REPORT NO.
UCB/SEMM-91/01

STRUCTURAL ENGINEERING,
MECHANICS and MATERIALS

CAL - 91

COMPUTER ASSISTED LEARNING

OF STATIC AND DYNAMIC ANALYSIS OF STRUCTURAL SYSTEMS

by

EDWARD L. WILSON

JANUARY 1991

DEPARTMENT OF CIVIL ENGINEERING
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA

CAL-91

COMPUTER ASSISTED LEARNING OF STATIC AND DYNAMIC ANALYSIS OF STRUCTURAL SYSTEMS

BY

EDWARD L. WILSON

UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA
JANUARY 1991

ACKNOWLEDGEMENTS

CAL-91 is the latest version of a series of computer programs for the Computer Assisted Learning of Structural Analysis which have been developed by the author. The first program was a matrix interpretive program, SMIS, which was developed in 1960 under the direction of Professor Ray Clough. The author would like to take this opportunity to thank Professor Clough for his continued support and suggestions for the improvement of the program during the past 30 years.

Several graduate students have contributed to the development of CAL. Martin Button, John Dickens and Eduardo Bayo programmed the frequency domain commands. Marc Hoit participated in the development and verification of the CALSAP development system. Pierre Leger added the Ritz vector subroutines. Since all commands have been reprogrammed for this version of the program the author is responsible for all errors which may exist in the CAL program or the users manual.

The CAL series of computer programs were developed as part of the teaching responsibilities of the author at the University of California at Berkeley, without external financial support. Research on the development of many of the numerical methods for the dynamic analysis of structures, which are special CAL commands, was funded by the National Science Foundation during the past 20 years.

TABLE OF CONTENTS

1. INTRODUCTION

At the present time, a large number of computer programs exist which evaluate the displacements and member forces within complex three-dimensional structures. After the computer model and input data is prepared, the analysis process is completely automated. Therefore, it is now possible for students, professional engineers, and professors to perform structural analysis and design using computer programs without a complete knowledge of the approximations which have been incorporated in the computer program. In addition, many individuals do not assume the responsibility or have the ability to independently verify the results produced by a structural analysis program.

The major reasons for the misuse of computer programs for structural analysis and design is the lack of understanding of the engineering fundamental of equilibrium, force-deformation requirements, and displacement compatibility equations. Another reason, which contributes to the inappropriate use of computer programs, is that the traditional hand calculation methods of structural analysis are not used within modern computer programs. Therefore, many engineers use structural analysis programs as "black boxes" in which the approximations used within the program are not appreciated. This is particularly true if a program is used for three-dimensional dynamic response analysis.

One possible educational solution to this problem is to require students to write their own programs for structural analysis which are based on the direct stiffness method and on modern three-dimensional numerical methods of dynamic analysis. This would result in an improvement in the students' programming ability; however, this may be accomplished at the expense of a basic understanding of structural behavior.

Since programming knowledge is not essential to the responsible use of a structural analysis program, an alternative approach is to use a special language which was designed to subdivide structural analysis into a sequence of logical steps. The use of such an approach requires that the student understand basic structural theory and behavior without the need to perform a large number of time-consuming hand calculations. This philosophy was the motivation for the development of the CAL series of programs.

1.1 PURPOSE OF PROGRAM

The basic purpose of the CAL language is to bridge the gap between traditional methods of teaching structural analysis and the use of automated structural analysis programs. As a result of using CAL it is hoped that engineers will understand the theory and approximations which are used in modern structural analysis programs.

Since the FORTRAN program can be easily modified and new CAL commands added, the program can be used as an effective research tool. New numerical algorithms for the static or dynamic analysis of structures can be programmed and tested within a few hours.

CAL has been used to verify other programs for structural analysis. Simple static and dynamic problems can easily be solved with CAL and their results compared with programs which are being used for the first time.

It should be remembered, however, that the basic purpose of CAL is educational. No attempt has been made to extend the capacity of the program by using sparse matrix methods or to effectively use disk storage. Since CAL is designed for the solution of small educational and research problems, under 100 degrees of freedom, it has limited values for the solution of large practical structures. One of the many large capacity programs, with graphics and design post processors, should be used for this class of problems.

1.2 DEVELOPMENT OF PROGRAM

The last release of the program was CAL-86. The present version of the program has several additional commands and more examples. Also, the theoretical background of several commands has been expanded. It is hoped that the ease of use of the program has been improved.

The FORTRAN files of all programs are given in order for the user to verify the exact numerical methods which are used within the program. The program is based on the use of the CALSAP development system. It is hoped that students will study these listings and learn to appreciate both the good and poor programming methods. However, the program is protected by copyright laws; therefore, it cannot be resold, in whole or in part, and it cannot be used for commercial purposes without the written permission of the author.

The specific version of the FORTRAN source statements, which is given on the CAL program disk, is for MS-DOS microcomputer systems and can be directly compiled with the Microsoft FORTRAN Compiler Version 3.31. However, since the program is written in standard FORTRAN 77 it can be easily modified for other compilers and made to operate on other small micro or large mainframe computer systems.

1.3 REQUIRED USER BACKGROUND

Since the CAL program performs basic matrix operations it can be used in lower division courses which use matrix methods. At the University of California at Berkeley it is used in approximately five different courses on the static and dynamic analysis of structures at both the graduate and undergraduate level. Therefore, it is a program which requires additional theoretical information in order to solve structural problems. Chapter 6 contains a reprint of a survey paper, written in 1977, which presents additional information on the Dynamic Analysis Commands.

The CAL program, as presented in this manual, is to be used on personal computers which use the DOS operating system. Appendix A contains a summary of the DOS operating system. Students which require a summary of the most common DOS operations should read this material.

A working knowledge of a standard text editor is required. The ShareWare editor EDED is included on the CAL disk for individuals who do not have experience with their own editor.

1.4 PROGRAM INSTALLATION

The **CAL.EXE** executable program is the only file which is required in order to execute all CAL commands. It can be executed directly from the diskette or it can be copied to a directory on the hard disk which is defined in the PATH command.

The DOS computer must have a minimum of 512 k of storage. A hard disk is not required. The CAL program will run on computers without a floating point processor. However, it will operate approximately five times faster, for large problems, if a floating point processor exists. For both hardware configurations, all floating point calculations are carried out in double-precision 64 bit accuracy.

1.5 THE CAL COMMAND LANGUAGE

Data on a "command line" must be separated by commas, or, one or more blanks. A typical CAL command line has the following form:

COMMAND M_1 M_2 - - A=a_1,a_2 - B=? : (Comment)

Where "COMMAND" is a one to six character name of the CAL command
"M_i" is a one to four character array name
"a_i" is optional data for some commands

The notation "M+" indicates that a new array will be created by the operation. If the array name for new data has previously been used the old array will be eliminated before the operation is executed. The notation "M-" indicates that the array will be modified by the CAL operation.

Typical data, "a_i", to be used by the operation can be in either integer or floating point form. In the case of a floating point number, it can have the form of arithmetic statements. For example, 2.5+4*2-6/2 will be interpreted as ((2.5+4)*2-6)/2.

A "C" in column one of a command or data line indicates that the line will be a comment line which is used to make the CAL input file more readable.

1.6 DATA PREPARATION

CAL is a computer program which is designed to interpret a sequence of commands which are supplied by the user. The commands can be given directly in an "interactive mode" or the program can read the commands from a "batch data file". Commands for matrix analysis, direct stiffness structural analysis and dynamic response analysis are possible.

CAL commands and data can be entered interactively or supplied within a data input file which is prepared by any file editor which produces a generic character file without "hidden" characters. The ShareWare editor, EDED, can be used and is distributed with the CAL program. The name of the CAL input data file must not have an extension.

1.7 EXECUTION OF THE CAL PROGRAM

After the CAL-91 program is executed, the program asks the user for the "**name**" of the input data file which contains the CAL batch command. The program will give the **CAL-91>** prompt which indicates that any CAL command can be entered interactively. CAL commands which are contained within the file are executed by the **RUN** command which is entered interactively. The results of every CAL run are saved on the output file name "**name.OUT**". If no file name is specified a default name of "**I**" is used and all interactive commands are saved on the file **I.OUT**.

If a CAL run is terminated by an **EXECUTION ERROR** or the **STOP** command all arrays which are within the computer storage are saved on the file "name.COR". The CAL program can then be restarted, with the same problem "name", with the **RESUME** command. The CAL **LIST** and **PRINT** commands can then be used interactively to examine the data arrays contained within the computer storage.

After the CAL program is terminated the output file, **name.OUT**, can be printed with the DOS PRINT command. Also, it can be displayed on the computer console with the DOS TYPE command. In addition, it can be modified, dissected and selectively printed with the same editor which was used to create the input data file.

2.
BASIC CAL COMMANDS

The group of CAL commands presented in this section controls the flow of execution of the CAL commands. Also, it allows for input, output, duplication or generation of arrays within the computer's memory, RAM, or the transfer of arrays between RAM and low speed disk storage. Note that the "+" sign after an array name indicates that the array is created by the CAL command and the "-" sign after an array name indicates that the CAL command modifies or deletes the data within the array.

RUN
The RUN command will terminate the interactive mode and the subsequent CAL command will be read directly from the input batch file. This command can only be given in the interactive mode.

RETURN
The RETURN command will terminate the execution of the batch input mode and return the CAL program to the interactive mode. This command can only be given in the batch input file mode.

HELP or H
If the HELP command is executed in the interactive mode a list of all possible CAL commands will be displayed.

SAVE or STOP or S
The SAVE command will terminate the execution of the program and return control to the computer's **D**isk **O**perating **S**ystem. All arrays which are contained in the computer's storage will be saved in the file "**name.COR**" where "name" is the problem (input file) name which has been specified by the user.

RESUME
The RESUME command reads all arrays from the file "**name.COR**" and allows the user to continue to enter CAL interactive commands from the point the previous CAL execution was terminated.

QUIT or Q
The CAL command QUIT causes the execution of CAL to be terminated. The incore data base is not saved.

LIST or L
If the interactive LIST command is executed a list of the name and size of all arrays which are contained in the computer storage is displayed.

LOAD M_1+ R=? C=?
The LOAD command will create a matrix named "M_1" with "R" rows and "C"columns. The data must immediately follow the LOAD command. The data must be supplied one row per line. The data is separated by commas, or, one or more blanks. A line of data may be continued on the next line by the use of a "\" at the end of the line. If the data for a row is greater than 160 characters the matrix must be loaded by the use of submatrix operations. All input data should be checked with a PRINT command.

ZERO M_1+ R=? C=? T=? D=?
The ZERO command will create a R x C matrix named M_1. If "T=?" is specified all terms of the matrix will be set to "T". If the matrix is square the diagonal terms will be set to "D".

PRINT or P M_1
The PRINT command will cause the matrix "M_1" to be displayed on the terminal and written to the "????.OUT" print file, which can be printed by the DOS PRINT command.

DELETE or D M_1-
The array named M_1 will be deleted and the storage within the computer will be compacted.

MODIFY M_1-
The MODIFY command can be used interactively to modify individual terms in the matrix named M_1.

DUP M_1 M_2+
The DUP command forms a new matrix named M_2 in which all terms are equal to the terms in matrix M_1.

WRITE M_1 and **READ M_1**
The WRITE command can be used to save the array named M_1 as a binary disk file called "name.M_1. Therefore, M_1 must not have over a three character name. The read command reads the array on the binary file named "name.M_1" into RAM. Since files can be renamed by the DOS command REN, data generated in one CAL problem can be used as input to another CAL problem with a different "name".

LOOP endloop N=?
The LOOP command will cause the group of CAL commands in the input file before the line which contains the one to six character name **endloop** to be executed "N" times. The LOOP will be executed "N" times unless terminated by an IF operation. LOOP commands cannot be nested. If more than one LOOP command is is in a data file each **endloop** name must be different.

IF M_1 M_2
If the absolute value of $M_1(1,1)$ is less than $M_2(1,1)$ the LOOP executed will be terminated. This command should be placed immediately before the **endloop** line.

endloop
The user-selected name **endloop** must start in column one and is used to designate the last command in the execution of a group of CAL commands by the **LOOP** command. The one to six character **endloop** name cannot be the same name as one of the existing CAL commands.

3. STANDARD MATRIX OPERATION COMMANDS

This group of CAL commands is intended to perform standard matrix operations. Note that the **TMULT** command eliminates, for most problems, the need to perform a matrix transpose. Also, it is always more efficient to solve a set of equations directly without using matrix inversion and multiplication.

ADD M_1- M_2
This operation replaces the matrix M_1 with $M_1 + M_2$

SUB M_1- M_2
The **SUB** operation replaces the matrix M_1 with $M_1 - M_2$

MULT M_1 M_2 M_3+
The **MULT** command creates the matrix M_3 which is the product of the matrices M_1 and M_2. (The number of numerical operations required for matrix multiplication is NxMxL; where, M_1 is a N by M matrix and M_2 is a M by L matrix.)

TMULT M_1 M_2 M_3
Same as the **MULT** command except that M_1 is stored in transposed form.

TRAN M_1 M_2+
The **TRAN** command forms the matrix M_2 which is the transpose of the matrix M_1.

SCALE M_1+ M_2
The **SCALE** command multiplies each term in matrix M_1 by $M_2(1,1)$.

SOLVE A- B- S=? EQ=?

The **SOLVE** command operates on the matrix equation **AX = B** where **A** is a symmetric matrix and **B** is specified. The results **X** are stored in the same location as the **B** matrix. The following options are possible:

S=0 The matrix **A** is triangularized and **B** is replaced by the solution matrix **X**.
S=1 The matrix **A** is triangularized only.
S=2 The matrix **B** is reduced only - **A** must have been previously triangularized.
S=3 The matrix **B** is replaced by the solution matrix **X** by backsubstitution only.

EQ= The number of equations to be reduced - to be used in substructure analysis.

Any nonsingular set of equations can be made symmetric if both sides of the equation are multiplied by the transpose of **A**.

(The number of numerical operations required to triangularize the N x N matrix **A** is $N^3/6$. The number of operations required for forward reduction is $N^2L/2$ and for backsubstitution is $N^2L/2$; where, L is the number of columns in the matrix **B**.)

INVERT A-

The symmetric matrix A is replaced by its inverse.
(The number of numerical operations required to invert a symmetric matrix is $N^3/2$)

DUPSM M_1 M_2+ R=? C=? L=RL,CL

The command DUPSM creates a new matrix M_2 with "R" rows and "C" columns. The term $M_2(1,1)$ is identical to the term M_1(RL,CL).

STOSM M_1- M_2 L=RL,CL

The command STOSM stores the submatrix M_2 in matrix M_1. The term $M_2(1,1)$ is located at row RL and column CL in matrix M_1.

DUPDG M_1 M_2+

The command DUPDG creates a row matrix M_2 from the diagonal terms of matrix M_1.

STODG M_1- M_2

The command STODG stores the row matrix M_2 on the diagonal of the matrix M_1.

EXAMPLE OF CAL COMMANDS

The set of equations shown below must be solved. Since the **SOLVE** command only solves symmetrical systems we must use additional operations to make the system symmetrical.

$$\begin{bmatrix} 0 & 3.4 & -2.0 \\ 4.0 & -1.0 & 0.0 \\ 0.0 & 6.0 & 4.0 \end{bmatrix} \begin{bmatrix} X1 \\ X2 \\ X3 \end{bmatrix} = \begin{bmatrix} 3.0 \\ -1.0 \\ 4.0 \end{bmatrix}$$

An input file of the following form must be prepared:

```
C  SOLUTION OF NONSYMMETRIC SET OF EQUATIONS
C
LOAD A   R=3   C=3
0    3.4    2.0
4   -1.0      0
0    6        4
PRINT A
LOAD B   R=3 C=1
3
-1
4
PRINT B
TMULT A A ATA
TMULT A B X
SOLVE ATA X
PRINT X          : Print Results
MULT A X ERR
SUB ERR B
PRINT ERR        : Each value should be very close to zero
STOP             : Return to the DOS operating system
```

If the above data file is called "**CALEX**" it can be executed by giving the DOS command **CAL91.** The CAL program will then ask the user for the "name" of the input file and the user will respond by entering CALEX and a carriage return. The above data is then executed by the user entering the CAL **RUN** command. The results will be on the output file "**CALEX.OUT**" which can be transferred to the printer by the DOS PRINT operation.

The CAL program disk contains the file **CALEX**; therefore, the user can check the program by executing the above data file.

4. DIRECT STIFFNESS COMMANDS

The direct stiffness operations allow for the automatic formulation of element stiffness matrices, the direct addition of element stiffnesses to form the global stiffness matrix and the calculation of member forces in a local member coordinate system. The first step in the use of these commands is for the user to identify all displacement degrees of freedom at the joints of the structural system. These displacements should be numbered U_1, U_2, - - - -U_N. The corresponding external joint loads will be R_1, R_2 - - - - R_N. One equilibrium equation must be written in the direction of each unknown displacement. The structural members should also be numbered from 1 to M.

1.1 MATRIX FORMULATION OF THE SLOPE-DEFLECTION METHOD

The classical slope-deflection method, for horizontal beams and vertical columns, is a well-known approach for structural analysis. However, it is identical to the direct stiffness method in which axial deformations are assumed to be zero. Therefore, it will be used to illustrate the basic steps involved in the direct stiffness method.

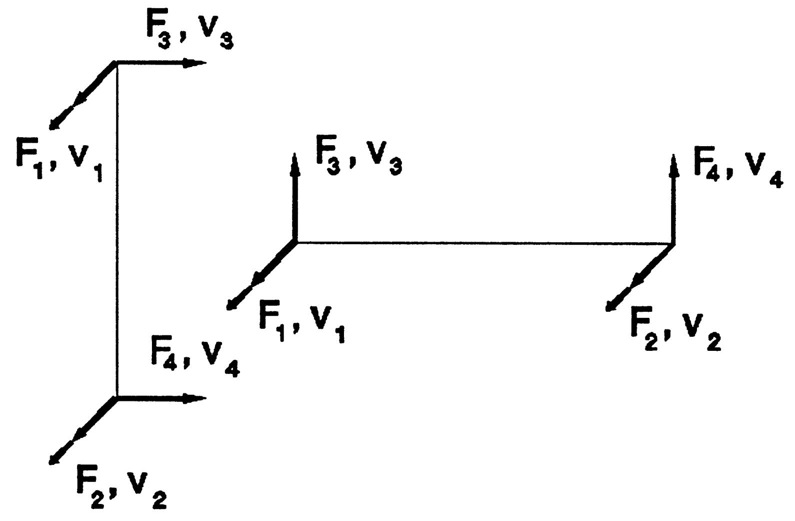


Figure 4.1 POSITIVE DEFINITION OF FORCES AND DISPLACEMENTS

For the sign convention shown in Figure 4.1 the classical slope-deflection equations can be written as

$$F_1 = \frac{EI}{L}\left[4v_1 + 2v_2 + \frac{6(v_3 - v_4)}{L}\right]$$

$$F_2 = \frac{EI}{L}\left[2v_1 + 4v_2 + \frac{6(v_3 - v_4)}{L}\right]$$

$$F_3 = -F_4 = \frac{(F_1 + F_2)}{L}$$

The relative displacement between the ends of the member, $v_3 - v_4$, has been written in terms of the absolute global displacements. Now, these equations can be written, for a typical member "M", as the following symmetric matrix equation:

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}_M = \frac{4EI}{L}\begin{bmatrix} 4 & 2 & \frac{6}{L} & -\frac{6}{L} \\ 2 & 4 & \frac{6}{L} & -\frac{6}{L} \\ \frac{6}{L} & \frac{6}{L} & \frac{12}{L^2} & -\frac{12}{L^2} \\ -\frac{6}{L} & -\frac{6}{L} & -\frac{12}{L^2} & \frac{12}{L^2} \end{bmatrix}_M \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}_M \qquad 4.1$$

Or symbolically, $\boldsymbol{F_M} = \boldsymbol{K_M}\,\boldsymbol{v_M}$; where, $\boldsymbol{K_M}$ is the 4 x 4 element stiffness matrix.

