MCR-76-217
Contract NAS8-31376

Volume IIIB

Explanations, Sparse FORMA Subroutines

May 1976

Expansion and Improvement of the FORMA System for Response and Load Analysis

([illegible]-144512) EXPANSION AND IMPROVEMENT OF THE FORMA SYSTEM FOR RESPONSE AND LOAD ANALYSIS. VOLUME 3B: EXPLANATIONS, SPARSE FORMA SUBROUTINES (Martin Marietta Corp.) [illegible] HC [illegible] CSCL 13M G3/39 N76-25595 Unclas 43040

MARTIN MARIETTA

MCR-76-217

Contract NAS8-31376

EXPANSION AND IMPROVEMENT OF THE FORMA SYSTEM FOR RESPONSE AND LOAD ANALYSIS

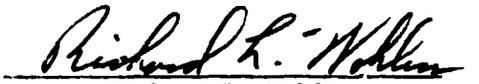
Volume IIIB - Explanations, Sparse FORMA Subroutines

May 1975

Author:

Richard L. Wohlen

Approved:



Richard L. Wohlen
Program Manager

Prepared for: National Aeronautics and Space Administration
George C. Marshall Space Flight Center
Huntsville, Alabama 35812

MARTIN MARIETTA CORPORATION
Denver Division
Denver, Colorado 80201

FOREWORD

This report presents results of the expansion and improvement of the FORMA system for response and load analysis. The acronym FORMA stands for FORTRAN Matrix Analysis. The study, performed from 16 May 1975 through 17 May 1976 was conducted by the Analytical Mechanics Department, Martin Marietta Corporation, Denver Division, under the contract NAS8-31376. The program was administered by the National Aeronautics and Space Administration, George C. Marshall Space Flight Center, Huntsville, Alabama under the direction of Dr. John R. Admire, Structural Dynamics Division, Systems Dynamics Laboratory.

This report is published in seven volumes:

Volume I - Programming Manual,
Volume IIA - Listings, Dense FORMA Subroutines,
Volume IIB - Listings, Sparse FORMA Subroutines,
Volume IIC - Listings, Finite Element FORMA Subroutines,
Volume IIIA - Explanations, Dense FORMA Subroutines,
Volume IIIB - Explanations, Sparse FORMA Subroutines, and
Volume IIIC - Explanations, Finite Element FORMA Subroutines.

CONTENTS

ABSTRACT

This report presents techniques for the solution of structural dynamic systems on an electronic digital computer using FORMA (FORTRAN Matrix Analysis).

FORMA is a library of subroutines coded in FORTRAN IV for the efficient solution of structural dynamics problems. These subroutines are in the form of building blocks that can be put together to solve a large variety of structural dynamics problems. The obvious advantage of the building block approach is that programming and checkout time are limited to that required for putting the blocks together in the proper order.

The FORMA method has advantageous features such as:

1. subroutines in the library have been used extensively for many years and as a result are well checked out and debugged;

2. method will work on any computer with a FORTRAN IV compiler;

3. incorporation of new subroutines is no problem;

4. basic FORTRAN statements may be used to give extreme flexibility in writing a program.

Two programming techniques are used in FORMA: dense and sparse.

ACKNOWLEDGMENTS

The editor expresses his appreciation to those individuals whose assistance was necessary for the successful completion of this report. Dr. John R. Admire was instrumental in the definition of the program scope and contributed many valuable suggestions. Messrs. Carl Bodley, Wilcomb Benfield, Darrell Devers, Richard Hruda, Roger Philippus, and Herbert Wilkening, all of the Analytical Mechanics Department, Denver Division of Martin Marietta Corporation, have contributed ideas, as well as subroutines, in the formulation of the FORMA library.

The editor also expresses his appreciation to those persons who developed FORTRAN, particularly the subroutine concept of that programming tool.

LIST OF SYMBOLS

$[\ ]$	matrix
$\{\ \}$	column matrix } vector
$\{\ \}^T$	row matrix } vector
T	transpose (when symbol is a superscript)
$[\ ]_{m \times n}$	m designates the row size of matrix n designates the column size of matrix
a_{ij}	a designates an element of matrix [A] i designates the ith row of matrix [A] j designates the jth column of matrix [A]

I. INTRODUCTION

This volume presents an explanation of the function of each sparse subroutine in the FORMA library. Example problems are given in some cases to clarify the operations performed by a subroutine.

II. SUBROUTINE EXPLANATIONS

The subroutines are given in alphabetical order with numbers coming before letters.

Subroutine YAA calculates the multiplication of a matrix, in FORMA sparse notation, by a scalar. In matrix notation,

$$[Z]_{NRxNC} = \alpha [A]_{NRxNC}$$

where

$$z_{ij} = \alpha a_{ij} \text{ for } \begin{pmatrix} i = 1, NR \\ j = 1, NC \end{pmatrix} .$$

NR is the number of rows of each matrix, and
NC is the number of columns of each matrix.

EXAMPLE

Consider input of $\alpha = 2$; and

$$[A]_{2x3} = \begin{bmatrix} 7. & 2. & -3. \\ 4. & 5. & 6. \end{bmatrix}$$

The reader can easily verify the output to be

$$[Z]_{2x3} = 2. \begin{bmatrix} 7. & 2. & -3. \\ 4. & 5. & 6. \end{bmatrix}$$

$$= \begin{bmatrix} 14. & 4. & -6. \\ 8. & 10. & 12. \end{bmatrix}$$

DESCRIPTION OF TECHNIQUE

[A] and [Z] are stored on utility tapes NUTA and NUTZ as row partitioned matrices in FORMA sparse notation.

Matrix [A] is read from NUTA and each non-zero element is multiplied by α. Each partition of [A] then becomes a partition of [Z] and is stored on NUTZ.

Subroutine YAABB calculates the summation of two matrices, each matrix multiplied by a scalar. Both matrices are in FORMA sparse notation. In matrix notation,

$$[Z]_{NR\times NC} = \alpha[A]_{NR\times NC} + \beta[B]_{NR\times NC}$$

where

$$z_{ij} = \alpha\, a_{ij} + \beta\, b_{ij} \qquad \begin{matrix} i = 1, NR \\ j = 1, NC \end{matrix}$$

NR is the number of rows of each matrix, and

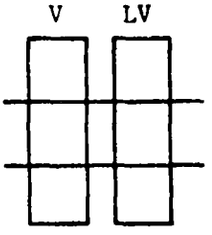
NC is the number of columns of each matrix.

The number of rows of [A] and [B] must be equal, and the number of columns of [A] and [B] must be equal.

DESCRIPTION OF TECHNIQUE

[A], [B], and [Z] are stored on utility tapes NUTA, NUTB, and NUTZ as row partitioned matrices in sparse notation.

Partitions of the matrices are operated on in core in the V and LV work spaces as follows: (ℓ, m, n refer to partition number)



[A] (ℓ) ℓ = 1, NPARTA

[B] (m) m = 1, NPARTB

[Z] (n) n = 1, NPARTZ

The locations of elements in [A] and [B] are compared with each other. If the locations are equal, elements a and b are summed and stored in [Z]. If the locations are not equal, the lesser location element is stored in [Z].

Subroutine YASSEM places (assembles) a matrix [A] into a second matrix [Z] starting at a designated row, column location (IRZ, JCZ, respectively) in [Z]. Both matrices are in FORMA sparse notation. The elements of [A] will replace corresponding elements in the original [Z] . Before the first use of this subroutine in forming [Z] , it is important that [Z] is correctly defined. For example, if [Z] is to be originally null, subroutine YZERO should be used. This subroutine may be called repeatedly to form [Z] from the assembly of several [A] matrices. The [A] matrix must be within the row, column limits of [Z] . In subscript notation,

$$z_{ij} = a_{k\ell} \text{ for } \begin{pmatrix} k = 1, \text{ NRA} \\ \ell = 1, \text{ NCA} \end{pmatrix}$$

where

i = k + IRZ -1;
j = ℓ + JCZ -1;
NRA is the number of rows in [A] ; and
NCA is the number of columns in [A] .

