  L = NF , NLM2                                               20MY8  .
               J = NLM2 + NF - L                                               20MY8  .
      CALL IOBIN(4HBKSP,1)                                                     19AG9IO
      CALL IOBIN(4HBKSP,2)                                                     19AG9IO
      CALL IOBIN(4HBKSP,2)                                                     19AG9IO
C         READ A COEFFICIENT FROM TAPE 1                                       04JA8  .
      CALL IOBIN(7HREADSKP,1,A,NK)                                             25AG9IO
C         READ B AND C COEFFICIENTS FROM TAPE 2                                17JA8  .
      CALL IOBIN(7HREADSKP,2,B,K2)                                             25AG9IO-
      CALL IOBIN(7HREADSKP,2,C,K2)                                             25AG9IO-
      CALL IOBIN(4HBKSP,1)                                                     19AG9IO
      CALL IOBIN(4HBKSP,2)                                                     19AG9IO
      CALL IOBIN(4HBKSP,2)                                                     19AG9IO
      CALL MFFV      (  B,  W(NF,J+1),  AM1,  L1,  1,  NK  )                   10AP9
      CALL MFFV      (  C,  W(NF,J+2),  AM2,  L1,  1,  NK  )                   10AP9
      CALL ASFV  ( AM1, AM2, AM1, L1 , 1  , NK , +1 )                          20MY8  .
      CALL ASFV      (  A,  AM1,  W(NF,J),  L1,  1,  NK,  +1  )                10AP9
 2000     CONTINUE                                                             04JA8  .
C                  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ..
      RETURN                                                                   4JA8
      END                                                                      04JA8
C
```

THE AUTHORS

Allen E. Kelly is currently an Associate Professor in the School of Civil Engineering, Oklahoma State University, Sillwater. At The University of Texas at Austin, he was an Assistant Professor in the Department of Civil Engineering and a Research Engineer with the Center for Highway Research, where his research concerned computer simulation of beam-columns and slab systems. He is the author of several technical papers and reports.

Hudson Matlock is a Professor of Civil Engineering at The University of Texas at Austin. He has a broad base of experience, including research at the Center for Highway Research and as a consultant for other organizations, such as the Alaska Department of Highways, Shell Development Company, Humble Oil and Refining Company, Penrod Drilling Company, and Esso Production Research Company. He is the author of over 50 technical papers and 25 research reports and received the 1967 ASCE J. James R. Croes Medal. His primary areas of interest in research include (1) soil-structure interaction, (2) experimental mechanics, and (3) development of computer methods for simulation of civil engineering problems.