In order to illustrate the direct stiffness method we will consider the rigid frame example structure shown in Figure 4.2. The first step required, in the analysis of a structure by the slope defection method, is to select the independent displacement degrees of freedom. Since axial deformations are set to zero all lateral displacements at a story level are the same. Therefore, there are only two lateral displacements, u_1 and u_2, and one vertical displacement of the center column, u_3. Since the base of the right column is pinned and can rotate there are six rotational degrees of freedom, u_4 to u_9. It is apparent that we can only apply loads, R_1 to R_9, in the same directions as the displacements, u_1 to u_9. For example, load R_2 is the total lateral load on the three joints at the first story level of the structure.

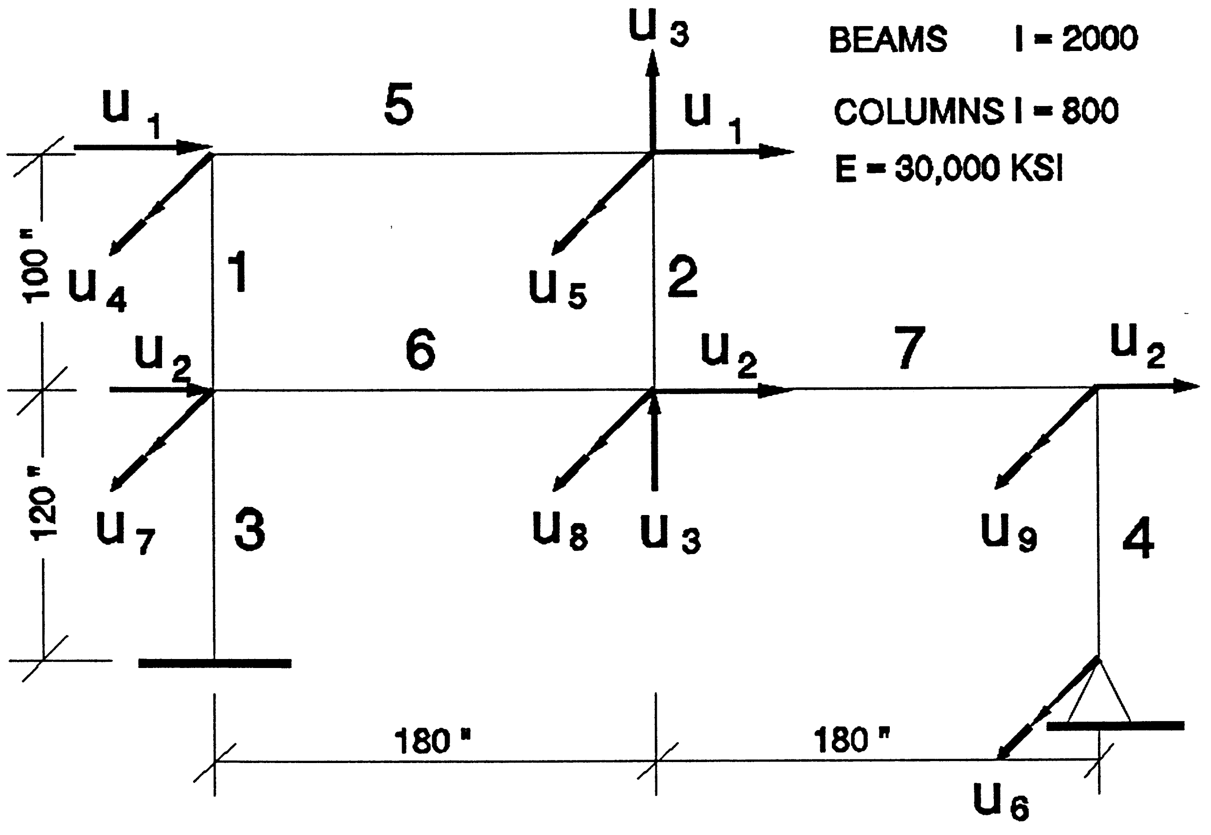


Figure 4.2 DISPLACEMENT DEGREES OF FREEDOM FOR RIGID FRAME

The next step, after the degrees of freedom are identified, is to write one equilibrium equation for each degree of freedom. For this structure, the nine equilibrium equations can be written in the following matrix form:

$$\begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \\ R_7 \\ R_8 \\ R_9 \end{bmatrix} = \begin{bmatrix} F_3 \\ F_4 \\ 0\_ \\ F_1 \\ 0\_ \\ 0\_ \\ F_2 \\ 0\_ \\ 0\_ \end{bmatrix}_1 + \begin{bmatrix} F_3 \\ F_4 \\ 0\_ \\ 0\_ \\ F_1 \\ 0\_ \\ 0\_ \\ F_2 \\ 0\_ \end{bmatrix}_2 + \begin{bmatrix} 0\_ \\ F_3 \\ 0\_ \\ 0\_ \\ 0\_ \\ 0\_ \\ F_1 \\ 0\_ \\ 0\_ \end{bmatrix}_3 + \begin{bmatrix} 0\_ \\ F_3 \\ 0\_ \\ 0\_ \\ 0\_ \\ F_2 \\ 0\_ \\ 0\_ \\ F_1 \end{bmatrix}_4 + \begin{bmatrix} 0\_ \\ 0\_ \\ F_4 \\ F_1 \\ F_2 \\ 0\_ \\ 0\_ \\ 0\_ \\ 0\_ \end{bmatrix}_5 + \begin{bmatrix} 0\_ \\ 0\_ \\ F_4 \\ 0\_ \\ 0\_ \\ 0\_ \\ F_1 \\ F_2 \\ 0\_ \end{bmatrix}_6 + \begin{bmatrix} 0\_ \\ 0\_ \\ F_3 \\ 0\_ \\ 0\_ \\ 0\_ \\ 0\_ \\ F_1 \\ F_2 \end{bmatrix}_7 \qquad (4.2)$$

From matrix Equation (4.2), we can also create a table indicating what member forces contribute to each of the nine global equilibrium equations.

Table 4.1 Member FORCE Contributions to GLOBAL EQUILIBRIUM EQUATIONS

Member	1	2	3	4	5	6	7
F_1	4	5	7	9	4	7	8
F_2	7	8		6	5	8	9
F_3	1	1	**2**	**2**			3
F_4	**2**	**2**			3	3	

Equation (4.2) is nothing more than a simple statement, in matrix form, that the external applied loads equals the sum of the forces acting at the ends of the seven elements. It clearly represents the summary of nine different free body diagrams. It is suggested that the student draw each of these diagrams in order to understand this simple, but extremely important, force equilibrium statement. For example, $R_2 = F_{41} + F_{42} + F_{33} + F_{34}$.

These nine equilibrium equations can be directly expressed in terms of joint displacements by the substitution of Equation (4.1) for all seven members. Therefore, the force-displacement relationship for each member is satisfied.

The final step involves the use of the joint displacement compatibility condition. The displacements at the ends of all members connected to a joint must have the same global displacements. For this example, Table 4.2 defines the relationship between the four member displacements for each member and the nine global joint displacement numbers.

Table 4.2 Member DISPLACEMENTS Compatible with GLOBAL DISPLACEMENTS

Member	1	2	3	4	5	6	7
v_1	4	5	7	9	4	7	8
v_2	7	8		6	5	8	9
v_3	1	1	2	2			3
v_4	2	2			3	3	

It is apparent that the integer arrays, given in Tables 4.1 and 4.2, are identical. The blank positions are reaction points and have zero displacements. The equilibrium equations can be written in the following form:

$$\boldsymbol{R} = [\boldsymbol{K}_1^* + \boldsymbol{K}_2^* + \text{-----} + \boldsymbol{K}_7^*]\,\boldsymbol{u} = \sum \boldsymbol{K}_M\,\boldsymbol{u} = \boldsymbol{K}_T\boldsymbol{u} \tag{4.3}$$

Each element stiffness can be expanded to a 9 x 9 matrix, $\boldsymbol{K}_M^*$, and the total stiffness matrix, $\boldsymbol{K}_T$, can be formed by direct matrix addition. Or, the total stiffness matrix can be formed directly. This would involve the addition of each term in the 4 x 4 element stiffness matrix, $\boldsymbol{K}_M$, into the correct location in the 9 x 9 total stiffness matrix, $\boldsymbol{K}_T$. The procedure is time-consuming, repetitive, and prone to error if done by hand calculations. For this reason special commands have been added to the CAL program to perform these direct additions of stiffness operations. The ADDK command adds the rows in the element stiffness to the correct rows in the equilibrium equations, Table 4.1, and to the appropriate columns, Table 4.2.

4.2 SUMMARY OF THE DIRECT STIFFNESS COMMANDS

SLOPE KM E=? I=? L=?

The slope command forms the 4 x 4 member stiffness matrix named **KM** for a beam or a column. Where "E" equals the modulus of elasticity, "I" equals the moment of inertia and "L" equals the length of the member.

LOADI M_1 R=? C=?

The **LOADI** command allows an integer array named **M1** to be loaded which has "R" rows and "C" columns. The integer array given in Table 4.1 or 4.2 can be loaded with this CAL command. The array loaded by the **LOADI** command can be checked by the CAL **PRINT** command.

ADDK K KM ID N=?

The element stiffness matrix named "KM" is added to the total stiffness matrix named "K". The row and column numbers where the terms are to be added are obtained from the "N" column of the integer array loaded by the **LOADI** command. The **ADDK** command is used for each member to add the element stiffness matrices to the total (global) stiffness matrix.

After the joint loads are defined, the joint equilibrium equations are solved for joint displacements by the **SOLVE** command. The use of the **MEMFRC** command for each member allows member forces to be calculated.

MEMFRC T U ID F+ N=?
The member forces are evaluated by the multiplication of the matrix named "**T**" by the joint displacement matrix named "**U**" and the results are stored in a matrix named "**F**". The joint displacements which are to be used in multiplication are obtained from the "**N**" column of the integer array named "**ID**". If "**T**" is the element stiffness matrix the member forces are given according to the global sign convention. If "**T**" is a special member force-displacement transformation matrix the member forces will be given in a local member coordinate system.

4.3 EXAMPLE OF SLOPE-DEFLECTION ANALYSIS

A complete analysis of the structure shown in Figure 4.1 will now be given. In order to illustrate that the properties of the stiffness matrix are not a function of the loading on the structure let us use the following two different lateral load conditions in the same analysis:

$$\boldsymbol{R}^T = \begin{bmatrix} 10 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 20 & -10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The CAL input data file is shown in Table 4.3 and is on the CAL program disk under the name SLOPEX. It is suggested that the student run this example problem and PRINT the output file named "**SLOPEX.OUT**". Shear and bending moment diagrams should be plotted and the overall static of both load conditions should be checked.

Since this problem could have been solved by the traditional slope-deflection method it is extremely important that each step of the direct stiffness method, using matrix notation and the CAL program, be completely understood before other types of elements are used.

Some important points to remember when using the direct stiffness method are:

1. The positive direction of forces and displacements for both members and joints must be the same.

2. The basic equations which are written and solved are simple equilibrium equations in which the external loads are set equal to the sum of the forces acting ot the ends

of the elements.

3. The integer "ID" array represents a summary of the equilibrium equations as well as a summary of joint compatibility conditions.

4. Matrix notation and the CAL **ADDK** and **MEMFRC** commands ares simple operations which minimize bookkeeping functions.

Table 4.3 CAL INPUT DATA FILE FOR SLOPE-DEFLECTION EXAMPLE

```
C SLOPE-DEFLECTION ANALYSIS EXAMPLE
C
SLOPE KCU L=100 I=800  E=30000 : FORM STIFFNESS FOR MEMBERS 1 AND 2
SLOPE KCL L=120 I=80   E=30000 : FORM STIFFNESS FOR MEMBERS 3 AND 4
SLOPE KB  L=180 I=2000 E=30000 : FORM STIFFNESS FOR MEMBER 5, 6 AND 7
LOADI ID R=4 C=7       : LOAD INTEGER (EQILIBRIUM) ARRAY
4  5  7  9  4  7  8
7  8  0  6  5  8  9
1  1  2  2  0  0  3
2  2  0  0  3  3  0
PRINT ID               : CHECK INPUT DATA
ZERO KT R=9 C=9        : START WITH ZERO TOTAL STIFFNESS MATRIX
ADDK KT KCU ID N=1     : ADD MEMBER 1 STIFFNESS TO TOTAL STIFFNESS
ADDK KT KCU ID N=2     : ADD MEMBER 2 STIFFNESS TO TOTAL STIFFNESS
ADDK KT KCL ID N=3     : ADD MEMBER 3 STIFFNESS TO TOTAL STIFFNESS
ADDK KT KCL ID N=4     : ADD MEMBER 4 STIFFNESS TO TOTAL STIFFNESS
ADDK KT KB  ID N=5     : ADD MEMBER 5 STIFFNESS TO TOTAL STIFFNESS
ADDK KT KB  ID N=6     : ADD MEMBER 6 STIFFNESS TO TOTAL STIFFNESS
ADDK KT KB  ID N=7     : ADD MEMBER 7 STIFFNESS TO TOTAL STIFFNESS
LOAD R  R=9 C=2        : LOAD TWO LOAD CONDITIONS
10  10
20 -10
0  0
0  0
0  0
0  0
0  0
0  0
0  0
P R                    : CHECK LOAD DATA
SOLVE KT R             : SOLVE FOR GLOBAL DISPLACEMENTS
P R                    : PRINT DISPLACEMENTS
MEMFRC KCU R ID F N=1  : CALCULATE AND PRINT MEMBER 1 FORCES
P F
MEMFRC KCU R ID F N=2  : CALCULATE AND PRINT MEMBER 2 FORCES
P F
MEMFRC KCL R ID F N=3  : CALCULATE AND PRINT MEMBER 3 FORCES
P F
MEMFRC KCL R ID F N=4  : CALCULATE AND PRINT MEMBER 4 FORCES
P F
MEMFRC KB  R ID F N=5  : CALCULATE AND PRINT MEMBER 5 FORCES
P F
MEMFRC KB  R ID F N=6  : CALCULATE AND PRINT MEMBER 6 FORCES
P F
MEMFRC KB  R ID F N=7  : CALCULATE AND PRINT MEMBER 7 FORCES
P F
STOP                   : RETURN TO DOS OPERATING SYSTEM
```

4.4 TWO-DIMENSIONAL FRAME ANALYSIS WITH AXIAL DEFORMATIONS

FRAME KM TM [K_G] I=? A=? E=? X=X_i,X_j Y=Y_i,Y_j [P=?]

The FRAME command forms the 6 x 6 element stiffness matrix named "KM" and a 4 x 6 force-displacement matrix named "TM" for a general two-dimensional bending member with axial deformations included in the formulation. The properties of the member are given as:

I	= the Moment of Inertia of the member,
A	= the Axial Area of the member, and
E	= the Modulus of Elasticity of the member.

The coordinates of the "i" and "j" ends of the member are defined by X_i,Y_i and X_j,Y_j respectively. Note that the user is responsible for the definition of the "i" and "j" ends of the member. If **P** is specified the 6 x 6 geometric stiffness matrix K_G is formed.

The element stiffness matrix, "KM", is formed with respect to the positive definition of global forces and displacements as shown below.

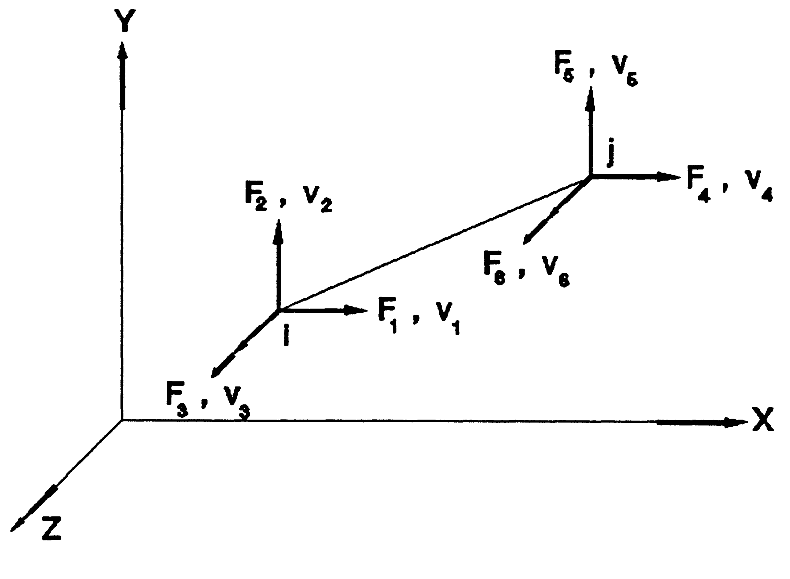


Figure 4.3 POSITIVE DEFINITION OF FORCES AND DISPLACEMENTS

The member forces, with respect to the member's local coordinate system, can be evaluated by the use of the **MEMFRC** operation which multiplies the matrix "**TM**" by the joint displacements. The positive definition of the member forces in the local coordinate system is shown below. The **MEMFRC** command will evaluate the local member forces in the order P_1 - - P_4.