EXAMPLE

Consider a matrix defined as

$$[Z]_{3x4} = \begin{bmatrix} 1. & 0. & 0. & 0. \\ 0. & 2. & 0. & 0. \\ 0. & 0. & 0. & 0. \end{bmatrix}.$$

Matrix $[A]_{2x3} = \begin{bmatrix} 3. & 4. & 5. \\ 6. & 7. & 8. \end{bmatrix}$ is to be assembled

into [Z] starting at the 2, 1 location (IRZ = 2, JCZ = 1) of [Z] .

The result of this operation will be

$$[Z]_{3x4} = \begin{bmatrix} 1. & 0. & 0. & 0. \\ 3. & 4. & 5. & 0. \\ 6. & 7. & 8. & 0. \end{bmatrix}$$

DESCRIPTION OF TECHNIQUE

[A] and [Z] are stored on utility tapes NUTA and NUTZ as row partitioned matrices in FORMA sparse notation.

Matrix [A] is read from NUTA one partition at a time. The location numbers of its non-zero elements are revised by adding the quantity [100000*(IRZ-1) + (JCZ-1)] to them and the revised matrix [$\bar{A}$] is stored on NUT1.

In the next operation, matrix [Z] is read from NUTZ one partition at a time. Any non-zero elements existing in the area of the matrix where [$\bar{A}$] is to be inserted, are eliminated and [$\bar{Z}$] is stored on NUTZ.

Matrix [$\bar{Z}$] is read from NUT2 and written on NUTZ. Matrix [$\bar{A}$] is then read from NUT1 and stored with [$\bar{Z}$] on NUTZ.

The final step is accomplished by calling subroutine YLORD to correctly order the remaining non-zero elements of [$\bar{Z}$] together with the revised elements of [A] , and the resulting matrix is stored on NUTZ.

Subroutine YBSL3A solves the linear simultaneous algebraic equations [A] [Z] = [B], where [A] has been previously decomposed into $[U]^T$ [D] [U]. In matrix notation,

$$[A]_{N\times N}\ [Z]_{N\times M} = [B]_{N\times M} \qquad [1]$$

Decomposition of [A] gives

$$[U]^T\ [D]\ [U]\ [Z] = [B] \qquad [2]$$

Letting

$$[U]\ [Z] = [Y] \qquad [3]$$

and

$$[D]\ [Y] = [G] \qquad [4]$$

gives

$$[U]^T\ [G] = [B]. \qquad [5]$$

Because

$$[U]^T = \begin{bmatrix} 1 & & & 0 \\ u_{12} & 1 & & \\ u_{13} & u_{23} & 1 & \\ \vdots & & & \\ u_{1N} & u_{2N} & u_{3N} \cdots & 1 \end{bmatrix},$$

the elements of [G] can be solved as

$$g_{1j} = b_{1j} \qquad j = 1, M$$

$$g_{ij} = b_{ij} - \sum_{k=1}^{i-1} u_{ik}^T\ g_{kj} \qquad \begin{pmatrix} i = 2, N \\ j = 1, M \end{pmatrix}$$

Using [4] we can now solve for [Y] as

$$y_{ij} = g_{ij}/d_{ii}.$$

Now, because

$$[U] = \begin{bmatrix} 1 & u_{12} & u_{13} & \cdots & u_{1N} \\ & 1 & u_{23} & \cdots & u_{2N} \\ & & 1 & \cdots & u_{3N} \\ & & & \ddots & \vdots \\ 0 & & & & 1 \end{bmatrix}$$

the back solution for [Z] is

$$z_{Nj} = y_{Nj} \qquad j = 1, M$$

$$z_{ij} = y_{ij} - \sum_{k=i+1}^{N} u_{ik}\, z_{kj} \qquad \begin{pmatrix} i = 2, N \\ j = 1, M \end{pmatrix}$$

Subroutine YBSL3A uses [U] and [D] as input, but expects them in a special format. They are not in FORMA sparse notation. Input matrix [B] and output matrix [Z] are in standard FORMA sparse notation.

Subroutine YBTAB calculates the triple matrix product for matrices in FORMA sparse notation. In matrix notation

$$[Z]_{NCB \times NCB} = [B]^T_{NCB \times NRB} \; [A]_{NRB \times NRB} \; [B]_{NRB \times NCB}$$

where

$$z_{ij} = \sum_{\ell=1}^{NRB} \sum_{k=1}^{NRB} b_{\ell i} \; a_{\ell k} \; b_{kj} \quad \begin{pmatrix} i = 1, NCB \\ j = 1, NCB \end{pmatrix} .$$

NRB is the number of rows of $[B]$ and the size of $[A]$ (square).
NCB is the number of columns of $[B]$ and the size of $[Z]$ (square).

Because the number of columns of $[A]$ must be equal to the number of rows of $[B]$, and because the number of columns of $[B]^T$ (or the number of rows of $[B]$) must be equal to the number of rows of $[A]$, then $[A]$ must be a square matrix. The answer $[Z]$ is likewise a square matrix.

Theorm: If $[A]$ is symmetric (that is, $[A] = [A]^T$), then $[Z]$ is symmetric.

Proof:
$$[Z]^T = ([B]^T [A] [B])^T$$
$$= [B]^T [A]^T ([B]^T)^T$$
$$= [B]^T [A] [B]$$
$$= [Z] .$$

DESCRIPTION OF TECHNIQUE

$[A]$, $[B]$, and $[Z]$ are stored on utility tapes NUTA, NUTB, and NUTZ respectively as row partitioned matrices in FORMA sparse notation.

Partitions of matrix $[B]$ are read from NUTB and the location numbers of the non-zero elements are transposed. These partitions now are partitions of $[B]^T$ and are stored on NUTZ.

Subroutine YMULT is called to form the product of $[B]^T [A]$. This product is stored on NUT1.

If [A] is not symmetric, subroutine YMULT is called to complete the triple matrix product $([B]^T [A])\ [B]$, and the resulting [Z] is stored on NUTZ.

If [A] is symmetric, the result of $[B]^T$ [A] is read into core, one partition at a time, from NUT1. Each row of $[B]^T$ [A] is postmultiplied by [B] giving a row of [Z] . To save computer time, only the z_{ij} for j = 1,i are formed givirg the lower half of the symmetric matrix [Z] . [Z] is stored on NUTZ. The logic used in this portion of YBTAB is a slight modification of the logic used in YMULT.

Subroutine YDCM3A decomposes [A] a matrix in FORMA sparse notation to form an upper triangular matrix with ones on the diagonal [U] and a diagonal matrix [D] such that $[A] = [U]^T [D] [U]$. [A] must be real, square, and symmetric. The decomposition technique is attributed to Gauss. In matrix notation

$$[A]_{N\times N} = [U]^T_{N\times N} [D]_{N\times N} [U]_{N\times N} \qquad [1]$$

where

$$[U]_{N\times N} = \begin{bmatrix} 1 & u_{12} & u_{13} & \cdots & u_{1N} \\ 0 & 1 & u_{23} & \cdots & u_{2N} \\ 0 & 0 & 1 & \cdots & u_{3N} \\ . & . & . & . & . \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \qquad [1a]$$

and

$$[D]_{N\times N} = \begin{bmatrix} d_{11} & 0 & & \cdots & 0 \\ 0 & d_{22} & 0 & \cdots & 0 \\ 0 & 0 & d_{33} & \cdots & 0 \\ . & . & & & . \\ 0 & 0 & 0 & \cdots & d_{NN} \end{bmatrix} .$$

N is the size of the matrices (square):

DESCRIPTION OF TECHNIQUE

To determine the elements of [U] and [D], first consider the multiplication of two matrices.

$$[A] = [C] [U] \qquad [2]$$

where [C] is a lower triangular matrix of the form

$$[C] = \begin{bmatrix} c_{11} & 0 & \cdots & 0 \\ c_{21} & c_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ c_{N1} & c_{N2} & \cdots & c_{NN} \end{bmatrix}$$

and [U] has been previously defined.