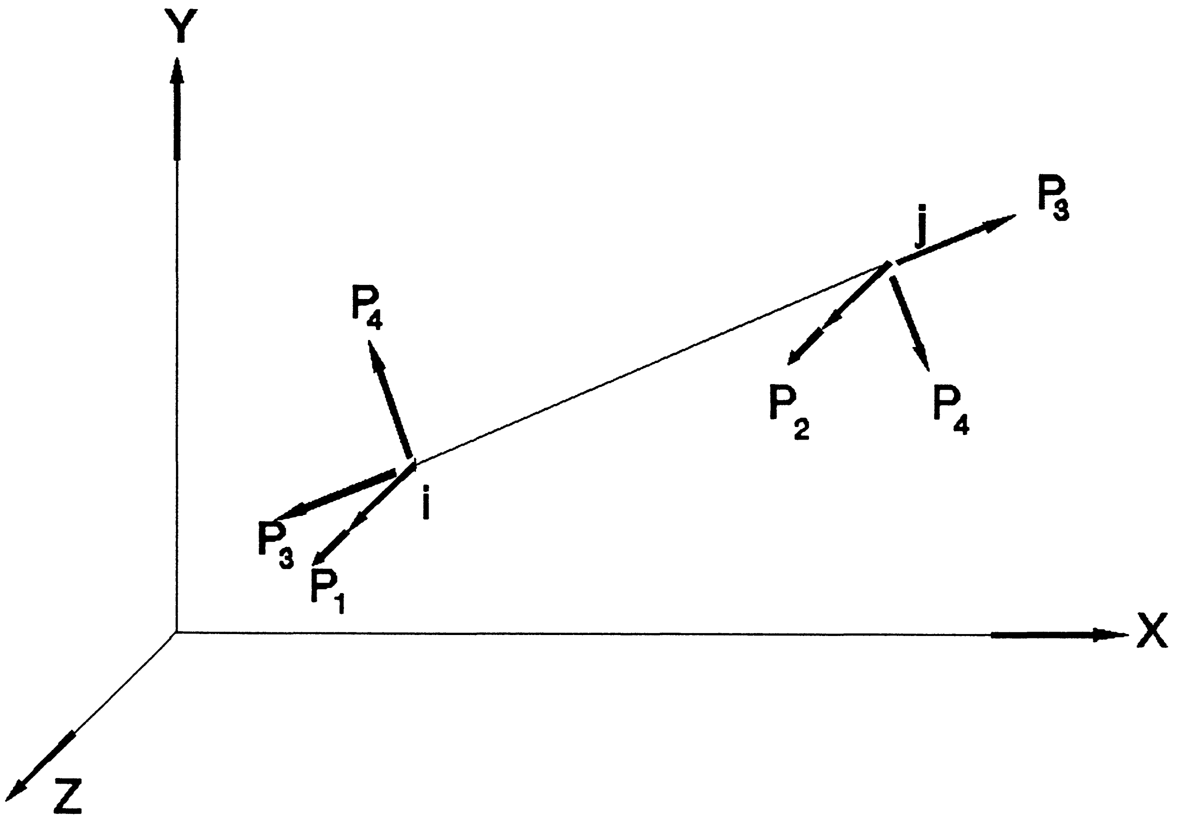


Figure 4.4 POSITIVE DEFINITION OF OUTPUT FORCES

It should be noted that the shear force P_4 is not an independent force. From statics it is

$$P_4 = (P_1 + P_2)/L$$

It is calculated and printed by the program to eliminate this simple hand calculation.

4.5 THREE-DIMENSIONAL MEMBERS

The three-dimensional CAL commands **TRUSS** and **FRAME3** illustrate how structural analysis can be further automated. Both of these reference a node coordinate array, Nx3 in size, named "**XYZ**" which contain the node point coordinates; therefore, only the node point numbers need be used.

TRUSS KM TM A=? E=? N=I,J

The **TRUSS** command forms the 6 x 6 element stiffness matrix named "**KM**" and a 1 x 6 force-displacement matrix named "**TM**" for a general three-dimensional member with axial deformations only included in the formulation. The Axial Area of the member is "**A**". The Modulus of Elasticity of the member is "**E**".

Node numbers **I** and **J** refer to the row numbers in the "XYZ" array. The element stiffness matrix, "**KM**", is formed with respect to the positive definition of global forces and displacements as shown below.

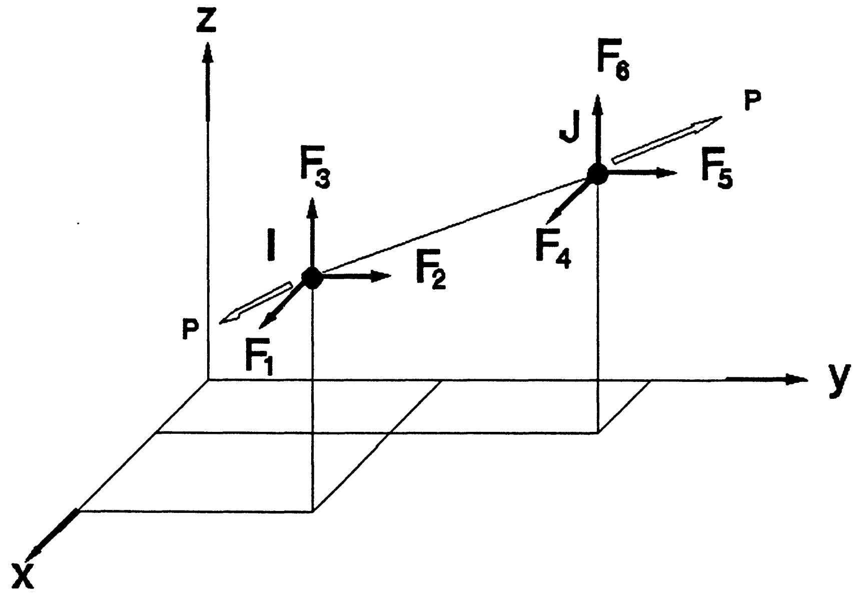


Figure 4.5 POSITIVE DEFINITION OF TRUSS FORCES AND DISPLACEMENTS

The member axial force can be evaluated by the use of the MEMFRC operation which multiplies the matrix "**TM**" by the joint displacements. A positive axial force indicates tension.

FRAME3 KM TM II=I_{33},I_{22} A=? JJ=? E=? G=? N=I,J P=P_1,P_2

The **FRAME3** command forms the 12 x 12 element stiffness matrix named "**KM**" and an 8 x 12 force-displacement matrix named "**TM**" for a general three-dimensional member with axial, bending and torsional deformations included in the formulation. The properties of the member are given as

I_{33}	= the Moment of Inertia about the 3-axis
I_{22}	= the Moment of Inertia about the 2-axis
JJ	= the Torsional Moment of Inertia about the 1-axis
A	= the Axial Area of the member
G	= the Shear Modulus, and
E	= the Modulus of Elasticity of the member.

The coordinates of joint number **I**, **J**, $\mathbf{P_1}$ and $\mathbf{P_2}$ must have been previously loaded in an array named "**XYZ**".

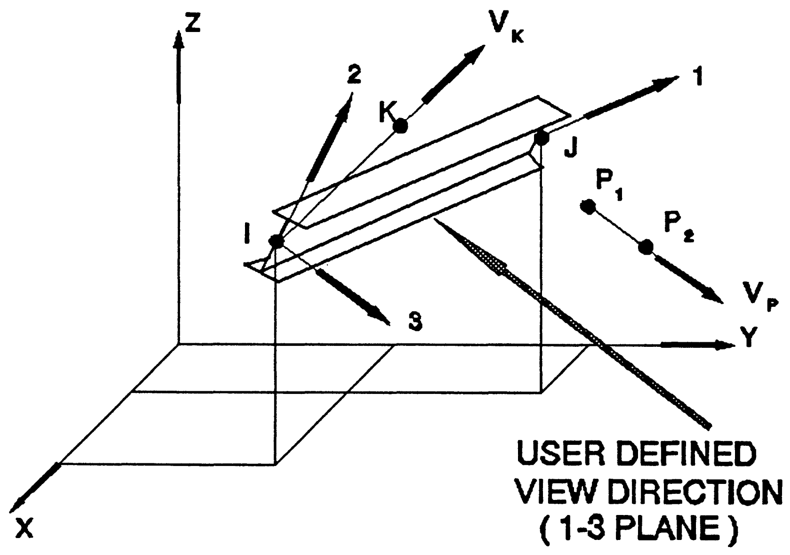


Figure 4.6 TYPICAL THREE DIMENSIONAL FRAME ELEMENT

The element stiffness matrix, "**KM**", is formed with respect to the positive definition of global forces and displacements as shown below.

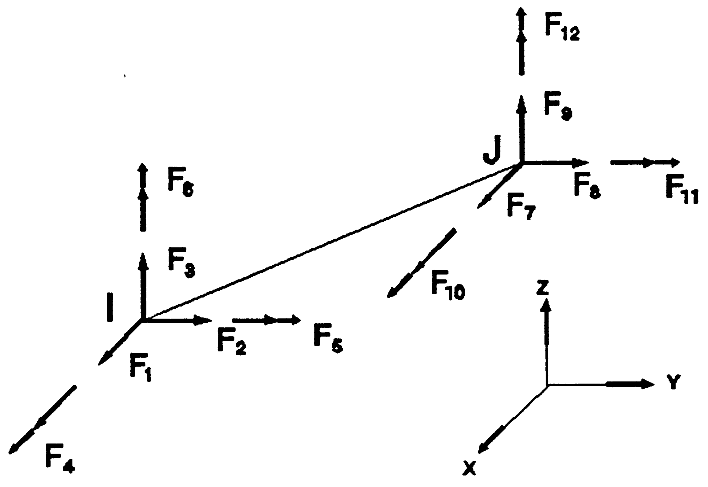


Figure 4.6 POSITIVE DEFINITION OF FORCES AND DISPLACEMENTS

The member forces, with respect to the member's local coordinate system, can be evaluated by the use of the **MEMFRC** operation which multiplies the matrix "**TM**" by the joint displacements. The positive definition of the member forces in the local coordinate system is shown below. The **MEMFRC** command will evaluate the local member forces in the order P_1 - - P_8.

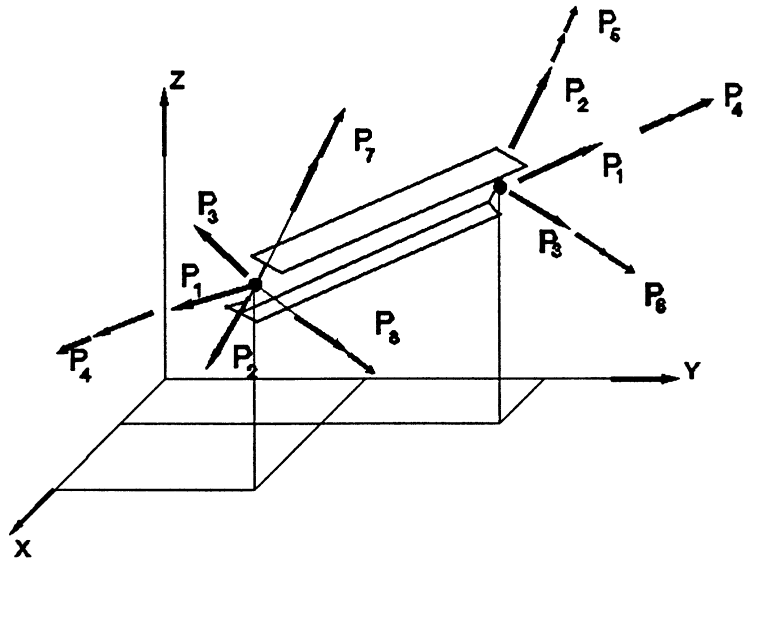


Figure 4.8 POSITIVE FORCES IN LOCAL 1-2-3 SYSTEM

The section properties I_{22} and I_{33} of a three-dimensional frame member must be specified with respect to a 1-2-3 local member coordinate system. In addition, member forces, which are produced by the computer programs, are defined in reference to this local right-hand coordinate system. Therefore, it is the user's responsibility to define the member 1-2-3 system in reference to the global x-y-z system.

The positive 1-axis, V_1 vector, is defined by a line along the axis of the member from node point "I" to node point "J".

The 2 and 3-axes can be specified, with the $P=P_1,P_2$ option, by any one of the following three methods:

METHOD 1 - GLOBAL PLANES ONLY - P=?,0

xy plane	P=1,0	3-axis is the Z-axis	and	$V_2 = V_3 \times V_1$
zx plane	P=2,0	3-axis is the Y-axis	and	$V_2 = V_3 \times V_1$
yz plane	P=3,0	3-axis is the X-axis	and	$V_2 = V_3 \times V_1$

METHOD 2 - SPECIFICATION OF "V_p" VECTOR - $P=P_1,P_2$

The coordinates of node numbers P_1 and P_2 are specified by the user in the joint coordinate information. The vector V_p is normal to the 2-axis and is defined by the line from node point P_1 to node point P_2. The 2 and 3-axes are then calculated as follows:

$$V_2 = V_p \times V_1 \qquad \text{and} \qquad V_3 = V_1 \times V_2$$

Node points P^1 and P_2 may be "dummy nodes" which are not connected to members.

METHOD 3 - SPECIFICATION OF "K" NODE - P=0,K

The V_K vector is defined by the line from node "I" to node "K". The 3 and 2-axes are then calculated as follows:

$$V_3 = V_1 \times V_K \qquad \text{and} \qquad V_2 = V_3 \times V_1$$

4.6 TWO DIMENSIONAL FRAME EXAMPLE

The direct stiffness method is very simple, compared to the slope-deflection method of analysis, if axial deformations in two dimensional frame members are included. Three displacement degrees of freedom (one rotation, one horizontal displacements and one vertical displacement) exist at all rigid joints. Also, all members have six global forces and six global displacements has shown in Figure 4.3 and there is no difference between beams and columns. For example, let use consider the frame shown below.

MEMBER PROPERTIES

Members 1 to 4
$I = 1000 \text{ in}^4$
$A = 20 \text{ in}^2$
$E = 30{,}000$ ksi

Members 5 and 6
$I = 2000 \text{ in}^4$
$A = 30 \text{ in}^2$
$E = 30{,}000$ ksi

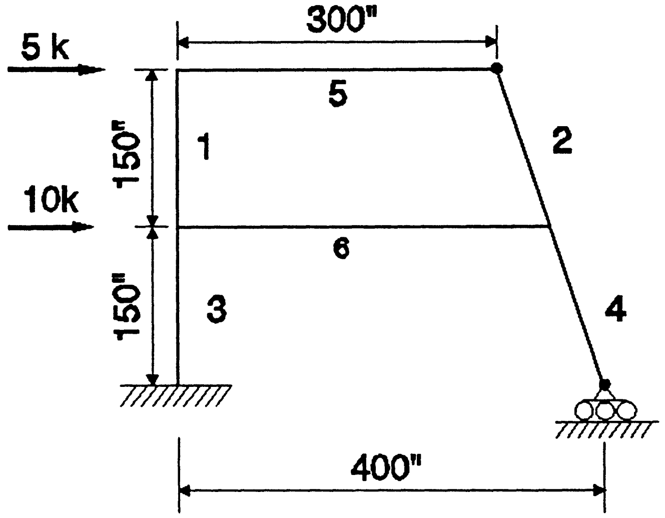


Figure 4.9 Two Dimensional Frame Example

The pin at the right end of the top beam indicates that the ends of both members 2 and 5 can rotate independent; therefore, two rotations must exist at this point. There are 15 degrees of freedom for this structure which are shown in Figure 4.10. The user is responsible for numbering these displacements. The numerical ordering of the displacements is completely arbitrary.

The summary of the member force contributions to the equilibrium equations, for the numbering system shown, is given in Table 4.4. Note that the user must designate what end of member is joint "I" and what end is joint "J"; since, the first three forces of the element stiffness matrix are associated with joint "I" (see Figure 4.3). The student should solve this problem with different numbering systems in order to illustrate that the equilibrium equations can be written in any order.

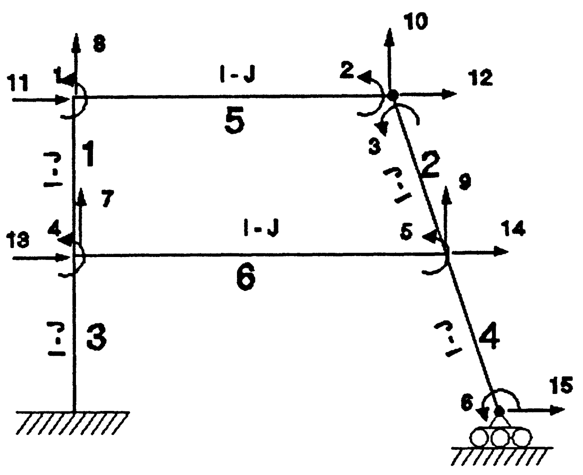


Figure 4.10 User Define Displacement Degrees of Freedom

The CAL input data file for this example, as shown in Table 4.5, is named **FRAMEX** and is on the CAL program. The student should insert additional print statements into the file to verify and understand the direct stiffness commands. Also, the order of some of the CAL commands and the names of the arrays are arbitrary. It is the responsibility of the student to completely understand the procedure and not blindly copy the same approach which is given in the example.

Table 4.4 Contribution of Member Forces to Equilibrium Equations

Member FORCE	MEMBER NUMBER					
	1	2	3	4	5	6
F_1	13	14	0	15	11	13
F_2	7	9	0	0	8	7
F_3	4	5	0	6	1	4
F_4	11	12	13	14	12	14
F_5	8	10	7	9	10	9
F_6	1	3	4	5	2	5

Figure 4.5 CAL INPUT DATA FILE - 2D EXAMPLE FRAME

```
C EXAMPLE OF TWO-D FRAME ELEMENT
C ENTER LOCATION ARRAY
LOADI ID R=6 C=6
13  14   0  15  11  13
 7   9   0   0   8   7
 4   5   0   6   1   4
11  12  13  14  12  14
 8  10   7   9  10   9
 1   3   4   5   2   5
C FORM STIFFNESS MATRIX
ZERO K R=15 C=15
FRAME K1 T1  X=0,0     Y=150,300  I=1000  A=20  E=30000
FRAME K2 T2  X=350,300 Y=150,300
FRAME K3 T3  X=0,0     Y=0,150
FRAME K4 T4  X=400,350 Y=0,150
FRAME K5 T5  X=0,300   Y=300,300
FRAME K6 T6  X=0,350   Y=150,150
ADDK K K1 ID  N=1
ADDK K K2 ID  N=2
ADDK K K3 ID  N=3
ADDK K K4 ID  N=4
ADDK K K5 ID  N=5
ADDK K K6 ID  N=6
C ENTER JOINT LOADS
LOAD RT R=1  C=15
0 0 0 0 0 0 0 0 0 0 5 0 10 0 0
PRINT RT
TRAN RT R
C SOLVE FOR DISPLACEMENTS
DUP R U
SOLVE K U
P U
C EVALUATE MEMBER FORCES
MEMFRC T1 U ID F1 N=1
MEMFRC T2 U ID F2 N=2
MEMFRC T3 U ID F3 N=3
MEMFRC T4 U ID F4 N=4
MEMFRC T5 U ID F5 N=5
MEMFRC T6 U ID F6 N=6
P F1
P F2
P F3
P F4
P F5
P F6
RETURN
```

5. COMMANDS FOR DYNAMIC ANALYSIS

In this section several commands are presented which allow CAL to perform linear dynamic analysis of small structural systems. With the aid of other CAL commands it is possible to solve the following types of dynamic problems:

A. Evaluation of free-vibration mode shapes and frequencies.

B. Automatic generation of Ritz vectors to be used in a mode superposition analysis or response spectra analysis.

C. Mode superposition analysis due to arbitrary loading.

D. Step-by-step analysis of structural systems with arbitrary viscous damping.

E. Dynamic analysis in the frequency domain.

All commands assume that the mass and stiffness matrices have been calculated by other CAL commands. The PLOT command can be used to produce printer plots of results.

5.1 EVALUATION OF MODE SHAPES AND FREQUENCIES

EIGEN K- V+ M-

This command solves the following eigenvalue problem for the mode shapes and frequencies:

$$\mathbf{KV = MVe}$$

Where "K" is the name of the N x N stiffness matrix K. The command is restricted to a diagonal mass matrix; therefore, the array named "M" must be given as a row or column array of the diagonal terms of the N x N mass matrix M.

The N x N matrix V, which contains all the eigenvectors (mode shapes) stored columnwise, is named "V" and is normalized in order that $\mathbf{V^T\,M\,V = I}$.

The N x N matrix **e** is a diagonal matrix of eigenvalues w^2 (frequencies w_i are in radians per sec.^{2}). The EIGEN command stores the eigenvalues e_i in place of the mass terms M_i in the array named "M".

The program uses the standard Jacobi method; therefore, both K and M must be symmetric and positive definite matrices.

JACOBI K- V+ M- E+

This command solves the following eigenvalue problem for the mode shapes and frequencies:

$$\mathbf{KV = MVe}$$

Where "K" is the name of the N x N stiffness matrix K and "M" is the name of the N x N mass matrix. The N x N matrix V, which contains all eigenvectors (mode shapes) stored columnwise, is named "V" and is normalized in order that $\mathbf{V_T\,M\,V = I}$.

The N x N matrix **e** is a diagonal matrix of eigenvalues $w_i{}^2$ (frequencies w_i are in radians per sec.^{2}). The JACOBI command stores the eigenvalues e_i as a N x 1 column matrix named "E". The program uses a modified Jacobi method where both K and M must be symmetric and positive definite matrices.

SQREL M1

The SQREL command replaces each term in matrix M1 with the square root of the term.

INVEL M1

The INVEL command replaces each term in matrix M1 with the inverse of the term.

RITZ K- M F- V+ NV=? S=?

Given a N x N stiffness matrix named "K", a N x 1 mass matrix named "M" and a N x 1 force vector named "F", a N x NV matrix of orthogonal vectors, V, named "V" is generated using a LOAD DEPENDENT algorithm. The matrix V is normalized in order that $\mathbf{V^T M V = I}$.

The generated vectors V are not orthogonal with respect to the stiffness matrix K. If "S" is a nonzero number the static vector response is not included in the response.

NORM M1 M2+ T=?

A row matrix M2 is formed in which each column contains the sum of the corresponding column of the matrix M1. If "T" is not equal to zero the square root of the sum of the square is calculated.

PROD M1 D

This command forms a 1 x 2 array named "D" which contains the product of all terms in the array named "M1". The product is stored as two numbers of the form $D(1)\ 10^{D(2)}$. This command is used to evaluate the determinate of a matrix.

5.2 DYNAMIC RESPONSE BY MODE SUPERPOSITION

DYNAM W C F G(t) X(t)+ DT=? N=?

This command evaluates a set of "I" uncoupled second order differential equations which are generated in the mode superposition analysis of a structural system. The typical equation is of the following form:

$$X_i + 2\, c_i\, w_i\, X_i + w^2\, X_i = f_i\, g(t)\ ; \qquad i = 1,\text{---}I$$

Where

W is a row or column array of the frequencies w_i in radians per second.

C is the name of a row or column array of the damping ratios c_i.

F is the name of I x 1 column array of the terms f_i.

G is the name of a 2 x M array which can be used to define the time function g(t).
X(t) is the name of the I x N array where the results are stored.

DT is the time increment for which the results are produced.

The array G defines a time function in terms of straight line segments where G(1,J) defined the time t_j and G(2,J) is the value $g(t_j)$. The time function must be defined in the range T = 0 to T_{max}, where T_{max} = N x DT. Therefore, the maximum value of G(1,M) must be greater than Tmax.

The accuracy of the solution is not a function of the output time increment "DT" since the command produces the exact solution for straight line segments.

MAX X(t) Xmax+

The MAX command locates the maximum absolute value in each row of the array named X(t) and stores the results in a column matrix Xmax. The maximum value and its column number is also printed or displayed.

PLOT M1 N=? R=R1,R2 - - RN S=S1,S2 - - RN

The PLOT command will produce a printer plot of "N" rows of matrix M1. Where "Ri" is the row number to be plotted and "Si" is the symbol used.

FUNCT G F(t)+ N=? DT=?

The FUNCTion command forms a 1 x N array named F(t). The terms are extracted at "DT" intervals from the time function defined in the array named G. The array G defines a time function in terms of straight line segments where G(1,J) defines the time t_j and G(2,J) is the value $g(t_j)$. The time function must be defined in the range T = 0 to Tmax, where Tmax = N x DT. Therefore, the maximum value of G(1,M) must be greater than Tmax.

5.3 DYNAMIC RESPONSE DY DIRECT STEP-BY-STEP INTEGRATION

STEP K- M C UVA- U+ P F(t) DT=? L=Li,Lmax P=delta,alpha,theta

This command evaluates the displacements U, at equal time steps, of a structural system where the dynamic equilibrium equations are specified in the following form:

Ma(t) + Cv(t) + Ku(t) = PF(t)

Where a(t), v(t) and u(t) are the time-dependent acceleration, velocity and displacement vectors respectively. K, M and C are the names of the N x N stiffness matrix, {K}, mass matrix, M, and damping matrix, C respectively.

The loads are specified as the product of a N x 1 matrix P and a 1 x J array {F(t)} named F(t). The loads F(i) are given at equal time steps as specified by "DT".

UVA is the name of a N x 3 array of initial conditions in which

The first column is a vector of initial displacements U(0)

The second column is a vector of initial velocities V(0)

The third column is a vector of initial accelerations A(0)

After STEP is executed this array will contain the displacements, velocities and accelerations at the last time step.

U is the name of the displacements which are stored as an N x Lmax array. The step-by-step integration is conducted with a time increment "DT"; however, the displacements are stored at Li time steps (or at "Li x DT" time intervals). Therefore, the number of loads specified "J" must be greater than " Li x Lmax ".

The Newmark-Wilson step-by-step integration method is used where the parameters are specified by delta,alpha and theta. The following table lists possible values:

	delta	alpha	theta
Newmark's Average Acceleration	1/2	1/4	1.00
Newmark's Linear Acceleration	1/2	1/6	1.00
Theta Method - Low Damping	1/2	1/6	1.42
Theta Method - High Damping	1/2	1/6	2.00

If the P parameters are not specified the linear acceleration method is used.

5.4 DYNAMIC ANALYSIS IN THE FREQUENCY DOMAIN

The CAL operations DFT, IDFT and FSOLVE are provided in order to solve linear dynamic analysis problems in the frequency domain. This approach can be very effective for problems in which the loading is periodic over a very large time span such as machine vibrations or wind and wave loading on structures. It can be used for other types of loading, (i.e. earthquake ground motions), if the period of the loading is selected to be sufficiently long to assure that the response of the structure at the end of each loading period is essentially zero. If the damping of the system is small a very large period may be required if accurate results are to be obtained for loading which is not basically periodic. Following is a summary of these basic CAL operations:

DFT F(m,t)- DT=?

The M x N array named F(m,t) contains M different time functions. Each row in the array contains values of the function at equal time intervals DT. The time functions F(m,t) represent a time span of $-\infty$ to $+\infty$; however, only the values within a typical period (N DT) are specified as shown below:

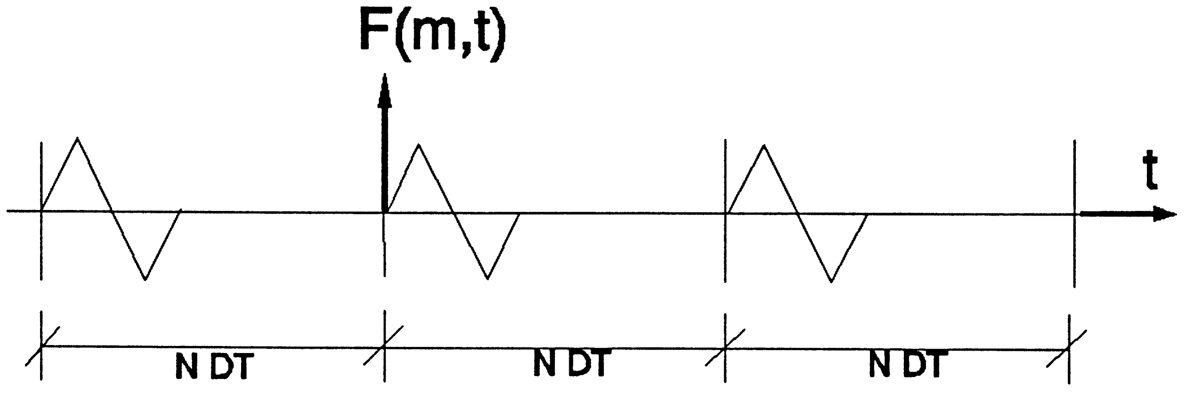


Figure 5.1 TYPICAL TIME FUNCTION "m"

The term F(m,1) represents the value of the function "m" at the beginning and end of the basic time period N x DT. The DFT operation expands the time functions in a series of the following form:

$$F(t) = F_0 + \sum Fc_k Cos(k\ d\omega) + \sum Fs_k Sin(k\ d\omega)$$

where k=1,2,----- (N-1)/2 (for N odd), or, (N-2)/2 (for N even), and dw = 2π/(N DT). The calculated constants for time function "m" are stored in the mth row in the order F_0, Fc_1, Fs_1, Fc_1 - - and replace the original terms in the F(m,t) array. For N even the Nth column will be zero.

IDFT F(m,w)-

This operation transforms the frequency domain functions back to the time domain. The M x N array named F(m,w), which is in the form generated by the DFT or FSOLVE operations, is replaced by time function values at equal time intervals.

RADIUS F(m,w) R(m,w)+

This command operates on the M x N F(m,w) array and creates a M x L R(m,w) array, where L = (N - 1) / 2. The terms are calculated from the following equations:

$$R(m,i) = \sqrt{F(m,2i)^2 + F(m,2i+1)^2}$$

FSOLVE W C F P(w) Y(m,w)+ DT=?

This operation evaluates the solution of a set of uncoupled second order differential equations which are generated in the mode superposition analysis of a structural system for which the loading has been transformed to the frequency domain. W, C, and F are M x 1 arrays and have the same definition as given by the DYNAM operation. The 1 x N array named P(w) is in the same form as produced by the DFT operation and DT is the time step which was used to transform the time domain to the frequency domain.

The mth row in the M x N array named Y(m,w) contains the terms Y_0, Yc_1, Ys_1, yc_2 ----- which is the solution of the mth mode written in the following form:

$$Y(\omega) = Y_0 + \sum Yc_k \, Cos(kd\omega) + \sum Ys_k \, Sin(kd\omega)$$

The frequency domain solution Y(m,w) can be transformed to the modal time domain by the IDFT operation - IDFT Y(m,t) .

6. NUMERICAL METHODS FOR DYNAMIC ANALYSIS

Reprint: Numerical Methods For Dynamic Analysis, by E. L. Wilson, Proceedings, International Conference on Numerical Methods for Offshore Structures, Swansea, 1977.

6.1 INTRODUCTION

The value of the results of a dynamic analysis depends on the approximations involved in the establishment of mathematical models for the structure and foundation and in the selection of the various dynamic load conditions. In general the establishment of the models and the interpretation of the results are the most critical phases of a dynamic analysis. This assumes that the particular method of dynamic analysis used does not introduce additional errors in the solution of the model for the specified loads. It is important that the computer cost for any one analysis is not large in order that inexpensive re-analysis is possible and the results of the analysis can be used by the engineer to influence the basic design of the structure. Also an economical computer analysis will allow some of the basic assumptions used in selecting the model and loads to be varied and the sensitivity of the results evaluated.

The effectiveness of a numerical method for dynamic analysis depends primarily on two factors—the minimization of computer storage and the minimization of computer execution time. The purpose of this paper is to present a summary of various numerical methods which are used in the dynamic analysis of offshore structures and to comment with respect to their computer implementation and efficiency.

It should be noted that all structures, regardless of their simplicity, have an infinite number of degrees of freedom when subjected to dynamic loading. One of the main objectives of selecting a mathematical model is to reduce the infinite degree of freedom system to a model with a limited degree of freedom which will capture the significant physical behaviour of the system. Therefore a considerable insight into the expected dynamic behaviour of the system must be present if a realistic mathematical model is to be established. The model must be capable of representing both the significant wave propagation and structural vibration.

The force equilibrium of an offshore platform modelled as a system with a finite number of degrees of freedom may be written as

$$F_i + F_d + F_s = F \tag{6.1}$$

in which all forces are a function of time and defined as

F_i = The inertia force
F_d = The internal and external damping forces
F_s = The forces carried by the structural members
F = The external applied forces

Equation (6.1) holds for both linear and non-linear systems. However the appropriate numerical method for solution will depend on the degree of non-linearity which is present and if linearization is possible.

6.2 LINEAR DYNAMIC ANALYSIS

For linear systems with viscous damping Equation (6.1) can be written in the form

$$M\ddot{U}(t)+C\dot{U}(t)+KU(t)=F(t) \tag{6.2}$$

in which M is the mass matrix (lumped or consistent), C is the damping matrix (normally not given in the form) and K is the stiffness matrix for the system of structural elements. The time dependent vectors $U(t)$, $\dot{U}(t)$ and $\ddot{U}(t)$ are the displacements, velocities and accelerations respectively.

The time dependent external forces $F(t)$ may be due to wave or seismic forces. Normally the calculation of wave forces is a complicated procedure which depends on the shape of the structural elements. Other papers in these proceedings are referred to for the evaluation of wave forces. In the case of earthquake motion in three-dimensions the loading is of the form

$$F(t)=-M_x\ddot{U}_{xg}-M_y\ddot{U}_{yg}-M_z\ddot{U}_{zg} \tag{6.3}$$

where $\ddot{U}_{ig}$ is the ground acceleration in direction i and M_i is a column matrix which represents the sum of all columns in the mass matrix M associated with displacements in the i-direction.[1] This definition of the seismic loading is valid only if the vector is defined as the displacement relative to the displacement at the base of the structure.

In many cases wave and seismic loads in each direction are specified in terms of a spectrum of maximum response values. In the case of three-dimensional behaviour, however, great care must be taken in the application of the technique, since the basic input in the various directions must be statistically independent if the results are to be combined in a probabilistic manner.

Figure 6.1 indicates the possible solution techniques which can be employed for the dynamic analysis of linear systems. Four different solution approaches are possible. Two methods involve the evaluation of the undamped mode shapes and frequencies as the first step in the analysis—the mode superposition method and the spectrum analysis method. The frequency domain method

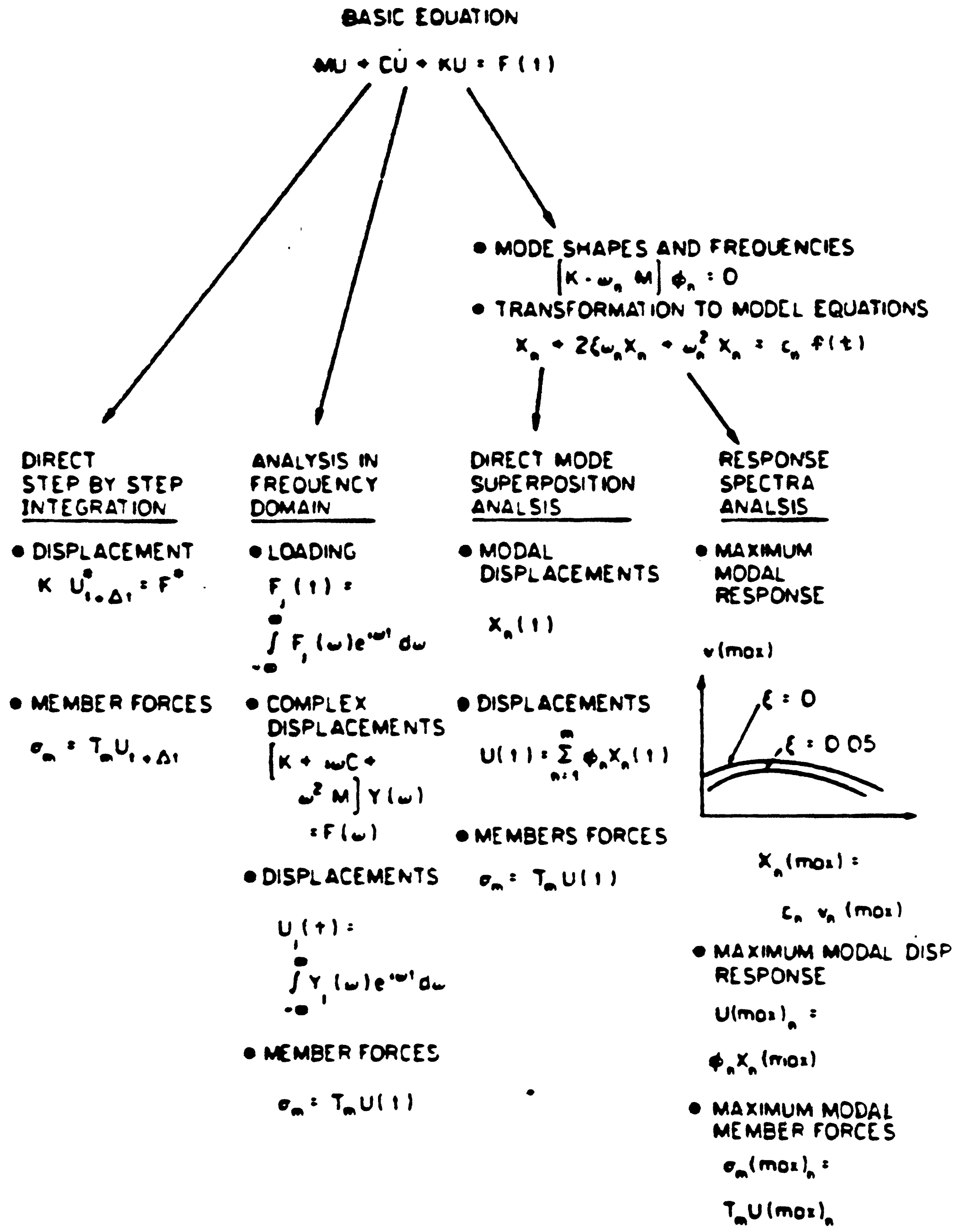

BASIC EQUATION
MÜ + CU̇ + KU = F(t)
• MODE SHAPES AND FREQUENCIES
[K - ω_n M] φ_n = 0
• TRANSFORMATION TO MODEL EQUATIONS
Ẍ_n + 2ξω_n Ẋ_n + ω_n² X_n = c_n f(t)
DIRECT STEP BY STEP INTEGRATION
• DISPLACEMENT
K U*_{t+Δt} = F*
• MEMBER FORCES
σ_m = T_m U_{t+Δt}
ANALYSIS IN FREQUENCY DOMAIN
• LOADING
F_i(t) = ∫_{-∞}^{∞} F_i(ω) e^{iωt} dω
• COMPLEX DISPLACEMENTS
[K + iωC + ω² M] Y(ω) = F(ω)
• DISPLACEMENTS
U_i(t) = ∫_{-∞}^{∞} Y_i(ω) e^{iωt} dω
• MEMBER FORCES
σ_m = T_m U(t)
DIRECT MODE SUPERPOSITION ANALSIS
• MODAL DISPLACEMENTS
X_n(t)
• DISPLACEMENTS
U(t) = Σ_{n=1}^{m} φ_n X_n(t)
• MEMBERS FORCES
σ_m = T_m U(t)
RESPONSE SPECTRA ANALSIS
• MAXIMUM MODAL RESPONSE
v(max)
ξ = 0
ξ = 0.05
X_n(max) = c_n v_n(max)
• MAXIMUM MODAL DISP RESPONSE
U(max)_n = φ_n X_n(max)
• MAXIMUM MODAL MEMBER FORCES
σ_m(max)_n = T_m U(max)_n

Figure 6.1 Methods for linear dynamic analysis

involves the expansion of the load in a Fourier Integral which reduces the dynamic analysis into a series of solutions of linear sets of complex equations. Another method which can be very efficient for certain systems is the direct step-by-step integration of the equations of motion.