Because [A] is symmetric,

$$[A] = [A]^T$$

$$= [U]^T [C]^T$$

$$= [U]^T \begin{bmatrix} c_{11} & 0 & \cdots & 0 \\ 0 & c_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & c_{NN} \end{bmatrix} \begin{bmatrix} 1 & \frac{c_{21}}{c_{11}} & \cdots & \frac{c_{N1}}{c_{11}} \\ 0 & 1 & \cdots & \frac{c_{N2}}{c_{22}} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Due to the uniqueness of the factorization of [A] into the product of the two triangular matrices, one of which has ones on the diagonal, it follows that

$$\begin{bmatrix} 1 & \frac{c_{21}}{c_{11}} & \cdots & \frac{c_{N1}}{c_{11}} \\ 0 & 1 & \cdots & \frac{c_{N2}}{c_{22}} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & & 1 \end{bmatrix} = [U].$$

By defining

$$[D] = \begin{bmatrix} c_{11} & 0 & \dots & 0 \\ 0 & c_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & c_{NN} \end{bmatrix},$$

Equation [1] is thus derived from Equation [2].

By performing the matrix products indicated by Equation [1], the elements in [A] are obtained generally as

$$a_{ij} = u_{1i}\, d_{11}\, u_{1j} + u_{2i}\, d_{22}\, u_{2j} + \dots + d_{ii}\, u_{ij}$$

and

$$a_{ii} = d_{11}\, u_{1i}^2 + d_{22}\, u_{2i}^2 + \dots + d_{ii}$$

From these two equations we calculate

$$d_{11} = a_{11}$$

$$u_{1j} = a_{1j}/d_{11} \qquad (j = 2, N)$$

$$d_{ii} = a_{ii} - \sum_{k=1}^{i-1} d_{kk}\, u_{ki}^2 \qquad (i = 2, N)$$

$$u_{ij} = \left(a_{ij} - \sum_{k-1}^{i-1} u_{ki}\, d_{kk}\, u_{kj}\right) \Big/ d_{ii} \qquad (i < j)$$

$$u_{ii} = 1.0 \qquad (i = 1, N)$$

$$u_{ij} = 0.0 \qquad (i > j)$$

Because of the shape of [A] and the corresponding shape and density of [U], this decomposition is performed using banded programming logic. [U] and [D] are not stored in FORMA sparse notation but are in a form to be used as input to subroutine YBSL3A.

MISCELLANEOUS

The diagonal elements of [D] are the determinant ratios of [A]. The determinant of [A] is the product of these determinant ratios. That is,

$$|A| = \prod_{i=1}^{N} d_{ii}$$

REFERENCE

V. N. Faddeeve, *Computational Methods of Linear Algebra*. Dover Publications, Inc., New York, 1959.

Subroutine YDCOM2 decomposes [A] a matrix in FORMA sparse notation to form an upper triangular matrix with ones on the diagonal [U] and a diagonal matrix [D] such that $[A] = [U]^T [D] [U]$. [A] must be real, square, and symmetric. The decomposition technique is attributed to Gauss. In matrix notation

$$[A]_{N\times N} = [U]^T_{N\times N} [D]_{N\times N} [U]_{N\times N} \qquad [1]$$

where

$$[U]_{N\times N} \begin{bmatrix} 1 & u_{12} & u_{13} & \cdots & u_{1N} \\ 0 & 1 & u_{23} & \cdots & u_{2N} \\ 0 & 0 & 1 & \cdots & u_{3N} \\ . & . & . & . & . \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \qquad [1a]$$

and

$$[D]_{N\times N} = \begin{bmatrix} d_{11} & 0 & 0 & \cdots & 0 \\ 0 & d_{22} & 0 & \cdots & 0 \\ 0 & 0 & d_{33} & \cdots & 0 \\ . & . & . & & . \\ 0 & 0 & 0 & \cdots & d_{NN} \end{bmatrix} .$$

N is the size of the matrices (square):

DESCRIPTION OF TECHNIQUE

To determine the elements of [U] and [D], first consider the multiplication of two matrices.

$$[A] = [C] [U] \qquad [2]$$

where [C] is a lower triangular matrix of the form

$$[C] = \begin{bmatrix} c_{11} & 0 & \cdots & 0 \\ c_{21} & c_{22} & \cdots & 0 \\ \cdot & \cdot & \cdots & \cdot \\ c_{N1} & c_{N2} & \cdots & c_{NN} \end{bmatrix}$$

and [U] has been previously defined.

Because [A] is symmetric,

$$[A] = [A]^T$$

$$= [U]^T [C]^T$$

$$= [U]^T \begin{bmatrix} c_{11} & 0 & \cdots & 0 \\ 0 & c_{22} & \cdots & 0 \\ \cdot & \cdot & \cdots & \cdot \\ 0 & 0 & \cdots & c_{NN} \end{bmatrix} \begin{bmatrix} 1 & \frac{c_{21}}{c_{11}} & \cdots & \frac{c_{N1}}{c_{11}} \\ 0 & 1 & \cdots & \frac{c_{N2}}{c_{22}} \\ \cdot & \cdot & \cdots & \cdot \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Due to the uniqueness of the factorization of [A] into the product of the two triangular matrices, one of which has ones on the diagonal, it follows that

$$\begin{bmatrix} 1 & \frac{c_{21}}{c_{11}} & \cdots & \frac{c_{N1}}{c_{11}} \\ 0 & 1 & \cdots & \frac{c_{N2}}{c_{22}} \\ \cdot & \cdot & \cdots & \cdot \\ 0 & 0 & & 1 \end{bmatrix} = [U].$$

By defining

$$[D] = \begin{bmatrix} c_{11} & 0 & \dots & 0 \\ 0 & c_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & c_{NN} \end{bmatrix},$$

Equation [1] is thus derived from Equation [2].

By performing the matrix products indicated by Equation [1], the elements in [A] are obtained generally as

$$a_{ij} = u_{1i}\, d_{11}\, u_{1j} + u_{2i}\, d_{22}\, u_{2j} + \dots + d_{ii}\, u_{ij}$$

and

$$a_{ii} = d_{11}\, u_{1i}^2 + d_{22}\, u_{2i}^2 + \dots + d_{ii}$$

From these two equations we calculate

$$d_{11} = a_{11}$$

$$u_{1j} = a_{1j}/d_{11} \qquad (j = 2, N)$$

$$d_{ii} = a_{ii} - \sum_{k=1}^{i-1} d_{kk}\, u_{ki}^2 \qquad (i = 2, N)$$

$$u_{ij} = \left(a_{ij} - \sum_{k=1}^{i-1} u_{ki}\, d_{kk}\, u_{kj}\right) \Big/ d_{ii} \qquad (i < j)$$

$$u_{ii} = 1.0 \qquad (i = 1, N)$$

$$u_{ij} = 0.0 \qquad (i > j)$$

Because of the shape of [A] and the corresponding shape and density of [U], this decomposition is performed using banded programming logic.

For the banded programming logic employed in YDCOM2, only half of [A] is stored. Rows of [A] are stored which include only the diagonal element through the last non-zero in the row. Output matrices [U] and [D] are written on NUTU and NUTD in FORMA sparse notation.

MISCELLANEOUS

The diagonal elements of [D] are the determinant ratios of [A] . The determinant of [A] is the product of these determinant ratios. That is,

$$|A| = \prod_{i=1}^{N} d_{ii}$$

REFERENCE

V. N. Faddeeve, "Computational Methods of Linear Algebra". Dover Publications, Inc., New York, 1959.

Subroutine YDCOM3 decomposes [A] a matrix in FORMA sparse notation to form an upper triangular matrix with ones on the diagonal [U] and a diagonal matrix [D] such that $[A] = [U]^T [D] [U]$. [A] must be real, square, and symmetric. The decomposition technique is attributed to Gauss. In matrix notation

$$[A]_{N\times N} = [U]^T_{N\times N} [D]_{N\times N} [U]_{N\times N} \qquad [1]$$

where

$$[U]_{N\times N} = \begin{bmatrix} 1 & u_{12} & u_{13} & \cdots & u_{1N} \\ 0 & 1 & u_{23} & \cdots & u_{2N} \\ 0 & 0 & 1 & \cdots & u_{3N} \\ . & . & . & & . \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \qquad [1a]$$

and

$$[D]_{N\times N} = \begin{bmatrix} d_{11} & 0 & 0 & \cdots & 0 \\ 0 & d_{22} & 0 & \cdots & 0 \\ 0 & 0 & d_{33} & \cdots & 0 \\ . & . & . & . & . \\ 0 & 0 & 0 & \cdots & d_{NN} \end{bmatrix} .$$

N is the size of the matrices (square):

DESCRIPTION OF TECHNIQUE

To determine the elements of [U] and [D], first consider the multiplication of two matrices.

$$[A] = [C] [U] \qquad [2]$$

where [C] is a lower triangular matrix of the form

$$[C] = \begin{bmatrix} c_{11} & 0 & \cdots & 0 \\ c_{21} & c_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ c_{N1} & c_{N2} & \cdots & c_{NN} \end{bmatrix}$$

and [U] has been previously defined.

Because [A] is symmetric,

$$[A] = [A]^T$$

$$= [U]^T [C]^T$$

$$= [U]^T \begin{bmatrix} c_{11} & 0 & \cdots & 0 \\ 0 & c_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & c_{NN} \end{bmatrix} \begin{bmatrix} 1 & \frac{c_{21}}{c_{11}} & \cdots & \frac{c_{N1}}{c_{11}} \\ 0 & 1 & \cdots & \frac{c_{N2}}{c_{22}} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Due to the uniqueness of the factorization of [A] into the product of the two triangular matrices, one of which has ones on the diagonal, it follows that

$$\begin{bmatrix} 1 & \frac{c_{21}}{c_{11}} & \cdots & \frac{c_{N1}}{c_{11}} \\ 0 & 1 & \cdots & \frac{c_{N2}}{c_{22}} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & & 1 \end{bmatrix} = [U].$$

By defining

$$[D] = \begin{bmatrix} c_{11} & 0 & \cdots & 0 \\ 0 & c_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & c_{NN} \end{bmatrix}$$

Equation [1] is thus derived from Equation [2].

By performing the matrix products indicated by Equation [1], the elements in [A] are obtained generally as

$$a_{ij} = u_{1i}\, d_{11}\, u_{1j} + u_{2i}\, d_{22}\, u_{2j} + \ldots + d_{ii}\, u_{ij}$$

and

$$a_{ii} = d_{11}\, u_{1i}^2 + d_{22}\, u_{2i}^2 + \ldots + d_{ii}$$

From these two equations we calculate

$$d_{11} = a_{11}$$

$$u_{1j} = a_{1j}/d_{11} \qquad (j = 2, N)$$

$$d_{ii} = a_{ii} - \sum_{k=1}^{i-1} d_{kk}\, u_{ki}^2 \qquad (i = 2, N)$$

$$u_{ij} = \left(a_{ij} - \sum_{k-1}^{i-1} u_{ki}\, d_{kk}\, u_{kj}\right) \Big/ d_{ii} \qquad (i < j)$$

$$u_{ii} = 1.0 \qquad (i = 1, N)$$

$$u_{ij} = 0.0 \qquad (i > j)$$

Because of the shape of [A] and the corresponding shape and density of [U], this decomposition is performed using banded programming logic.

For the banded programming logic employed in YDCOM3, only half of [A] is stored. Columns of [A] are stored which include only the top non-zero down to the diagonal element of the column. Output matrices [U] and [D] are written on NUTU and NUTD in FORMA sparse notation.

MISCELLANEOUS

The diagonal elements of [D] are the determinant ratios of [A] . The determinant of [A] is the product of these determinant ratios. That is,

$$|A| = \prod_{i=1}^{N} d_{ii}$$

REFERENCE

V. N. Faddeeve, Computational Methods of Linear Algebra. Dover Publications, Inc., New York, 1959.

Subroutine YDIAG places the elements from a vector (row or column matrix $\{A\}$ into the corresponding diagonal locations of a square matrix $[Z]$. $\{A\}$ and $[Z]$ are stored on NUTA and NUTZ in FORMA sparse notation. In subscript notation,

$$z_{ii} = a_i \qquad (i = 1, N)$$

$$z_{ij} = 0. \qquad (i \neq j)$$

The z_{ij} are not stored.

In matrix notation

$$[Z]_{NxN} = \begin{bmatrix} a_1 & & & \\ & a_2 & & \\ & & \ddots & \\ & & & a_N \end{bmatrix}$$

where N is the size of $[Z]$ (square), and the length of $\{A\}$.

EXAMPLE

Consider input of

$$\{A\}_{Nx1} = \begin{bmatrix} 1. \\ 2. \\ 3. \end{bmatrix}$$

where N = 3.

The result of this subroutine will give

$$[Z]_{3x3} = \begin{bmatrix} 1. & 0. & 0. \\ 0. & 2. & 0. \\ 0. & 0. & 3. \end{bmatrix}.$$

DESCRIPTION OF TECHNIQUE

Vector $\{A\}$ is read from NUTA, one partition at a time. The location

numbers for the non-zero elements of $\{A\}$ are modified to reflect diagonal elements. These partitions are then written on NUTZ to form $[Z]$.

Subroutine YDISA removes (disassembles) a matrix [Z] from matrix [A] starting at a designated row, column location (IRA, JCA, respectively) in [A] . The [Z] matrix must be within the row, column limits of [A] . Matrices [A] and [Z] are stored in FORMA sparse notation. In subscript notation,

$$z_{ij} = a_{k\ell} \qquad \begin{pmatrix} i = 1, \text{ NRZ} \\ j = 1, \text{ NCZ} \end{pmatrix}$$

where

$k = i + IRA - 1$

$\ell = j + JCA - 1$

NRZ is the number of rows of [Z] , and

NCZ is the number of columns of [Z] .

EXAMPLE

Consider a matrix defined as

$$[A]_{3x4} = \begin{bmatrix} 1. & 0. & 0. & 0. \\ 3. & 4. & 5. & 0. \\ 6. & 7. & 8. & 0. \end{bmatrix}$$

Matrix $[Z]_{2x3}$ is to be obtained from [A] starting at the 2,1 location (IRA = 2, JCA = 1) of [A] . The result of this operation will be

$$[Z]_{2x3} = \begin{bmatrix} 3. & 4. & 5. \\ 6. & 7. & 8. \end{bmatrix}$$

The matrix [A] remains as originally defined.

DESCRIPTION OF TECHNIQUE

Partitions of [A] are read from NUTA into the first quarters of the V and LV workspaces. The element locations of [A] are searched for

locations in the range of [Z] . Any elements found in this range are moved to the second quarter of V and location numbers for these elements are formed in the second quarter of LV. The data in the second quarters of V and LV become partitions of [Z] which is stored on NUTZ.

Subroutine YSTOD converts a dense matrix in core to a FORMA sparse matrix written on a peripheral device. Only the nonzero elements of dense matrix [A] are stored in sparse matrix [Z]. For each a_{ij}, a location number is formed as

LV(k) = 100000 * i + j and

V(k) = a_{ij}.