Each phase of the possible solution methods suggested in Figure 6.1 involves a numerical method which must be formulated in effective form for computer implementation. Solution of linear equations, evaluation of eigenvalues and eigenvectors, numerical evaluation of integrals, transformation to the frequency domain, evaluation by the fast Fourier transform and the step-by-step numerical integration of the coupled equations of motion will be discussed in detail in the following sections.

6.2.1 Solution of linear equations

The solution in the frequency domain, the evaluation of eigenvectors and step-by-step solution methods can involve the solution of a set of linear equations; therefore an efficient solution method for this phase can be very worthwhile. The set of equations to be solved can be written symbolically as

$$AX = b \tag{6.4}$$

where A is an $N \times N$ symmetrical matrix and X is a vector of unknowns corresponding with the specified vector b. One of the most important aspects in the computer solution of a large set of equations is the method used to store the terms in the A matrix. One method which has been found to be very effective is the active column storage technique in which only the nonzero terms in the reduced matrix are stored. If a basic elimination method is used the only storage required will be from the first non-zero terms in each column down to the diagonal term in that column. Therefore the matrix with terms initially located as indicated in Figure 6.2(a) can be stored as a one-dimensional array as shown in Figure 6.2(b) along with an integer pointer array indicating the location of each diagonal term. Figure 6.2(c) indicates how the storage technique is extended to approximately equal size blocks which can be stored on low speed storage. The solution technique requires two blocks in high speed core storage at any particular time. It has been demonstrated that most of the popular methods for the solution of equations are very similar, with respect to the number of numerical operations, and they can be considered to be variations of the Gauss elimination method.[2]

The basic factorization algorithm for the solution of equation stored in active column form may be summarized by the following three steps:

(*a*) *Triangularization of A* ($j = 2$ *to* N)

$$A = LU \quad \text{or} \quad A = LDL^T \tag{6.5}$$

in which the jth column of upper triangular matrix U is evaluated from

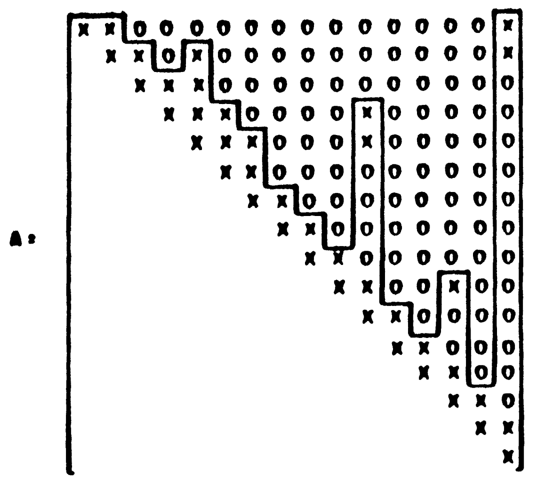
$$U_{ij} = A_{ij} - \sum_{k=km}^{j-1} L_{ik} U_{kj} \qquad i = fj \text{ to } j \tag{6.6}$$

and the jth row of lower triangular matrix is given by

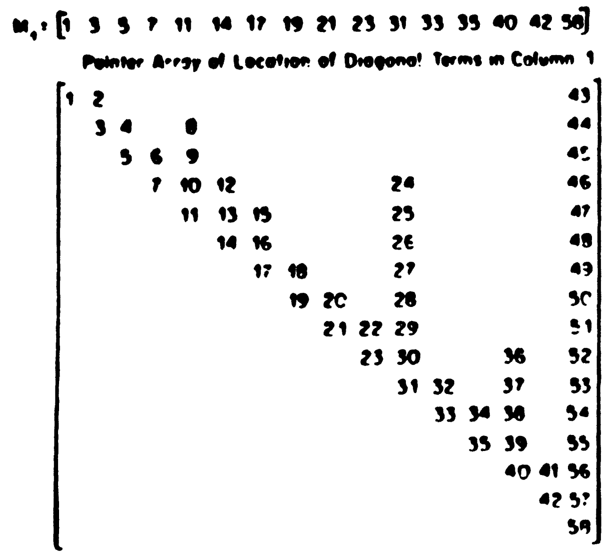
$$L_{ji} = U_{ij} / D_{ii} \qquad i = fj \text{ to } j-1 \tag{6.7}$$

where D is a diagonal matrix, fj is the first non-zero term in column j and fi is

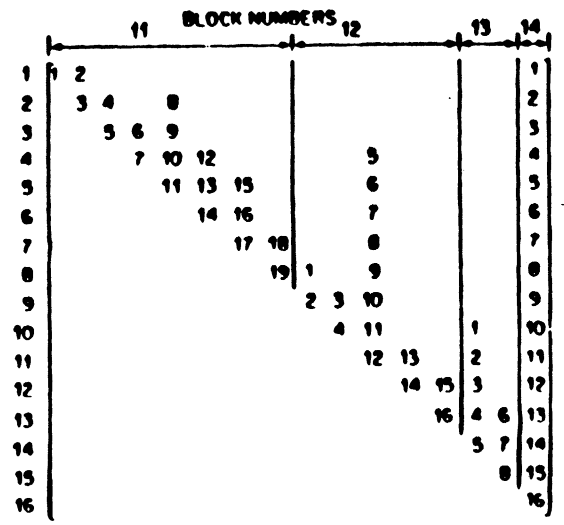
$$AX = b \qquad A = LU = LDL^T$$



(a) Elements in Original Matrix


M_1 = [1 3 5 7 11 14 17 19 21 23 31 33 35 40 42 58]
Pointer Array of Location of Diagonal Terms in Column 1

(b) Storage Sequence for in core Active Column Equation Solver



K(1)	K(2)	K(3)	K(4)	M(J)							
1	1	1	8	1	3	5	7	11	14	17	19
2	1	9	13	2	4	12	14	16			
3	2	14	15	5	9						
4	1	16	16	16							

(c) Example of Matrix Stored in a Maximum of 20 Blocks

Figure 6.2

the first non-zero term in column i. The symbol km represents maximum of fj and fi.

The diagonal terms U_{ii} and D_{ii} are identical since L_{ii} is normalized as 1·0. It is most convenient to evaluate U_{ij} within a computer program column-wise with $i = fj$ to j. After each column is complete L_{ji} is evaluated row-wise with $i = fj$ to $j-1$ and stored in transposed form as L^*_{ij} where U_{ij} was previously located. The diagonal term U_{ii} or D_{ii} remain at the same location as A_{ii}.

(*b*) *Forward reduction*
Equation (6.4) can be written as $Lz = b$, where $z = DL^T$. Therefore

$$z_i = b_i - \sum_{k=fi}^{i-1} L^*_{ki} z_k \qquad i = 1 \text{ to } N \tag{6.8}$$

If y is defined as $y = D^{-1}z$ then $y = L^T x$; or

$$y_i = z_i / D_{ii} \qquad i = 1 \text{ to } N \tag{6.9}$$

(*c*) *Backsubstitution*
From $y = L^T x$

$$x_i = y_i - \sum_{k=i+1}^{N} L^*_{ik} y_k \qquad i = N \text{ to } 1 \tag{6.10}$$

It is important to note that all zero operations are skipped by this technique and that neither the number of operations or the the required storage is a function of the band width. Also the triangularization operation on the A matrix is independent of the forward or backsubstitution operations; therefore this operation need be done only once: The forward and backsubstitution phases for each load condition normally involve a small number of operations compared to triangularization. Recognition of this can greatly minimize the numerical effort required in some eigenvalue methods and in the direct step-by-step integration of the equations of motion.

6.2.2 Step-by-step integration

The direct integration of the linear dynamic equations of motion is a simple approach which can have considerable advantages for some problems. The basic equation is satisfied at discrete points in time, 0, Δt, $2\,\Delta t$, $3\,\Delta t, \ldots t$, $t + \Delta t, \ldots T$. The solution starts from a point in time where the displacements, velocities and accelerations are known. Based on an assumption on the behaviour of the system during the next small increment of time the displacements, velocities and accelerations at the next point in time can be evaluated. Many different methods have been developed for this purpose.[2-12] However only two techniques will be summarized in this paper.

(a) The central difference method

In the case of a diagonal mass and damping matrix and for systems where the shortest period is not too small the central difference method has proven to be most effective. At time t the equation to be satisfied is

$$M\ddot{U}_t + C\dot{U}_t + KU_t = F_t \tag{6.11}$$

The following standard finite difference relationships are used:

$$\ddot{U}_t = \frac{1}{\Delta t^2}(U_{t-\Delta t} - 2U_t + U_{t+\Delta t}) \tag{6.12}$$

$$\dot{U}_t = \frac{1}{2\,\Delta t}(U_{t+\Delta t} - U_{t-\Delta t}) \tag{6.13}$$

Equations (6.12) and (6.13) can be substituted into Equation (6.11) to form a set of linear equations of the form

$$M^* U_{t+\Delta t} = F^* \tag{6.14}$$

where

$$M^* = \frac{1}{\Delta t^2}M + \frac{1}{2\,\Delta t}C \tag{6.15}$$

$$F^* = F_t - KU_t + \frac{1}{\Delta t^2}M(2U_t - U_{t-\Delta t}) + \frac{1}{2\,\Delta t}CU_{t-\Delta t} \tag{6.16}$$

If M and C are diagonal one notes that the solution of Equation (6.14) is trivial; also computer storage will be minimized. Another very important technique which can be used to further minimize the number of numerical operations and computer storage requirements is not to form the complete stiffness K. The structural forces KU_t can be evaluated element by element, or

$$F_s = KU_t = \sum_m K_m U_t$$

in which K_m is the stiffness matrix for element m. If elements have identical stiffness matrices a further reduction in computer storage can be realized.

The solution $U_{t+\Delta t}$ is based on using the equilibrium at time t; therefore this approach is called an 'explicit integration method'. One of the most significant disadvantages of the central difference method is that it is only conditionally stable.[2] In order for the method to produce finite results the time step Δt must be less than T_n/π where T_n is the shortest period in the discrete model. For many structures this requires time steps so small that the method may be impractical.

(b) The Newmark–Wilson method

One of the most flexible step-by-step integration methods has been presented by Newmark.[8] This method is based on the following expressions for the velocity and displacement at the end of the time interval:

$$\dot{U}_{t+\Delta t} = \dot{U}_t + \Delta t\,(1-\delta)\ddot{U}_t + \Delta t\,\delta\ddot{U}_{t+\Delta t} \qquad (6.17)$$

$$U_{t+\Delta t} = U_t + \Delta t\,\dot{U}_t + \Delta t^2\,(\tfrac{1}{2}-\alpha)\ddot{U}_t + \Delta t^2\,\alpha\ddot{U}_{t+\Delta t} \qquad (6.18)$$

where α and δ are selected to produce the desired accuracy and stability. If $\delta = \frac{1}{2}$ and $\alpha = \frac{1}{6}$ the well known linear acceleration is produced, which is also a conditionally stable method. One of the most widely used methods is the constant-average-acceleration method ($\delta = \frac{1}{2}$ and $\alpha = \frac{1}{4}$) which is an unconditionally stable method without numerical damping.

This method is called an 'implicit integration method' since it satisfies the equilibrium equations of motion at time $t+\Delta t$, or

$$M\ddot{U}_{t+\Delta t} + C\dot{U}_{t+\Delta t} + KU_{t+\Delta t} = F_{t+\Delta t} \qquad (6.19)$$

This equation can be solved by iteration; however Equations (6.17), (6.18) and (6.19) can be combined into a step-by-step algorithm which involves the solution of a set of equations at each time step of the form

$$K^* U_{t+\Delta t} = F^* \qquad (6.20)$$

Since K^* is not a function of time it can be triangularized once at the beginning of the calculations. The computer solution time for this type of algorithm is basically proportional to the number of time steps required.

The Wilson θ method is a technique which can be used to modify the basic Newmark method in order to increase the stability limits and to add numerical damping.[13] The θ method was first applied to the linear acceleration method in order to improve stability and has been used to damp out high frequency oscillations which often develop in linear and non-linear step-by-step integration. The technique involves using the Newmark method to find the solution at $t+\theta\,\Delta t$, then, based on linear acceleration, calculating the results at $t+\Delta t$ for use as initial conditions for the next time step. The Newmark–Wilson algorithm is summarized in Table 6.1. With $\theta = 1$ the approach is the standard Newmark method. An unconditionally stable method with large damping in the higher modes is produced with $\delta = \frac{1}{2}$, $\alpha = \frac{1}{6}$ and $\theta = 1{\cdot}4$.[12]

6.2.3 Frequency domain approach

An alternative to the direct integration of the coupled linear equations of motion is to use a formal mathematical transformation to eliminate the time function from the equations before solution progresses.[1] The basic approach involves the expansion of the time-dependent loads in terms of a series of

Table 6.1 The Newmark-Wilson algorithm for linear step-by-step integration

A. INITIAL CALCULATIONS:

1. Form stiffness matrix K, mass matrix M and damping matrix C.
2. Initialize U_0, $\dot{U}_0$, and $\ddot{U}_0$.
3. Specify algorithm parameters α, δ and θ

$$\delta \geq 0{\cdot}50; \qquad \alpha \geq 0{\cdot}25(0{\cdot}5+\delta)^2; \qquad \theta \geq 1{\cdot}0$$

4. Calculate integration constants:

$$\tau = \theta\,\Delta t \qquad a_3 = \frac{1}{2\alpha} - 1 \qquad a_7 = \Delta t\,\delta$$

$$a_0 = \frac{1}{\alpha\tau^2} \qquad a_4 = \frac{\delta}{\alpha} - 1 \qquad a_8 = \Delta t^2(\tfrac{1}{2} - \alpha)$$

$$a_1 = \frac{\delta}{\alpha\tau} \qquad a_5 = \frac{\tau}{2}(\delta/\alpha - 2) \qquad a_9 = \alpha\,\Delta t^2$$

$$a_2 = \frac{1}{\alpha\tau} \qquad a_6 = \Delta t(1-\delta)$$

5. Form effective stiffness matrix: $K^* = K + a_0 M + a_1 C$
6. Triangularize K^*: $K^* = LDL^T$

B. FOR EACH TIME STEP:

1. Calculate effective load vector at time $t+\tau$:

$$F^* = F_{t+\tau} + M(a_0 U_t + a_2 \dot{U}_t + a_3 \ddot{U}_t)$$
$$+ C(a_1 U_t + a_4 \dot{U}_t + a_5 \ddot{U}_t)$$

2. Solve for displacements at time $t+\tau$:

$$LDL^T U_{t+\tau} = F^*$$

3 Calculate accelerations, velocities and displacement at $t+\Delta t$:

$$\ddot{U}_{t+\tau} = a_0(U_{t+\tau} - U_t) - a_2\dot{U}_t - a_3\ddot{U}_t$$

$$\ddot{U}_{t+\Delta t} = \ddot{U}_t + \frac{1}{\theta}(\ddot{U}_{t+\tau} - \ddot{U}_t)$$

$$\dot{U}_{t+\Delta t} = \dot{U}_t + a_6\ddot{U}_t + a_7\ddot{U}_{t+\Delta t}$$

$$U_{t+\Delta t} = U_t + \Delta t\dot{U}_t + a_8\ddot{U}_t + a_9\ddot{U}_{t+\Delta t}$$

harmonic functions. One can use the standard Fourier Series in which the loads $F(t)$ are expanded in a series of the form

$$F(t) = \sum_{n=0}^{\infty} A_n \cos\frac{n\pi}{d}t + \sum_{n=0}^{\infty} B_n \sin\frac{n\pi}{d}t \tag{6.21}$$

in which d is the duration of the loading. The Fourier Coefficients can be evaluated and exact solutions found for the harmonic functions $A_n \cos(n\pi/d)t$

and $B_n \sin(n\pi/d)t$. It is assumed that the loading can be approximated by a finite number of terms. Therefore the total solution in time is a summation of the exact solution for each harmonic function. For systems without damping this straightforward Fourier Series approach is numerically very effective since the response of an undamped structure to a harmonic sin or cos function loading is also sin or cos displacement solution.