LV(k) and V(k) for k = 1, NNZA are then written on the peripheral device, where NNZA is equal to the number of nonzeroes in [A].

Subroutine YIN is used to transfer real data from a computer peripheral device into core. Utilization of this subroutine enables a programmer to easily take advantage of computer-dependent input routines which may be more efficient than FORTRAN read statements. Data transmitted using subroutine YOUT may be retrieved using subroutine YIN.

Subroutine YINI is used to transfer integer data from a computer peripheral device into core. Utilization of this subroutine enables a programmer to easily take advantage of computer dependent input routines which may be more efficient than FORTRAN read statements. Data transmitted using subroutine YOUTI may be retrieved using subroutine YINI.

The logic in FORMA sparse subroutines depends on the fact that the locations of elements in the matrices are in order by row and column.

Subroutine YLORD orders all nonzero matrix element locations and the elements. Ordering is defined as follows: all elements of a row are ordered by column from left to right; all rows are then ordered from the first through the last.

DESCRIPTION OF TECHNIQUE

Because no partition of a FORMA sparse matrix exceeds KV/4 in length, four partitions of [A] are brought into core together and the element locations and elements of all four are ordered simultaneously. The method of ordering employed in this process is derived from the method of R. C. Singleton.* All partitions of [A] are ordered, four at a time, using this process. The ordered partitions then are merged together to form the completely ordered matrix [Z].

---

*R. C. Singleton: *An Efficient Algorithm for Sorting with Minimal STorage.* Research Memorandum, SRI Project 387531-132, September 1968.

Subroutine YMOD2A calculates the mode shapes and natural frequencies of a structural model described as

$$[\hat{K}]\ [\phi] = [M]\ [\phi]\ [\Omega^2] \qquad [1]$$

where

$$[\hat{K}] = [K] - \lambda_S\ [M]$$

and

$$[\Omega^2] = [\omega^2] - \lambda_S\ [I].$$

If $\lambda_S = 0$, then $[\hat{K}] = [K]$ and $[\Omega^2] = [\omega^2]$, and [1] becomes the more familiar eigen problem

$$[K]\ [\phi] = [M]\ [\phi]\ [\omega^2].$$

Subroutine YMOD2A uses the composite structure iterative Rayleigh-Ritz method of Dr. John Admire.

If initial Rayleigh-Ritz coordinate displacements [q] are available, they may be input to the subroutine. Otherwise random numbers will be used.

Input matrices [K], [M], and [q] and output matrix [ϕ] are in FORMA sparse notation.

Subroutine YMULT calculates the product of two matrices stored in FORMA sparse notation. In matrix notation,

$$[Z]_{NRA \times NCB} = [A]_{NRA \times NRB} \; [B]_{NRB \times NCB}$$

where

$$z_{ij} = \sum_{k=1}^{NRB} a_{ik} \, b_{kj} \qquad \begin{pmatrix} i = 1, \; NRA \\ j = 1, \; NCB \end{pmatrix}$$

NRA is the number of rows of [A] and [Z] . NRB is the number of rows of [B] and the number of columns of [A] . NCB is the number of columns of [B] and [Z] . The number of columns of [A] must be equal to the number of rows of [B] .

Theorem: Multiplication of matrices is not commutative in general. That is,

$$[A] \; [B] \neq [B] \; [A]$$

for any values of a_{ij} and b_{ij}

Theorem: Multiplication of matrices is associative. That is,

$$[A] \; ([B] \; [C]) = ([A] \; [B]) \; [C] .$$

Theorem: Multiplication of matrices is distributive. That is,

$$[A] \; ([B] + [C]) = [A] \; [B] + [A] \; [C] .$$

EXAMPLE

Consider input of $[A]_{1x2} = [1. \quad 2.]$ and $[B]_{2x3} = \begin{bmatrix} 7. & -8. & 9. \\ 10. & 11. & 12. \end{bmatrix}$

The reader can easily verify the output to be

$$[Z]_{1x3} = [1. \quad 2.] \begin{bmatrix} 7. & -8. & 9. \\ 10. & 11. & 12. \end{bmatrix}$$

$$= [27. \quad 14. \quad 33.] .$$

DESCRIPTION OF TECHNIQUE

Partitions of [A] are read from NUTA into the first quarters of the V and LV workspaces. Partitions of [B] are read into the second quarters of the workspaces and are premultiplied by a row of [A] . A dense row of [Z] is formed in the third quarter of V based on the row of [A] multiplied by all of matrix [B] . When the dense row of [Z] is completely formed, the non-zeros are transferred to the fourth quarter of V and an appropriate vector of location numbers is formed in the fourth quarter of LV. The data generated in the fourth quarters of V and LV become partitions of [Z] and are written on NUTZ.

Limitation:

One fourth of the dimension size of V and LV, (KV/4), must exceed or be equal to NCB and NRB.

Subroutine YMULT1 calculates the special product of two FORMA sparse notation matrices. In matrix notation,

$$[Z]_{NRA\times NCB} = [A]_{NRA\times NRA}\ [B]_{NRA\times NCB}$$

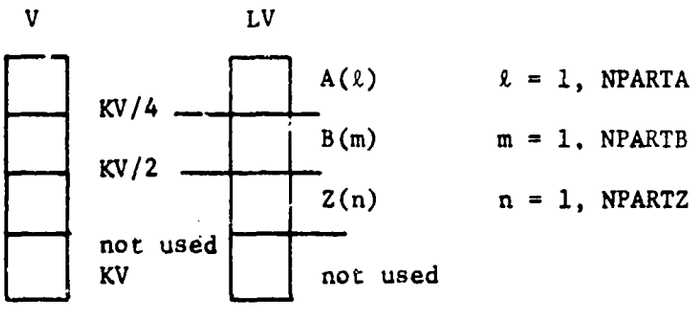
$$z_{ij} = \sum_{k=1}^{NRA} a_{ik}\, b_{kj} \qquad \begin{pmatrix} i = 1, NRA \\ j = 1, NCB \end{pmatrix}$$

NRA is the number of rows of [A], [B], and [Z] where [A] is square. NCB is the number of columns of [B] and [Z]. NRA is assumed to be greater than NCB.

[B] and therefore [Z] are assumed to have no elements equal to zero.

[A], [B], and [Z] are stored on utility tapes as row partitioned matrices in sparse notation.

Partitions of the matrices are operated on in central memory as follows (ℓ, m, and n refer to partition number):



where V and LV are dimensioned work spaces and KV is the dimension size of V and LV.

The number of nonzero terms in each partition of [Z], NNZPZ, is $\left(\frac{KV}{4}\right)\Big/NCB * NCB$.

The number of rows in each partition of [Z], NRPZ, is NNZPZ/NCB.

The number of partitions of [Z], NPARTZ (=NPARTB), is (NRA-1)/NRPZ + 1.

$$\begin{bmatrix} A_1 \\ \cdot\cdot\cdot\cdot\cdot\cdot\cdot\cdot \\ \cdot\cdot\cdot\cdot\cdot\cdot\cdot\cdot \\ A_{nparta} \end{bmatrix} \begin{bmatrix} \\ B \\ \\ \end{bmatrix} = \begin{bmatrix} Z_1 \\ \cdot\cdot\cdot\cdot \\ \cdot\cdot\cdot\cdot \\ Z_{npartz} \end{bmatrix} \begin{matrix} \text{NFRPZ(1)} = 1 \\ \text{NLRPZ(1)} \\ \text{NFRPZ(NPARTZ)} \\ \text{NLRPZ(NPARTZ)} = \text{NRA} \end{matrix}$$

$[Z_\ell] = [A_{ik}]\ [B_{kj}]$

$i = \text{NFRPZ}(\ell), \text{NLRPZ}(\ell)$
$j = 1, \text{NCB}$
$k = 1, \text{NRA}$
$\ell = 1, \text{NPARTZ}$

Where

NFRPZ is the number of the first row in a partition of Z, and

LLRPZ is the number of the last row in a partition of Z.

Subroutine YMULT2 calculates the special product of two FORMA sparse notation matrices, in matrix notation,

$$[Z]_{NCB\times NCB} = [A]^{T}_{NCB\times NRB}\ [B]_{NRB\times NCB}$$

$$z_{ij} = \sum_{k=1}^{NRB} a_{ki}\, b_{kj} \qquad \begin{pmatrix} i = 1, NCB \\ j = 1, NCB \end{pmatrix}$$

NRB is the number of rows of [A] and [B] and the number of columns of $[A]^{T}$. NCB is the number of rows of $[A]^{T}$ and [Z] and the number of columns of [A], [B], and [Z]. [A], [B], and [Z] are assumed to have no elements equal to zero and [Z] is symmetric.

Statements from FORMA subroutine ATXBB2 are used in this subroutine.

Subroutine YMULT4 calculates the special product of a FORMA sparse notation matrix postmultiplied by a square matrix stored in core. In matrix notation,

$$[Z]_{NRA\times NCB} = [A]_{NRA\times NCB}\ [B]_{NCB\times NCB}$$

$$z_{ij} = \sum_{k=1}^{NCB} a_{ik}\, b_{kj} \qquad \begin{pmatrix} i = 1, NRA \\ j = 1, NCB \end{pmatrix} ,$$

where

NRA is the number of rows of [A] and [Z], and NCB is the number of rows of [B] and the number of columns of [A], [B] and [Z]. NRA is greater than NCB.

[A] and [Z] are assumed to have no elements equal to zero. They are stored on utility tapes in FORMA sparse notation.

Subroutine YMULTA calculates the product of two matrices stored in FORMA sparse notation. In matrix notation

$$[Z]_{NRA \times NCB} = [A]_{NRA \times NRB} \; [B]_{NRB \times NCB}$$

where

$$z_{ij} = \sum_{k=1}^{NRB} a_{ik} \, b_{kj} \qquad \begin{pmatrix} i = 1, \; NRA \\ j = 1, \; NCB \end{pmatrix}$$

NRA is the number of rows of [A] and [Z] . NRB is the number of rows of [B] and the number of columns of [A] . NCB is the number of columns of [B] and [Z] . The number of columns of [A] must be equal to the number of rows of [B] .

Theorem: Multiplication of matrices is not commutative in general. That is,

$$[A] \; [B] \neq [B] \; [A]$$

for any values of a_{ij} and b_{ij}.

Theorem: Multiplication of matrices is associative. That is,

$$[A] \; ([B] \; [C]) = ([A] \; [B]) \; [C] \; .$$

Theorem: Multiplication of matrices is distributive. That is,

$$[A] \; ([B] + [C]) = [A] \; [B] + [A] \; [C] \; .$$

EXAMPLE

Consider input of $[A]_{1x2} = [1. \quad 2.]$ and $[B]_{2x3} = \begin{bmatrix} 7. & -8. & 9. \\ 10. & 11. & 12. \end{bmatrix}$

The reader can easily verify the output to be

$$[Z]_{1x3} = [1. \quad 2.] \begin{bmatrix} 7. & -8. & 9. \\ 10. & 11. & 12. \end{bmatrix}$$

$$= [27. \quad 14. \quad 33.] \; .$$

DESCRIPTION OF TECHNIQUE

Partitions of [A] are read from NUTA into the first quarters of the V and LV workspaces. Partitions of [B] are read into the second quarter of the workspaces and are premultiplied by a row of [A] . A dense row of [Z]is formed in the third quarter of V based on the row of [A] multiplied by all of matrix [B] . When the dense row of [Z] is completely formed, the non-zeros are transferred to the fourth quarter of V and an appropriate vector of location numbers is formed in the fourth quarter of LV. The data generated in the fourth quarters of V and LV become partitions of [Z] and are written on NUTAZ. Matrix [A] is, therefore, replaced by matrix [Z] in this subroutine.

Limitation:

One fourth of the dimension size of V and LV, (KV/4), must exceed or be equal to NCA and NCB.

Subroutine YMULTB calculates the product of two matrices stored in FORMA sparse notation. In matrix notation

$$[Z]_{NRAxNCB} = [A]_{NRAxNRB} \; [B]_{NRBxNCB}$$

where

$$z_{ij} = \sum_{k=1}^{NRB} a_{ik} \, b_{kj} \qquad \begin{pmatrix} i = 1, \; NRA \\ j = 1, \; NCB \end{pmatrix}$$

NRA is the number of rows of [A] and [Z] . NRB is the number of rows of [B] and the number of columns of [A] . NCB is the number of columns of [B] and [Z] . The number of columns of [A] must be equal to the number of rows of [B] .

Theorem: Multiplication of matrices is not commutative in general. That is,

$$[A] \; [B] \neq [B] \; [A]$$

for any values of a_{ij} and b_{ij}.

Theorem: Multiplication of matrices is associative. That is,

$$[A] \; ([B] \; [C]) = ([A] \; [B]) \; [C] \; .$$

Theorem: Multiplication of matrices is distributive. That is,

$$[A] \; ([B] + [C]) = [A] \; [B] + [A] \; [C] \; .$$

EXAMPLE

Consider input of $[A]_{1x2} = [1. \quad 2.]$ and $[B]_{2x3} = \begin{bmatrix} 7. & -8. & 9. \\ 10. & 11. & 12. \end{bmatrix}$

The reader can easily verify the output to be

$$[Z]_{1x3} = [1. \quad 2.] \begin{bmatrix} 7. & -8. & 9. \\ 10. & 11. & 12. \end{bmatrix}$$

$$= [27. \quad 14. \quad 33.]$$

DESCRIPTION OF TECHNIQUE

Partitions of [A] are read from NUTA into the first quarters of the V and LV workspaces. Partitions of [B] are read into the second quarters of the workspaces and are premultiplied by a row of [A] . A dense row of [Z] is formed in the third quarter of V based on the row of [A] multiplied by all of matrix [B] . When the dense row of [Z] is completely formed, the non-zeros are transferred to the fourth quarter of V and an appropriate vector of location numbers is formed in the fourth quarter of LV. The data generated in the fourth quarters of V and LV become partitions of [Z] and are written on NUTBZ. Matrix [B] is therefore replaced by matrix [Z] in this subroutine.

Limitation:

One fourth of the dimension size of V and LV, (KV/4), must exceed or be equal to NCA and NCB.

Subroutine YNOZER scans each partition of a FORMA sparse matrix and deletes any terms equal to zero. The logic of sparse matrix subroutines becomes inefficient if matrix terms equal to zero are stored and operated on. If prior matrix operations have computed values in [A] such that any a_{ij} are equal to zero, these a_{ij} are deleted.

Subroutine YOUT is used to transfer real data from computer central memory to a peripheral device. Utilization of this subroutine enables a programmer to easily take advantage of computer-dependent output routines which may be more efficient than FORTRAN write statements. Data transmitted using subroutine YOUT may be retrieved using subroutine YIN.

Subroutine YOUTI is used to transfer integer data from computer central memory to a peripheral device. Utilization of this subroutine enables a programmer to easily take advantage of computer dependent output routines which may be more efficient than FORTRAN write statements. Data transmitted using subroutine YOUTI may be retrieved using subroutine YINI.

Subroutine YPART partitions newly generated matrices and repartitions existing matrices in which the partition length is not compatible with the work space dimension (KV) of the current computer program. Partitioning ensures that no matrix row spans two partitions and the length of all partitions is less than or equal to one fourth of KV.