An alternate method of eliminating the time variable from the dynamic equilibrium equations is to express the loads as an infinite integral, or

$$F(t) = \int_0^\infty (A(\omega)\cos\omega t + B(\omega)\sin\omega t)\,d\omega \tag{6.22}$$

The functions $A(\omega)$ and $B(\omega)$ in the Fourier integral are given by

$$A(\omega) = \frac{2}{\pi}\int_0^d F(t)\cos\omega t\,dt \tag{6.23}$$

$$B(\omega) = \frac{2}{\pi}\int_0^d F(t)\sin\omega t\,dt \tag{6.24}$$

Also the Fourier integral can be written in complex form as

$$F(t) = \int_{-\infty}^{\infty} \bar{F}(\omega)\,e^{i\omega t}\,d\omega \tag{6.25}$$

in which

$$\bar{F}(\omega) = \tfrac{1}{2}[A(\omega) - iB(\omega)] \tag{6.26}$$

$$\bar{F}(-\omega) = \tfrac{1}{2}[A(\omega) + iB(\omega)] \tag{6.27}$$

since

$$e^{i\omega t} + e^{-i\omega t} = 2\cos\omega t$$

$$e^{i\omega t} - e^{-i\omega t} = 2i\sin\omega t$$

The general equilibrium equations can now be written as

$$M\ddot{U} + C\dot{U} + KU = \int_{-\infty}^{\infty} \bar{F}(\omega)\,e^{i\omega t}\,d\omega \tag{6.28}$$

The solution is assumed to be of the form

$$U(t) = \int_{-\infty}^{\infty} Y(\omega)\,e^{i\omega t}\,d\omega \tag{6.29}$$

therefore

$$\dot{U}(t) = \int_{-\infty}^{\infty} i\omega Y(\omega)\,e^{i\omega t}\,d\omega \tag{6.30}$$

$$\ddot{U}(t)=\int_{-\infty}^{\infty}-\omega^2 Y(\omega)e^{i\omega t} \tag{6.31}$$

Hence the following complex set of equations must be solved for various values of ω:

$$(K+i\omega C-\omega^2 M)Y(\omega)=\bar{F}(\omega) \tag{6.32}$$

If $\bar{F}(\omega)$ requires a large number of points to define the complete function it is apparent that a large number of solutions of complex equations will be necessary. This large numerical effort can be minimized by solving for the basic eigenvalues of the system and transforming the equations to a smaller system expressed in model coordinants [Equation (6.67)]. The evaluation of the complex loads $\bar{F}(\omega)$ is not a major computational problem as compared to the multiple solution of a large system of complex equations. Furthermore the Fast Fourier transform algorithm can minimize both the evaluation of the Fourier transforms and the recovery of the displacements, Equation (6.29).

The major advantage of the Frequency Domain approach is in its application to substructure analysis. Structure-foundation or structure-fluid systems are considered by the development of the frequency-dependent matrices for the separate systems. Ritz techniques can also be used to effectively reduce the size of the system.

6.2.4 Numerical evaluation of mode shapes and frequencies

For large structural systems one of the most time consuming phases of a dynamic analysis may be the evaluation of eigenvalues and eigenvectors of the $N\times N$ matrix equation

$$M\ddot{U}+\bar{\omega}^2 KU=0 \tag{6.33}$$

The undamped free vibration of the structural model has a solution form of

$$U(t)=\sum_{n=1}^{N}e^{i\bar{\omega}t}\phi_n$$

Therefore the resulting eigenvalue problem must be solved

$$(K-\bar{\omega}_n^2 M)\phi_n=0 \tag{6.34}$$

(a) Static condensation

One technique which is often used to reduce the size of the system before the evaluation of eigenvalues is to eliminate the massless degrees of freedom from the system. For this case Equation (6.34) is rewritten as

$$\begin{bmatrix}K_{aa} & K_{ab}\\ K_{ba} & K_{bb}\end{bmatrix}\begin{bmatrix}\phi_a\\ \phi_b\end{bmatrix}=\bar{\omega}^2\begin{bmatrix}0 & 0\\ 0 & M_b\end{bmatrix}\begin{bmatrix}\phi_a\\ \phi_b\end{bmatrix} \tag{6.35}$$

The first submatrix equation is

$$K_{aa}\phi_a + K_{ab}\phi_b = 0 \tag{6.36}$$

Therefore the massless degrees of freedom are related to the degrees of freedom at the mass points by

$$\phi_a = T\phi_b \tag{6.37}$$

where

$$T = -K_{aa}^{-1}K_{ab} \tag{6.38}$$

The resulting eigenvalue problem is of the form

$$K_{bb}^{*}\phi_b = \bar{\omega}^2 M_b\phi_b \tag{6.39}$$

in which

$$K_{bb}^{*} = K_{bb} + K_{ba}T \tag{6.40}$$

Within a computer program however these submatrix operations are not necessary since it is more efficient to perform the 'static condensation' directly on the massless degrees of freedom, similar to the Gauss elimination procedure,[14] without requiring submatrix storage. One important disadvantage of the static condensation approach is that the matrix K_{bb}^{*} tends to fill as more massless degrees of freedom are eliminated. Therefore the reduction in size of the system may not be economical from a computational viewpoint. In addition if the mass is physically lumped at the time of creating the mathematical model additional errors may be introduced.

(b) Reduction of size of system by Ritz functions

A more general approach to the reduction of the size of the eigenvalue problem [Equation (6.34)] is the application of the Ritz method. This technique is not restricted to a particular mass distribution—a full mass matrix does not increase the computational effort significantly. However for systems with a limited number of masses the method can be identical to the static condensation method.[15]

The method can be very accurate if some physical insight into the behaviour of the structure is known. Static load patterns are selected and corresponding displacement vectors are calculated. Or

$$KR = P \tag{6.41}$$

The true displacements of the system are approximated by a linear combination of the discrete Ritz vectors R. Or the true eigenvectors are approximated by

$$\phi = RX = R_1X_1 + R_2X_2 + R_3X_3 + \cdots R_LX_L \tag{6.42}$$

where L is the number of load patterns and is smaller than the size of the system N. The load patterns can be very simple; however they must be linearly independent. In order to produce the lower frequencies the load pattern should activate the large masses and areas of maximum flexibility. If a single unit load is used as a load pattern the method mathematically lumps the consistent mass of the system at that degree of freedom.

The reduced eigenvalue problem is produced by the substitution of Equation (6.42) into Equation (6.34) and premultiplication by R^T. Or,

$$K^*X - MX\Omega = 0 \tag{6.43}$$

where

$$K^* = R^TKR = R^TP \tag{6.44}$$

$$M^* = R^TMR \tag{6.45}$$

$$\Omega = \text{diag}\,(\omega_i^2) \tag{6.46}$$

Both K^* and M^* are full matrices because the Ritz vectors are not orthogonal. Since all the eigenvalues and eigenvectors are required and the system is relatively small, less than 100, the Jacobi method is one of the most effective for this type of eigenvalue problem.[2]

One advantage of the Ritz method for the reduction of the number of degrees of freedom for a very large structure is that it involves only a solution of a set of linear equations which may have a large number of zero terms in the stiffness matrix, Equation (6.41). Therefore a large number of structural elements can be used to model the basic structural behaviour. The use of Ritz functions can be considered as a formal mathematical method of evaluating an approximate generalized mass matrix for the purpose of dynamic analysis.

Figure 6.3 illustrates the selection of static load patterns for a simple tower type structure. This structure has 6 unconstrained joints or 36 degrees of freedom. It is of interest to point out that for the lateral load pattern shown the resulting displacement is complex and involves movements in all directions in addition to torsional behaviour.

For a structure of this type one would expect the lateral behaviour to be expressed by the first six mode shapes. Because of the geometric arrangement of the members joints 1, 2 and 3 and joints 4, 5 and 6 will act as separate units with very little relative movement between joints. Therefore the six possible load patterns which will capture this fundamental behaviour are easily established and are shown in Figure 6.3a. Of course the first six vibrational mode shapes for this structure will be composed of a linear combination of the resulting displacement patterns. Figure 6.3b illustrates six other load patterns which could have been used to produce identical results.

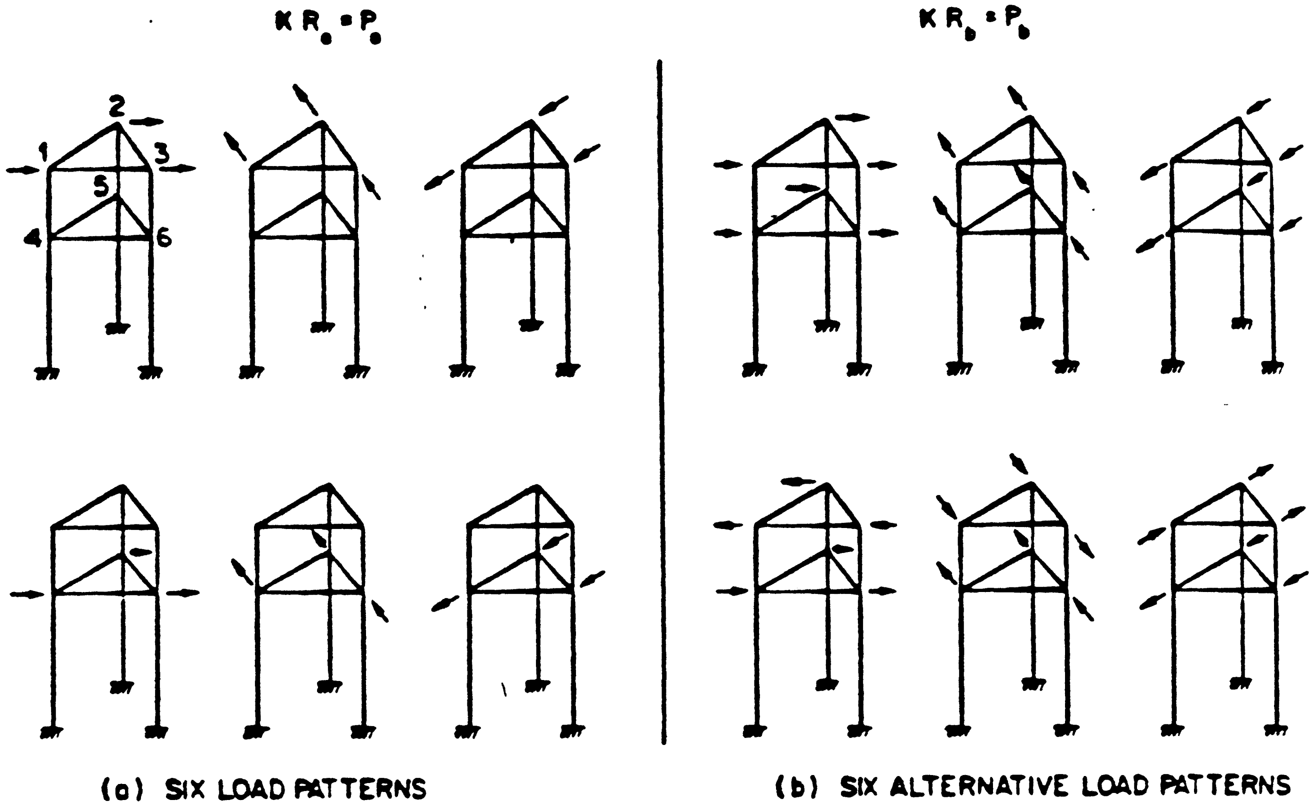


Figure 6.3 Example of selection of different load patterns for tower structure

The physical modes which have been neglected by this approach are the breathing modes between joints 1, 2 and 3 and joints 4, 5 and 6. Also the vertical vibrational modes have been omitted; but axial deformations are included in the six lateral modes.

(*c*) *Subspace iteration*

The subspace iteration method for the determination of eigenvalues and eigenvectors of very large structural systems is a significant extension of the Ritz reduction approach.[16,17,18] It is the only modern computer method for very large systems (over 5000 degrees of freedom) which will converge to the exact eigenvalues.

Table 6.2 summarizes the subspace iteration method. The computer implementation of the method and a FORTRAN listing is given in Reference 2. From Table 6.2 one notes that the initial calculations are identical to the Ritz method in which the first set of load patterns must be specified. After one Ritz solution is found and the first approximation of the first L eigenvalues of the system is calculated as

$$\phi^{(1)} = R_1 X_1 \tag{6.47}$$

an improved set of load patterns can be calculated from

$$P_2 = M\phi^{(1)}$$

Table 6.2 Summary of the subspace iteration algorithm for solution of large eigenvalue problem $K\phi_n = \omega_n^2 M\phi_n$

A. INITIAL CALCULATION:

1. Form stiffness matrix K and mass matrix M.
2. Triangularize K:

$$K = LDL^T \qquad (N \times N)$$

3. Specify initial load patterns:

$$P_1 = N \times L \text{matrix} \quad \text{where} \quad L \ll N$$

B. FOR EACH ITERATION, $k = 1, 2, 3, \ldots$:

1. Solve for Ritz vectors R_k:

$$LDL^T R_k = P_k \qquad (N \times L)$$

2. Calculate generalized stiffness in subspace:

$$K^{(k)} = R_k^T K R_k = R_k^T P_k \qquad (L \times L)$$

3. Calculate generalized mass in subspace:

$$M^{(k)} = R_k^T M R_k \qquad (L \times L)$$

4. Solve eigenvalue problem in subspace:

$$K^{(k)} X_k = M^{(k)} X_k \Omega^{(k)} \qquad (L \times L)$$

5. Calculate improved approximate eigenvectors:

$$\phi^{(k)} = R_k X_k$$

6. Check for convergence:

$$\Omega^{(k)} \rightarrow \text{diag}\,(\omega_n^2) \quad \text{and} \quad \phi^{(k)} \rightarrow \phi \quad \text{as} \quad k \rightarrow \infty$$

stop if converged—perform Sturm sequence check

7. Calculate improved load patterns for next iteration:

$$P_{k+1} = M\phi^{(k)} \qquad (N \times L)$$

8. Return to Step B-1.

It is assumed that P_2 is an improved estimation of the inertia forces associated with the basic mode shapes. From Table 6.2 it is clear that this iteration technique can be carried out to any desired degree of accuracy, assuming the method converges. The convergence of the method for various conditions is given in Reference 2.

Some additional comments associated with the advantages of the subspace iteration method with respect to numerical effort are:

(i) The total stiffness matrix for the system need be triangularized only once.

(ii) Since $K^{(k)}$ and $M^{(k)}$ tend to become diagonal as the iteration progresses the Jacobi method is very effective for this type of eigenvalue problem.

(iii) The size of the subspace L should be approximately 30 per cent more vectors more than the number of accurate eigenvalues required.

(iv) In order to insure participation of all modes one additional random vector should be added to the load pattern set before each iteration.

(v) Since the approach is basically a power method the lowest eigenvalues converge faster and are more accurate.

(vi) For most structures a high degree of accuracy is not required for the highest modes, because of their low participation in the dynamic response. Therefore the subspace iteration produces practical results with respect to required accuracy.

(d) Additional numerical techniques for eigenvalue problems

In a general computer program for the calculation of eigenvalues and eigenvectors several different numerical techniques may be useful in the solution strategy in order to improve the convergence and to minimize numerical effort.

(i) Inverse iteration

If only one load pattern is used in the iteration procedure given in Table 6.2 the subspace iteration approach is the standard inverse iteration method and will converge to the lowest eigenvalue. For this case the eigenvalue problem in the subspace is trivial. Or

$$\lambda_1 = \omega_1^2 \simeq K^{(k)}/M^{(k)} \quad \bullet \tag{6.49}$$

which is better known as the Rayleigh quotient.[6] After the first eigenpair λ_1 and ϕ_1 are determined inverse iteration can be used to calculate accurately additional eigenpairs if the techniques of shifting, determinant search, deflation and Sturm sequence checking are introduced.

(ii) Determinant search

This numerical technique can be explained by considering the following equation:

$$[K - \lambda M]X = 0 \tag{6.50}$$

For any numerical value of λ_k the following triangularization is possible

$$[K - \lambda_k M] = L_k D_k L_k^T$$

The determinant of this matrix for λ_k is

$$\det(K - \lambda_k M) = D_{11} D_{22} D_{33} \cdots D_{NN}$$

where N is the order of the matrix.

If the numerical value of the determinant is evaluated for a large number of different values of λ a function can be generated of the form shown in

Figure 6.4. This function $p(\lambda)$, which in this case is evaluated numerically, is a plot of the characteristic polynomial. $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \ldots \lambda_N$, are the eigenvalues of the system and correspond to zero values of the det $(K-\lambda M)$.

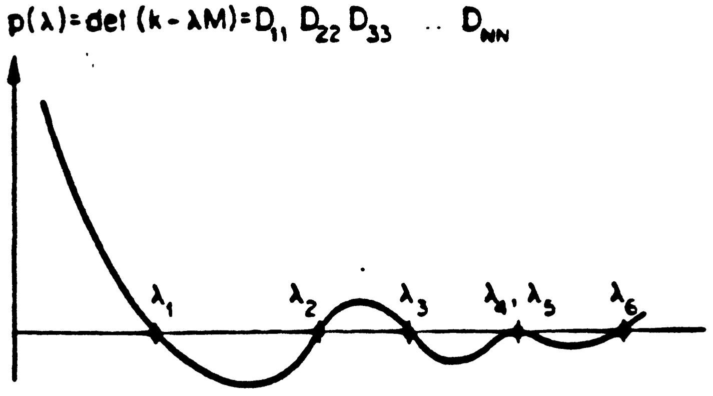


Figure 6.4 Characteristic polynomial $[K-\lambda M]$

(iii) Deflation

A numerical plot of a polynomial with the first root suppressed, or deflated, can be computed and would have the approximate shape as shown in Figure 6.5. The only reason for numerically evaluating $p_1(\mu_k)$ and $p_1(\mu_{k+1})$ is to obtain a better estimation for λ_s from the extrapolation equation

$$\lambda_s = \mu_k + \frac{\mu_k - \mu_{k+1}}{p_1(\mu_{k+1}) - p_1(\mu_k)} p_1(\mu_k) \tag{6.53}$$

It is this type of strategy which is used to obtain a value of λ_s which is close to a desired root. For this case two triangularizations are required in order to evaluate λ_s. Therefore this technique is effective for matrices with small band widths which can be triangularized with a minimum of numerical effort.[19]

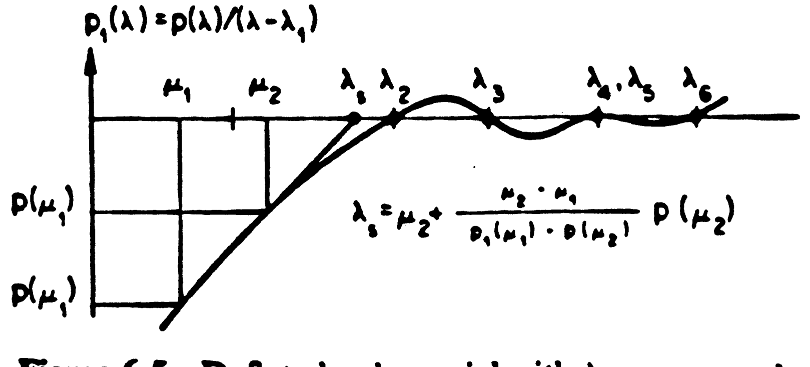


Figure 6.5 Deflated polynomial with λ_1 suppressed

(*iv*) *Shifting*

Inverse iteration converges to the numerically smallest eigenvalue. In order for the method to be used for other eigenvalues the following change of variable can be introduced

$$\lambda = \lambda_s + \rho \qquad (6.54)$$

Therefore Equation (6.50) can be written as

$$[K_s - \rho M]X = 0 \qquad (6.55)$$

where

$$K_s = K - \lambda_s M \qquad (6.56)$$

If λ_s is closer to λ_2 than to λ_1 the inverse iteration method when applied to Equation (6.55) will converge to λ_2. This follows from the standard power method proof.[1,2]

(*v*) *Sturm sequence check*

One of the potentially serious problems which can develop in the numerical evaluation of frequencies and mode shapes in a practical dynamic analysis is if important frequencies are neglected. The Sturm Sequence Check is a method which allows the engineer to verify the results of an eigenvalue problem. One can prove the Sturm Sequence Theorem as presented here from a direct examination of the complete family of deflated polynomials, $p(\lambda)$, $p_1(\lambda)$, $p_2(\lambda)$, ... One notes that they are all derived from the basic sequence for a given value of λ, or

$$\det(K - \lambda_s M) = D_{11} D_{22} D_{33} \cdots D_{NN} \qquad (6.57)$$

For a given value of λ the basic properties of this sequence of numbers are illustrated in Figure 6.6. It is apparent that the properties of this sequence of numbers can be used as a powerful technique in the numerical strategy of evaluating eigenvalues. If lowest eigenvalues are evaluated by any method the Sturm sequence can be calculated for

$$\lambda = 1{\cdot}001\lambda_n \qquad (6.58)$$

If all eigenvalues have been calculated the Sturm sequence should have n negative terms.[2] This assumes a desired accuracy of 0·001.