Subroutine YPNCHO punches a matrix [A] onto cards using octal format to eliminate round-off error. Matrix [A] is stored on NUTA in FORMA sparse notation. Octal representation of the matrix elements is used because it gives an exact replica of the binary number used by a digital computer. A decimal representation will not give an exact replica. The matrix on punched cards is to be used only as an emergency backup for the matrix written on a storage tape.

This matrix on cards is compatible with the input form for Subroutine YREADO. A group of up to three consecutive elements from a row of the matrix are punched on each card. If all of the elements of a group are zero, punching of this card is suppressed.

The first card punched contains the matrix name (in card columns 1-6), the matrix row size (in card columns 7-10), the matrix column size (in card columns 11-15), and the matrix shape (in card columns 16-21). This is followed by the matrix data. On any card of the matrix data, the first integer number (card columns 1-5) is the row number of the matrix elements on that card. The second integer number (card columns 6-10) is the column number of the matrix element in the first data field (card columns 14-25). The next group contains octal numbers (up to three numbers in card columns 14-25, 29-40, 44-55) that are the values of the matrix elements. This group of matrix elements is given in consecutive column order. The last card punched contains ten zeroes in card columns 1-10 to indicate the end of the matrix.

Subroutine YPUNCH punches a matrix [A] of real numbers (a Fortran term for numbers with a decimal point) onto cards. Matrix [A] is stored on NUTA in FORMA sparse notation. This matrix on cards is compatible with the input form for Subroutine YREAD. A group of up to four consecutive elements from a row of the matrix are punched on each card. If all of the elements of a group are zero, punching of this card is suppressed.

The first card punched contains the matrix name (in card columns 1-6), the matrix row size (in card columns 7-10), the matrix column size (in card columns 11-15), and the matrix shape (in card columns 16-21). This is followed by the matrix data. On any card of matrix data, the first integer number (card columns 1-5) is the row number of the matrix elements on that card. The second integer number (card columns 6-10) is the column number of the matrix element in the first data field (card columns 11-27). The next group contains real numbers (up to four numbers in card columns 11-27, 28-44, 45-61, 62-78) that are the values of the matrix elements. This group of matrix elements is given in consecutive column order. The last card punched contains ten zeroes in card columns 1-10 to indicate the end of the matrix.

Subroutine YREAD reads a matrix of real numbers (a FORTRAN term for numbers with a decimal point) from either cards or tape and stores the matrix on a utility tape in FORMA sparse notation. The matrix is then printed so that these input data are recorded with the answers of a run. A print suppression option is available for a matrix read from tape. On option, the matrix read from either cards or tape may be written on a tape (by Subroutine YWTAPE).

The first data card read by Subroutine YREAD contains information to indicate whether cards or tape will be used. The information entered on this card (and subsequent cards for card input) is given below. The format is identical to that used by Subroutine READ.

Card Data Input Form

Required entries are denoted by an * symbol below. Any other entry is optional

	Card Columns	Format Type (1)	Entry
First Card	1-6	A	*Matrix Name. Will appear in printout.
	7-10	I	*Matrix Row Size.
	11-15	I	*Matrix Column Size.
	16-69	A	Any remarks to further identify the input matrix.
	16-20	A	The word upper (or lower) must be specified if only the upper (or lower) half of a symmetric matrix is on the punched cards.
Write-Tape Options	72		$. Only if the Write-Tape is to be initialized by Subroutine INTAPE. The Write-Tape identification will be from card columns 73-78.
	72	or	Anything other than $ is the Write-Tape is not to be initialized.
	73-78	A	The Write-Tape identification. (e.g., T1234). Use with $ in card column 72.
	73-78	or	REWIND. The Write-Tape will be rewound before being used.

	Card Columns	Format Type (1)	Entry
(cont'd)	73-76	or	LIST. The Write-Tape will be listed by Subroutine LTAPE after the matrix has been written on the Write-Tape.
	73-78	or	Anything else will be ignored.
	79-80	I	The Write-Tape Number (e.g., 21).
		or	Blank if the matrix is not to be written on tape.
Middle Cards	1-5	I	*Row Number of matrix elements on card.
	6-10	I	*Column Number of matrix element in first data field.
	11-27	E	*First data field with matrix elements. (2)
	28-44	E	*Second data field with matrix elements. (2)
	45-61	E	*Third data field with matrix elements. (2)
	62-78	E	*Fourth data field with matrix elements. (2)
Last Card	1-10	I	*Ten zeroes.

Note (1) Format Type A allows any keypunch symbol. Format Type I allows only integer numbers right justified in the field. Format Type E allows only real numbers (a FORTRAN term for numbers with a decimal point) anywhere in the field.

Note (2) Only nonzero elements need be entered.

As an example of card input to Subroutine YREAD consider the following matrix:

$$[A1*C]_{3\times 6} = \begin{bmatrix} 1. & 0. & 3. & 0. & 6. & 5. \\ 0. & 2. & 4. & 0. & 0. & 0. \\ 0. & 7. & 0. & 0. & 0. & 0. \end{bmatrix}.$$

This matrix is also to be written on tape number 21 that is to be initialized and identified as T4334. Figure 1 demonstrates how this information could be written on a coding form to facilitate keypunching to cards.