Another important application of the Sturm Sequence Technique is to evaluate the number of frequencies in a certain frequency range—say between λ_L and λ_H. Therefore LDL^T triangularizations of $[K - \lambda_H M]$ and $[K - \lambda_L M]$ will indicate the number of eigenvalues below each value. The difference will be the number of values within the range. In order to calculate the eigenpairs in the range one can shift into the range and use inverse iteration or subspace iteration to evaluate only the values of interest.

6.2.5 Transformation to uncoupled modal equations

The basic numerical properties of the undamped free vibration mode shapes are

$$M^* = \phi^T M \phi = I \tag{6.59}$$

$$K^* = \phi^T K \phi = \text{diag}\,(\bar{\omega}_n^2) \tag{6.60}$$

In which the mode shapes have been normalized so the generalized mass is one. Or

$$\phi_n^T M \phi_n = 1 \tag{6.61}$$

The following transformation, change of variable, is introduced into the basic equilibrium equations.

$$U(t) = \sum_{n=1}^{M} \phi_n X_n(t) = \phi X(t) \tag{6.62}$$

Therefore the velocities and accelerations are

$$\dot{U}(t) = \phi \dot{X}(t) \tag{6.63}$$

$$\ddot{U}(t) = \phi \ddot{X}(t) \tag{6.64}$$

If Equations (6.62), (6.63) and (6.64) are substituted into Equation (6.2) and the resulting matrix equation premultiplied by ϕ^T we obtain

$$M^* \ddot{X}(t) + C^* \dot{X}(t) + K^* X(t) = P(t) \tag{6.65}$$

The matrices M^* and K^* are diagonal; however C^* is not diagonal unless an assumption is made on the basic form of viscous damping which exists in the structure. Since damping is normally small and is difficult to physically model and identify the following assumption is normally made:

$$C^* \simeq \text{diag}\,(2\xi_n \bar{\omega}_n) \tag{6.66}$$

where ξ_n is the ratio of damping in mode n to the critical damping for the mode. With this assumption of uncoupled modal damping the typical modal equation can be written as

$$\ddot{X}_n(t) + 2\xi_n \bar{\omega}_n \dot{X}_n(t) + \bar{\omega}_n^2 X_n(t) = p_n(t) = c_n f(t) \tag{6.67}$$

After the modal equations are evaluated the time dependent displacements are calculated from Equation (6.62).

The same technique can be used to uncouple Equation (6.32) in order to avoid the solution of a large number of complex linear equations. For this case the following transformation is introduced

$$Y(\omega)=\phi Z(\omega) \tag{6.68}$$

The substitution of Equation (6.68) into Equation (6.32) yields a typical modal equation in the frequency domain.

$$[\bar{\omega}_n^2+2i\omega\bar{\omega}_n\xi_n-\omega^2]Z_n(\omega)=\phi_n\bar{F}(\omega) \tag{6.69}$$

For structures which are formulated in the frequency domain and whose behaviour can be represented by a limited number of natural frequencies this is numerically the most efficient approach.

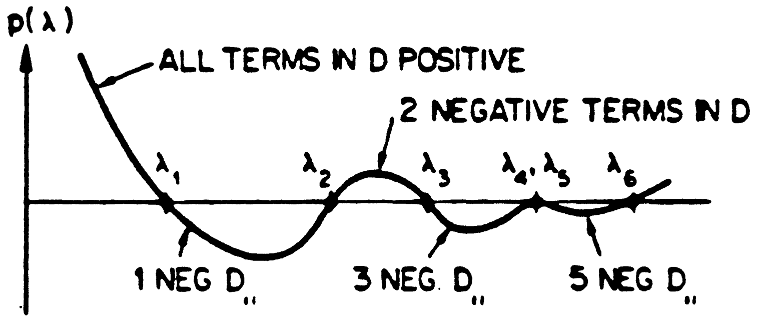


Figure 6.6 Sturm sequence properties of polynomial

6.2.5 Solution of modal equations

The solution to the single-degree of freedom modal equation given by Equation (6.67) can be accomplished by one of several methods. For certain loading which can be expressed as an analytic function exact mathematical solutions are possible.[1]

(a) Direct step-by-step solution
The most direct approach to the solution of this second order ordinary differential equation is to use a numerical finite difference method. The same techniques are possible which were used for the coupled equation. The step-by-step solution method given in Table 6.1 is numerically very efficient when applied to Equation (6.67).

(b) Duhamel Integral
In the case of arbitrary loading it is very common to express the solution in the form of the Duhamel Integral.[1] The Duhamel Integral is then numerically integrated. Since this numerical integration approach involves many numerical evaluations of trigonometric and exponential functions, which require series

expansions within a digital computer, the method cannot be considered a good numerical method. Also for high frequencies a very small integration interval is required for accuracy.

(*c*) *Transformation to the frequency domain*
The single degree of freedom modal equations can be transformed into the frequency domain. The Fourier integrals and transforms can be numerically evaluated by the Fast Fourier transform technique. This of course is identical to the approach suggested by Equation (6.69). This method introduces the same types of errors which are normally associated with the approximation of a function by a Fourier series or integrals.

(*d*) *Piecewise exact method*
Many types of loading can be represented by a series of straight lines between unequal time intervals. Most earthquake ground acceleration data is in this form. An exact mathematical solution is possible for a straight line loading subjected to displacement and velocity initial conditions. Therefore exact numerical values at any convenient time interval can be calculated. Table 6.3 summarizes the necessary equations for this approach for the numerical evaluation of the modal equations. This may not be the most numerically efficient method; but it is definitely the most accurate. For most structures the numerical solution of the modal equation involves an insignificant amount of computer time compared to the computer time required for the other phases of the problem—formation of stiffness and mass matrices, solution of eigenvalue problem, calculation of displacement and member stresses. For these reasons the piecewise exact method should be used whenever possible.

(*e*) *Response spectra analysis*
For many structural problems the dynamic load is not given in terms of a time dependent function; but the load is specified as a response spectra. By definition a response spectra is a plot of maximum values of displacement response v(max) obtained from the solution of the following equation for various values of ω.

$$\ddot{v}(t)+2\omega\xi\dot{v}(t)+\omega^2 v(t)=f(t) \tag{6.70}$$

A typical plot of the maximum, v(max), for specified ω and damping ratios ξ is shown in Figure 6.1.

The typical modal equation is of the form

$$\ddot{X}_n+2\bar{\omega}_n\xi_n\dot{X}_n+\bar{\omega}_n^2X_n=\phi_nF(t)=c_nf(t) \tag{6.71}$$

Therefore the maximum modal response can be calculated from

$$X_n(\max)=c_nv_n(\max) \tag{6.72}$$

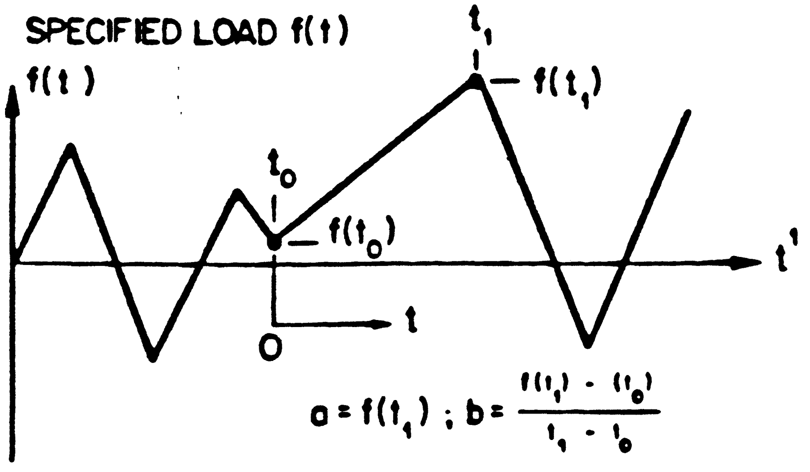
where v_n(max) is the value obtained from Figure 6.1 for values of ω_n and ξ_n.

Table 6.3 Piecewise exact solution method

BASIC EQUATION:

$$\ddot{X} + 2\xi\omega\dot{X} + \omega^2 X = f(t) = a + bt$$

SPECIFIED LOAD f(t)



$$a = f(t_1)\ ;\ b = \frac{f(t_1) - (t_0)}{t_1 - t_0}$$

EXACT DISPLACEMENT SOLUTION:

$$X(t) = A_0 + A_1\, t + A_2 e^{-\xi\omega t} \cos \omega^* t + A_3 e^{-\xi\omega t} \sin \omega^* t$$

where $A_0 = \frac{a}{\omega^2} - 2\frac{\xi b}{\omega^3}$

$$A_1 = \frac{b}{\omega^2}$$

$$A_2 = X(t_0) - A_0$$

$$A_3 = \frac{1}{\omega^*}(\dot{X}(t_0) + \xi\,\omega\, A_2 - A_1)$$

EXACT SOLUTION FOR VELOCITY:

$$\dot{X}(t) = A_1 + (\omega^* A_3 - \xi\omega\, A_2) e^{-\xi\omega t} \cos \omega^* t$$

$$- (\omega^* A_2 + \xi\omega\, A_3) e^{-\xi\omega t} \sin \omega^* t$$

After the maximum response in each modal equation is evaluated the maximum displacement in each modeshape for the complete structure is given by

$$U(\max)_n = \phi_n X_n(\max) \qquad (6.73)$$

For any particular degree of freedom a probabilistic value of displacement may be calculated from

$$u = \sqrt{(u_1^2 + u_2^2 + u_3^2 + \cdots u_m^2)} \qquad (6.74)$$

Other methods exist for adding maximum modal response; however the square root of the sum of the squares of the maximum modal values is one of the most common.

In order to evaluate member stresses it is first necessary to evaluate the member stresses due to each of the maximum modal responses. Or

$$\sigma(\max) = TU(\max)_n \tag{6.75}$$

where T is a stress-displacement transformation matrix for the member. The probabilistic member stress is estimated by

$$\sigma = \sqrt{(\sigma_1^2 + \sigma_2^2 + \sigma_3^2 + \cdots \sigma_M^2)} \tag{6.76}$$

Note that the probabilistic member stress cannot be calculated from the probabilistic displacements.

6.3 NON-LINEAR ANALYSIS

For offshore fixed structures several different types of non-linear behaviour for both static and dynamic loads are possible. Non-linear behaviour implies that the displacements and stresses produced by the different load conditions cannot be directly added; or the basic principle of superposition does not hold. Because there are so many different types of non-linearities there is not one general method which can be applied to all problems.

Large structures for which their weight is significant may have a dead load stress distribution which is highly dependent on the method of construction and installation sequence. The correct theoretical method of evaluating these stresses is to perform a complete analysis at each stage of construction with the internal stresses from the analysis of the previous stage used as initial conditions for the new analysis. This analysis technique can be used for the evaluation of installation stresses also. For subsequent analysis of the structure in which additional non-linear stresses are developed due to static or dynamic loads it may be extremely important to start the analysis with an accurate estimation of the initial stress conditions.

Perhaps the most common type of non-linear behaviour is due to non-linear materials. Most structural design requires that the structural materials remain in the elastic range during the design loads. However foundation stresses in soil may be non-linear under low stress levels. Under dynamic loads soil non-linearities are often approximated by an effective damping factor to be used in a linear dynamic analysis. Also during earthquake or large sea conditions some non-linear material behaviour can be tolerated without the collapse of the structure.

One of the most unlikely types of non-linear behaviour to be expected in a practical structure is the existence of large strains which require an alternate definition of stress. This type of non-linearity exists in only rubber-like materials.

For tall structures or structures supported by cables large displacements may exist under design loading. For this case it is extremely important that the static or dynamic equilibrium equations are satisfied in the deformed geometry.

For fixed offshore structures where the velocities of the structure are comparable to the water particle velocities the wave forces are non-linear and may be expressed as

$$F(t) = F(U(t), \dot{U}(t)) \tag{6.77}$$

This type of non-linear behaviour can be considered by the linear step-by-step methods. Based on the previous increment the velocity of the structure can be predicted from

$$\dot{U}_{t+\Delta t} = \dot{U}_t + \Delta t\, \ddot{U}_t \tag{6.78}$$

and a good estimate of the non-linear drag forces can be estimated. Since the properties of the structures do not change the effective stiffness matrix need not be modified or triangularized at each time increment. Therefore the method given in Table 6.1 can be used for this form of non-linearity.

A general formulation for the non-linear analysis of a structural system can be developed if Equation (6.1) is rewritten at time

$$(F_t^i + \Delta F_t^i) + (F_t^d + \Delta F_t^d) + (F_t^s + \Delta F_t^s) = F_{t+\Delta t} \tag{6.79}$$

in which the force changes are given by

$$\Delta F_t^i = M_t\, \Delta \ddot{U}_t, \quad \Delta F_t^d = C_t\, \Delta \dot{U}_t, \quad \Delta F_t^s = K_t U_t \tag{6.80}$$

where M_t, C_t and K_t are the approximate mass, damping and stiffness matrices at time t. Therefore Equation (6.79) can be rewritten as

$$M_t\, \Delta \ddot{U} + C_t\, \Delta \dot{U}_t + K_t\, \Delta U_t = F^* \tag{6.81}$$

where

$$F^* = F_{t+\Delta t} - F_t^i - F_t^d - F_t^S \tag{6.82}$$

A direct step-by-step method, as presented for linear systems, can be used for the evaluation of $\Delta \ddot{U}_t$, $\Delta \dot{U}_t$ and ΔU_t. Because the matrices M_t, C_t and K_t can only be approximated over the time interval, it is recommended that the forces F_t^i, F_t^d and F_t^s be recomputed at the end of the time increment. For example the structural forces F_t^s, which are consistent with U_t, should be evaluated as follows:

(a) From the displacement U_t calculate in member m the strain ε_m.

(b) From the specified non-linear stress strain behaviour calculate the stress τ_m.

(c) Using virtual work calculate the structural forces acting on element

$$F_m = \int A_m^T \tau_m \, dV_m$$

where A_m is the strain-displacement transformation matrix for member m.

(d) Calculate the total structural forces at time t from

$$F_t^s = \sum F_m$$

For structures with slight non-linearities this type of analysis may be accurate. However for very non-linear behaviour it may be necessary to iterate within a time step in order to minimize the accumulation of errors. For a more complete discussion of dynamic non-linear analysis methods the reader is referred to References 12, 20, 21, 22, 23, 24.

6.4 FINAL REMARKS

Several different numerical methods for the dynamic analysis of linear structural systems have been presented. Many of these methods have been incorporated into general purpose programs and have been successfully used in the solution of offshore structural systems.[24,25,26] General purpose programs for non-linear analysis have been developed based on the techniques presented.[22,24] However for non-linear analysis of complex offshore structures the development of special purpose computer programs is justifiable because of the unique nature of the structures and their loading.

REFERENCES

1. Clough, R. W. and Penzien, J. (1975). *Dynamics of Structures*, McGraw-Hill Book Company, New York, NY.
2. Bathe, K. J. and Wilson, E. L. (1976). *Numerical Methods in Finite Element Analysis*, Prentice-Hall, Inc., Englewood Cliffs, NJ.
3. Biggs, J. M. (1964). *Introduction to Structural Dynamics*, McGraw-Hill Book Company, New York, NY.
4. Hurty, W. C. and Rubinstein, M. F. (1964). *Dynamics of Structures*, Prentice-Hall, Inc., Englewood Cliffs, NJ.
5. Collatz, L. (1966). *The Numerical Treatment of Differential Equations*, Springer-Verlag, New York, NY, 116.
6. Crandall, S. H. (1956). *Engineering Analysis*, McGraw-Hill Book Company, New York, NY.
7. Forberg, C. E. (1969). *Introduction to Numerical Analysis*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts.
8. Newmark, N. M. (1959). 'A method of computation for struetural dynamics', ASCE, *Journal of Engineering Mechanics Division*, **85**, 67–94.
9. Houbolt, J. C. (1959). 'A recurrence matrix solution for the dynamic response elastic aircraft', *Journal of Aeronautical Science*, **17**, 540–550.

10. Wilson, E. L. (1960). 'A computer program for the dynamic stress analysis of underground structures', *Report UCSESM 68-1*, Department of Civil Engineering, University of California, Berkeley.
11. Wilson, E. L. (1969). 'Elastic dynamic response of axisymmetric structures', *Report UC SESM 69-2*, Department of Civil Engineering, University of California, Berkeley.
12. Wilson, E. L., Farhoomand, I., and Bathe, K. J. (1973). 'Non-linear dynamic analysis of complex structures', *International Journal of Earthquake Engineering and Structural Dynamics*, **1**, 241–252.
13. Bathe, K. J. and Wilson, E. L. (1973). 'Stability and accuracy analysis of direct integration methods', *International Journal of Earthquake Engineering and Structural Dynamics*, **1**, 283–291.
14. Wilson, E. L. (1974). 'The static condensation algorithm', *International Journal of Numerical Methods in Engineering*, **8**, 199–203.
15. Shubinski, R. P., Wilson, E. L., and Selna, L. G. (1966). 'Dynamic response of deepwater structures', *Civil Engineering in the Oceans*, ASCE Conference, San Francisco.
16. Bathe, K. J. (1971). 'Solution Methods of Large Generalized Eigenvalue Problems in Structural Engineering', *Report UC SESM 71-20*, Civil Engineering Department, University of California, Berkeley.
17. Bathe, K. J. and Wilson, E. L. (1972). 'Large eigenvalue problems in dynamic analysis', ASCE, *Journal of Engineering Mechanics Division*, **98**, 1471–1485.
18. Bathe, K. J. and Wilson, E. L. (1973). 'Eigensolution of large structural systems with small bandwidth', ASCE, *Journal of Engineering Mechanics Division*, **99**, 467–479.
19. Bathe, K. J. and Wilson, E. L. (1973). 'Solution methods for eigenvalue problems in structural mechanics', *International Journal for Numerical Methods in Engineering*, **6**, 213–226.
20. Chopra, A. K. (1974). 'Earthquake analysis of complex structures', *Proceedings ASME Conference, Applied Mechanics in Earthquake Engineering*, November 1974, New York.
21. Felippa, C. A. (1976). 'Procedures for computer analysis of large non-linear structural systems', *International Symposium on Large Engineering Systems*, University of Manitoba, Winnipeg, Canada, August 9–12, 1976.
22. Bathe, K. J., Ozdemir, H., and Wilson, E. L. (1974). 'Static and dynamic geometric and material nonlinear analysis', *Report UC SESM 74-4*, Department of Civil Engineering, University of California.
23. Bathe, K. J. (1976). 'An assessment of current finite element analysis of nonlinear problems in solid mechanics', *Proceedings Symposium on the Numerical Solution of Partial Differential Equations*, May 1975, Academic Press, Inc.
24. Bathe, K. J. (1975). 'ADINA—A finite element program for automatic dynamic incremental nonlinear analysis', *Report 82448-1*, Acoustics and Vibration Laboratory, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, Mass.
25. Bathe, K. J., Wilson, E. L., and Peterson, F. E. (1973). 'SAP IV—A structural analysis program for static and dynamic response of linear systems', *Report EERC 73-11*, College of Engineering, University of California, Berkeley, June 1973, revised April 1974.
26. *EAC/EASE 2 Dynamics—User Information Manual/Theoretical*, Control Data Corporation.

APPENDIX A

USE OF MS-DOS COMPUTER SYSTEMS

TABLE OF CONTENTS

1. SUMMARY

The major purpose of this "small document" is to provide the essential information for a new user/engineer to become productive on a DOS microcomputer with a minimum investment of time. It is assumed that the engineers' objectives are to use existing programs and to prepare data input files and technical documentation. In order to simplify the presentation only computers with large capacity fixed disks and a minimum of one removable floppy disk will be considered.

Only a small number of the most important DOS commands will be presented. Examples will be given on file duplication. In addition, the preparation of DOS BATCH COMMAND files will be illustrated by several examples.

2. FILE Preparation

The program which is used most often by an engineer at a microcomputer workstation is the "editor". This is normally a highly interactive program which allows the user to prepare input data files and to examine or modify any existing printable file. Therefore, the capability of this program is very important if the user/engineer is to be productive.

The editor "EDLIN", which is supplied with the DOS system, is a very difficult program to use. An engineer with a minimum of computer experience requires approximately two weeks to learn to effectively use this line oriented editor. In addition, the possibility of data input errors is greatly increased if EDLIN is used.

There is a large number of commercial editors available for DOS systems which can be purchased for $20 to $500. In most cases they have a large number of useful options; however, a significant amount of time is required to learn to use them productively. In addition, many of these editors create files with embedded "non-print" characters which are incompatible with standard DOS text, ASCII, input and output files.

Most of the well-known word proccessing programs can be used to prepare and edit standard DOS text files. For example, WORDSTAR can be used if the files are read and saved in the "nondocument mode". Also, WORDPERFECT can be used if files are read and saved in the "DOS text mode".

3. INTRODUCTION TO DOS

MS-DOS (Microsoft Disk Operating System) is the operating system for many computers with Intel 8088 to Intel 80486 series of Central Processing Units. The DOS monitor allows all computer programs rapid access to the DOS file management system. Since certain files are independent of the programming language it is possible to transfer information between programs which have been written in different programming languages such as FORTRAN, BASIC or C.

Files are stored on large capacity fixed disks or removable floppy disks. Each disk has a root "directory" and "subdirectories" which indicate the name, size and location of all files on the disk. System disks have the basic DOS operating system on a reserved section of the disk. When the computer is started or "booted" the DOS system and the COMMAND.COM are read into the computer, control is transferred to the DOS monitor program, and the "C>" (or "A>") prompt is displayed on the console indicating that "C" is the currently logged-in drive and that the DOS system is awaiting a command. The user can then enter any legal DOS operation followed by a carriage return (CR).

A DOS command is either one of the INTERNAL commands, contained in the COMMAND.COM file, or it is a request to execute a program which is stored on any disk drive (and directory) which is connected to the computer system. Also, the user can switch to another "logged disk" by typing the new drive name (A,B,C or D) followed by a colon (:) and terminated by a CR. Also, the user can transfer control to another directory with the change directory, CD, command.

FILE NAMES

A file name consists of two parts: the primary name (1 to 8 characters) and an extension name (1 to 3 characters) separated by a period. For example, the FORTRAN source program for the computer program CAL would be named CAL.FOR; its binary relocatable file would be named CAL.OBJ and its executable (command) file would be named CAL.EXE. The general form of a file name is

```
"drive":\path\"name"."extension"
```

If the "drive" is not specified it is taken as the currently logged drive. If the path is not specified it is taken as the current directory. The following symbols are not allowed as characters within a file name:

< > . , ; : = * [ ] / \

An ambiguous file name is used to refer to one or more files on a disk. For example:

SOLID.*	References all files on the disk with the first name SOLID
*.EXE	References all files on the disk with the second name EXE
CA*.FOR	References all .FOR files which have CA as the first two letters of its name

SPECIAL DOS CONTROL CHARACTERS

During the execution of DOS programs the following control characters can be entered at the keyboard:

control C	Causes termination of the execution of the program
control P	All subsequent screen output is also directed to the printer until the next control P is typed
control S	Screen output is stopped temporarily in order to view a segment of output. Screen output will continue when any other character is typed.
Prt Sc	Causes the current screen display to be printed

4. INTERNAL DOS COMMANDS

The following commands are built into DOS and are available when the COMMAND.COM program is loaded:

DIRECTORY COMMAND

DIR	Causes the names of all files on the currently logged disk to be listed on the console.
DIR A:*.FOR	Causes the names of all FORTRAN files on the disk mounted on drive A to be listed on the console.

The D program, given on page 4, can be used in place of DIR.

DELETE COMMAND

DEL XX.EE	The file named XX.EE is removed from the current directory and its disk space is no longer reserved.
DEL A:*.BAK	All files in the current directory on the disk mounted on drive A with the second name BAK are removed.
DEL *.*	All files are removed from the currently logged disk.

RENAME COMMAND

The name of a file on a disk may be changed by the REN command. For example:

REN ZQY.AA XX	Causes the file ZQY.AA to be renamed to XX.

TYPE COMMAND

The TYPE B:XX.YY command causes the contents of the file named XX.YY on drive B to be displayed on the console. Control S will cause the display to be halted temporarily. The TYPE XX.YY|MORE will cause the display to be halted every 24 lines.

COPY COMMAND

The COPY program is one of the most important programs which operates under the DOS system. It allows files to be duplicated and copied to other directories, disks or output devices. As examples:

```
COPY  CAL.FOR  CAL.BAK
```

The above command will create a new file CAL.BAK, within the current directory, which is identical to the file CAL.FOR.

```
COPY  *.*  A:\ZZ\
```

The above command will copy all files from the current directory to directory ZZ on drive A. The new files on A will have the same names.

```
COPY  *.FOR  A:
```

The above command will copy all files with extensions .FOR from the current directory to drive A. The files copied to drive A will replace files which previously existed with the same name.

DIRECTORY OPERATIONS

It is possible to use a DOS computer system without defining subdirectories. For such a system all programs and data files would exist in the "root directory" and when the DIR command is given a list of all files would be displayed. In addition, a potential problem with conflicting file names exists if several individuals use such a system. Also, systems without subdirectories are cumbersome to use and to maintain. It is very convenient to create separate directories for special functions. For other problems it is useful to create temporary work areas containing files which may be easily deleted after the problem is completed.

For most systems only one level of subdirectories is required. This approach greatly simplifies the use of "path names" and the general use and maintenance of a system. However, a good approach is to retain a minimum number of files in the root directory and to allow access to the important programs by creating a PATH to all directories which contain these programs with the use of the SET PATH command.

The DOS command MD new will create a new subdirectory, or work area, on the current disk named new which will be able to store data and DOS programs.

After a directory is made it is possible for the user to transfer control to that subdirectory by executing the command CD \new. To return to the root directory the command CD \ is executed.

If all files are deleted from a subdirectory and control is in the root directory the execution of RD new will delete the directory new.

4. ADDITIONAL DOS PROGRAMS

All computer programs which can be executed on a DOS system are stored as files of the form NAME.COM or NAME.EXE and are executed by typing the command NAME. If the program is on a drive other than the logged drive it is executed by the specification of the drive name. For example: A:xxxx will cause the program xxxx to be loaded from drive A and executed. The following is a partial list of some of the most useful programs (DOS commands).

FORMAT PROGRAM

This program must be used to initialize the density and record size for every new diskette. The standard international interchange format is the 360k double sided double density 5 1/4 inch disk. However, for maximum efficiency and speed, disks which are to be used locally on one computer system should be initialized at maximum capacity. The FORMAT operation defines the density of a disk which may be different than what the maker of the disk writes on the label.

DISKCOPY and DISKCOMP PROGRAM

The DISKCOPY is used to copy all files on one disk to another disk and FORMATS the new disk during the process. The DISKCOMP program is used to compare the contents of two disks.

PRINT PROGRAM

The PRINT program can be executed at the same time as other DOS commands or programs are being executed. Therefore, files can be printed in a "background mode" and the user can perform other tasks without delay. In addition, the printing of several files can be initiated at the same time as indicated in the following:

```
PRINT file1 file2 file3 - - -
```

The command PRINT/T can be used to terminate printing after the completion of the current file.

5. DOS BATCH CAPABILITY

This basic DOS function allows a series of DOS operations and user programs to be executed in sequence without the requirement that the user type the series of DOS commands. In order to utilize this option the series of commands must be stored in a "submit file" which is prepared by an editor. The second name of the batch file must be ".BAT". To illustrate this option let us assume that the series of programs SAP, FRAME, SOLVE and FRAMEF are to be executed in sequence. First: prepare a "batch file" with the name "SERIES.BAT" which contains the following information:

```
SAP
FRAME
SOLVE
FRAMEF
```

Next: type the DOS command SERIES and the above list of programs will be executed in sequence without the computer stopping after each program.

Execution of the batch operation can be terminated by typing a CONTROL C. The batch operation is extremely useful in "linking together" typically used DOS operations.

The more general form of the batch operation program is one which has several parameters (arguments) which are specified as variables within the .BAT file as indicated below:

```
"batch file name" P1 P2 - - - Pn
```

The variables %1 %2 - - within the batch file are replaced by names P1 P2 - - during the execution of the BATCH operation. An example of the use of this form of the batch operation is to erase a series of files which are generated by the program SAP without deleting the input data file which has the same first name and no extension. First, the following file named ER.BAT is prepared:

```
DEL TEMP
REN %1 TEMP
DEL %1.*
REN TEMP %1
```

If the input data file is named FRMEX, then all files with the first name FRMEX must be removed except the data file itself. Therefore, the command

```
ER  FRMEX
```

will perform this function and eliminate the need to type four DOS commands. The practicality of the batch operation is only limited by the creativity of the user.

6. CONFIG.SYS, COMMAND.COM and AUTOEXEC.BAT

When a DOS computer system is booted (started), or manually rebooted, DOS looks for the file CONFIG.SYS in the root directory. If the file is found, DOS interprets the information within the file in order to configure the DOS system parameters or to select a different COMMAND processor (SHELL). If the CONFIG.SYS file is not found the default values, shown below within [ ], are used.

If a different COMMAND processor is not defined the file COMMAND.COM is loaded from the root directory. Then, DOS automatically searches the root directory for the file AUTOEXEC.BAT and, if found, executes the DOS commands contained within the file. The files CONFIG.SYS and AUTOEXEC.BAT are optional files; however, their existence is very important if the computer system is to be configured and initialized for professional use. Both files can be prepared or modified by the editor EDED.

A typical CONFIG.SYS file, which is shown below, contains the five different options which are possible:

```
BREAK=ON            [OFF]     (allows CTRL/BREAK to terminate DOS
                              operations)
BUFFERS=40     [10]  (RAM buffers reserved for I/O operations)
FILES=15       [12]  (files which can be opened concurrently)
DEVICE=ANSI.SYS          (defines extended screen and keyboard functions)
SHELL=COMMAND.COM        (defines standard COMMAND processor)
```

A typical AUTOEXEC.BAT file is shown below:

```
VERIFY ON                          (all COPY operations will be verified)
SET PATH=C:\;C:\DOS;C:\SAP;C:\CAL;      (sets path)
SET PROMPT=$36$P$36$G      (DOS prompt will display directory name)
GRAPHICS                   (allows screen print of graphics)
VER                        (displays version of DOS)
```

The PATH command is one of the most useful commands in the DOS system. If a DOS program is executed, when operating within a directory on any disk, and the program is not stored within the directory disk space DOS will automatically search for the program in other directories in the order defined by the SET PATH operation. Therefore, there is no reason why duplicate copies of programs should exist in different directories.

D I R E C T   S T I F F N E S S   O P E R A T I O N S

```
SLOPE  Ki+     E=? I=? L=?  FORMS 4 X 4 STIFFNESS MATRIX
FRAME  Ki+ Ti+ Gi+  E=? I=? A=? S=?  X=?,? Y=?,?  P=?
          GEOMETRIC STIFF. MATRIX Gi IS FORMED IF P NOT ZERO
TRUSS  Ki+ Ti+ E=? A=? N=Ni,Nj  FORMS TRUSS STIFF.
FRAME3 Ki+ Ti+ E=? A=? I=I3,I2 J=? N=Ni,Nj P=P1,P2
LOADI  ID R=? C=? LOAD ARRAY "ID" OF INTEGER NUMBER
ADDK   K+ Ki ID N=?  ADD ELEMENT STIFFNESS TO TOTAL STIFFNESS
MEMFRC Ki U ID Fi+ N=?  EVALUATION OF MEMBER FORCES
```

S T R U C T U R A L   D Y N A M I C   O P E R A T I O N S

```
EIGEN  K- V+ M- EIGENVALUES OF KV = MVe  - DIAGONAL MASS
JACOBI K- V+ M- e EIGEN SOLUTION FOR FULL MASS MATRIX
SQREL  M1-      REPLACES EACH TERM OF M1 WITH ITS SQUARE ROOT
INVEL  M1-      REPLACES EACH TERM OF M1 WITH ITS INVERSE
DYNAM  W C F G(t) X(t) DT=? N=?  UNCOUPLED DYNAMIC RESPONSE
NORM   M1 M2+ T=?    FORMS COLUMN MATRIX M2 WHERE
                Where T=0 SUM ABS-VALUES OF ROWS
                      T=1 SRSS OF THE ROWS
MAX  X Xmax FORMS Xmax FROM MAXIMUM ABS. VALUES OF ROWS OF X
STEP K- M C UVA- U+ P F(t) DT=? L=Li,Lmax P=deta,alpha,theta
PLOT M1 N=? R=R1,R2,.. S=S1,S2,.. PLOTS "N" ROWS MATRIX M1
      Where Ri = THE ROWS TO PLOT and Si = SYMBOLS FOR ROW i
RITZ K- M R V+  NV=?  S=?
           NV= # OF RITZ VECTORS TO BE GENERATED
            S= NONZERO IF STATIC VECTOR IS NOT RETAINED
DFT  F(T)- DT=? DISCRETE FOURIER TRANS.-F(T) REPLACED BY F(W)
IDFT F(W)-     F(W) REPLACED BY F(T)
RADIUS F(W)  R(W)+
FSOLVE W C F G(W) Y(W)+ DT=? FREQUENCY DOMAIN SOLUTION
```

L O O P I N G   O P E R A T I O N S

```
LOOP endloop N=?     EXECUTE ALL OPERATIONS BEFORE LINE "endloop" ON
                     INPUT FILE. N = NUMBER OF TIMES TO SUBMIT (DEFAULT=1)

IF M1  M2            TERMINATES LOOP IF M1 IS LESS THAN M2
                     (PLACE BEFORE "endloop" LINE)

endloop              USER SELECTED NAME FOR LOOP TERMINATOR
```

B A S I C   C A L - 9 1   O P E R A T I O N S

```
RUN                 EXECUTE OPERATIONS  FROM INPUT FILE
HELP (H)            LIST CAL COMMANDS
LOAD M1 R=? C=?     LOAD MATRIX M1 OF REAL NUMBERS
ZERO  M1+ NR=? NC=? ZERO A REAL MATRIX M1
LIST (L)            LIST THE  DIRECTORY OF ALL ARRAYS
PRINT (P)  M1       LIST ARRAY NAMED "M1"'
SAVE OR STOP (S)    TERMINATE PROGRAM AND SAVE DATABASE
QUIT (Q)            TERMINATE PROGRAM
START               NEW PROBLEM NAME
DELETE (D) M1-      DELETE ARRAY NAMED "M1"'
MODIFY M1-          MODIFY TERMS IN MATRIX M1
RESUME              READ INCORE DATA BASE FROM PREVIOUS RUN
WRITE M1            WRITE ARRAY M1 TO DISK
READ  M1            READ ARRAY M1 FROM DISK
RETURN              RETURNS TO INTERACTIVE MODE
```

S T A N D A R D   M A T R I X   O P E R A T I O N S

```
MULT M1 M2 M3+      MULTIPLY M1 * M2 = M3
TMULT M1 M2 M3+     TRANSPOSE OF M1 * M2 = M3
ADD M1- M2          ADD MATRIX M2 TO MATRIX M1
SUB M1- M2          SUBTRACT MATRIX M2 FROM MATRIX M1
TRAN M1 M2+         TRANSPOSE MATRIX M1 TO FORM MATRIX M2
DUP M1 M2+          DUPLICATE MATRIX M1 TO MATRIX M2
STODG M1- M2        STORES MATRIX M2 ON DIAGONAL OF MATRIX M1
DUPDG M1 M2+        DUPLICATES DIAGONAL OF M1 TO MATRIX M2
SCALE M1- M2        SCALE MATRIX M1 BY THE TERM M2(1,1)
INVERT M1-          INVERSION OF SYMMETRIC MATRIX "M1"
SOLVE M1- M2- S=?  SOLVE M1 x  = M2
                   S=0  SOLVE Ax = B
                   S=1  TRIANGULARIZE M1 ONLY
                   S=2  FORWARD SUBSTITUTE ONLY
                   S=3  BACK SUBSTITUTE ONLY
STOSM M1 M2 L=L1,L2     STORES MATRIX M2 IN MATRIX
                        M1 AT LOCATION M1(L1,L2)
DUPSM M1 M2+ NR=? NC=? L=L1,L2  DUPLICATES SUBMATRIX M2
                  FROM LOCATION M1(L1,L2)   M2 IS NR x NC
```