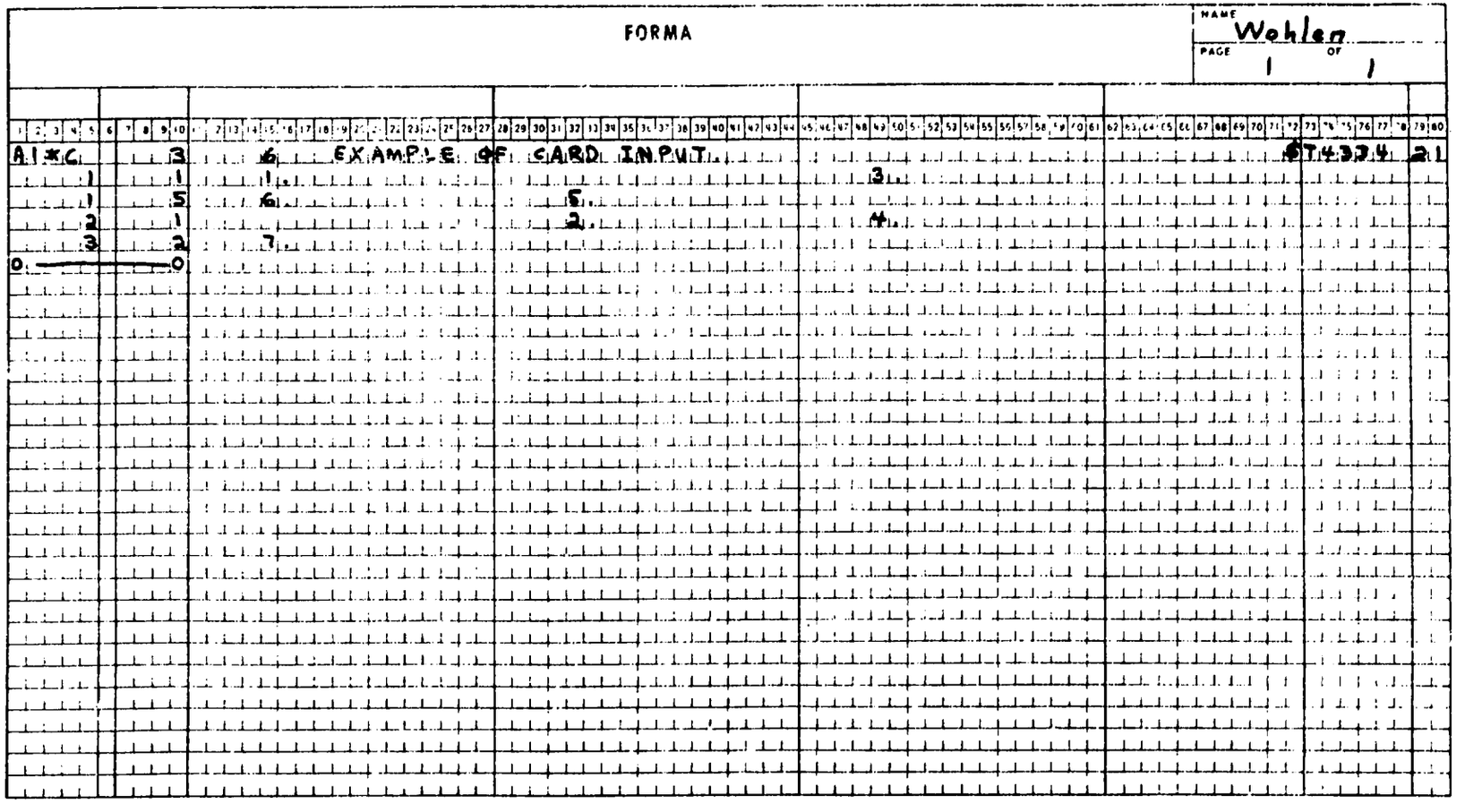


Figure 1 Example of Card Input for Subroutine YREAD

Tape Data Input Form

Required entries are denoted with an * symbol below. Any other entry is optional. Only one card is used for each matrix read.

	Card Columns	Format Type (1)	Entry
One Card	1-7	A	*Name of matrix to be read from the Read-Tape.
	10		Zero. The Read-Tape will move forward from its present position and search to the end of the tape. If the matrix is not found upon the first end-of-tape encounter, the tape will automatically rewind and make one more pass. If it is not found on the second end-of-tape encounter, an error message will be printed and the program will stop.
	7-10	I or	Minus the location number of matrix on the Read-Tape. Tape will be positioned at the beginning of the location specified and then continue as described above for a zero in column 10.
	11-15	I	*The Read-Tape Number (e.g., 11). If positive, the matrix read will be printed in the output. If negative, the matrix read will not be printed in the output.
	16-21	A	*Run number of matrix to be read from the Read-Tape.
	22-27		REWIND. The Read-Tape will be rewound before being used.
	22-25	or	LIST. The Read-Tape will be listed by Subroutine LTAPE.
	22-27	or	Anything else will be considered as part of the remarks described below.
	28-69	A	Any remarks to further identify the input matrix.

Card Columns	Format Type (1)	Entry
72		$. Only if the Write-Tape is to be initialized by Subroutine INTAPE. The Write-Tape identification will be from card columns 73-78.
72	or	Anything other than $ if the Write-Tape is not to be initialized.
73-78	A	The Write-Tape identification (e.g., T1234). Use with $ in card column 72.
73-78	or	REWIND. The Write-Tape will be rewound before being used.
73-76	or	LIST. The Write-Tape will be listed by Subroutine LTAPE after the matrix has been written on the Write-Tape.
73-78	or	Anything else will be ignored.
79-80	I	The Write-Tape Number (e.g., 21).
	or	Blank if the matrix is not to be written on tape.

Note (1) Format Type A allows any keypunch symbol. Format Type I allows only integer numbers right justified in the field.

As examples of tape input to Subroutine YREAD consider:

Example 1. A matrix named AB2 with run number of RUN-46 is to be read from tape number 11 into the computer and printed. This matrix is also to be written on tape number 22 that is to be initialized and identified as T4321.

Example 2. A matrix named XYZ4 with run number of TKD is on tape number 13 twice. The first time is at location 29 and the second time is at location 54. It is desired to read the second matrix.

Figure 2 demonstrates how these two examples would be written on a coding form to facilitiate keypunching to cards.

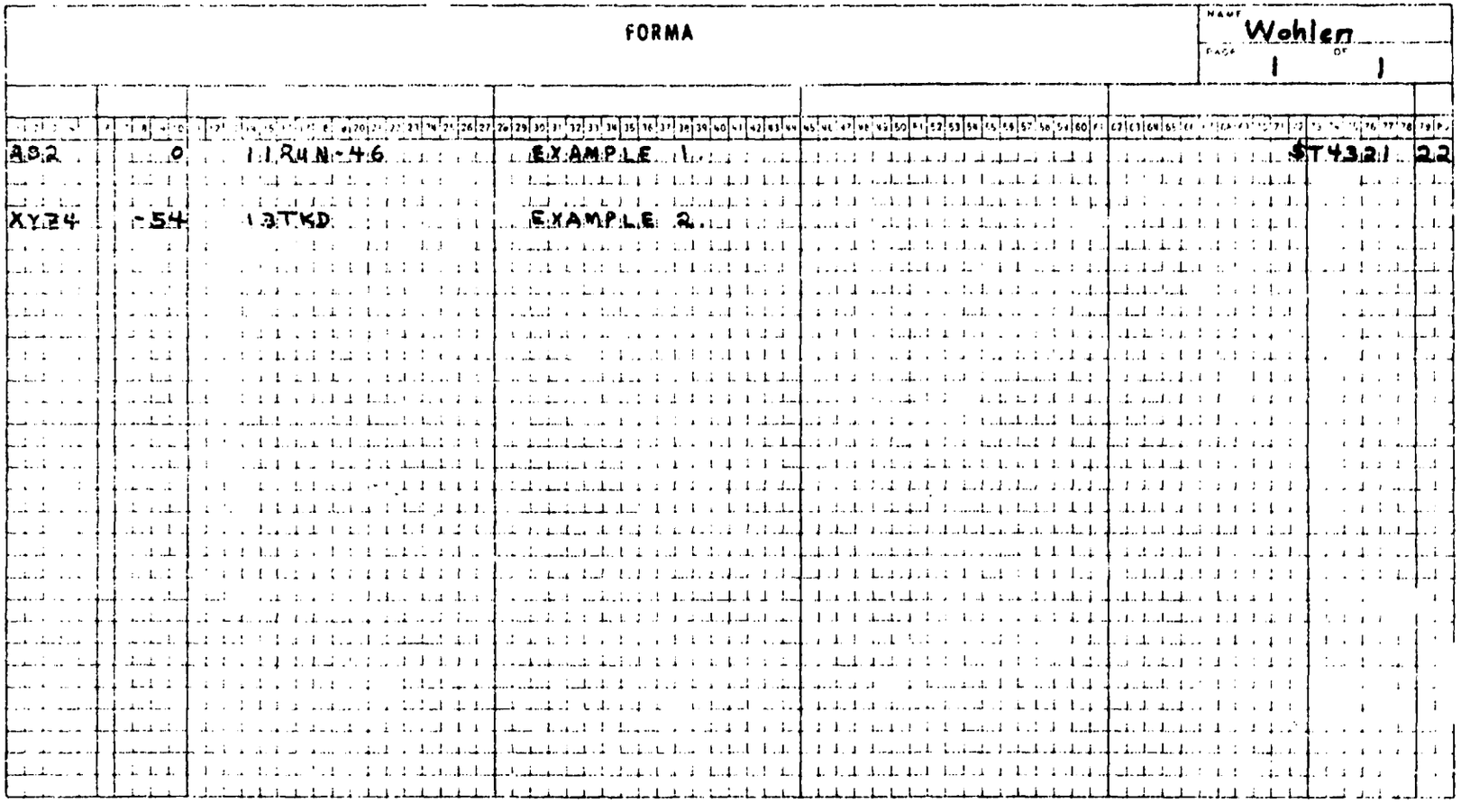


Figure 2 Examples of Tape Input for Subroutine YREAD

Subroutine YREADO reads a matrix [A] from octal numbers on cards. Matrix [A] is then stored on NUTA in FORMA sparse notation. The matrix is printed side by side in both octal and decimal so that these input data are recorded with the answers of a run.

The primary purpose of Subroutine YREADO is to read a matrix from punched cards without round-off error. The cards are punched by Subroutine YPNCHO. Octal representation of the matrix elements is used because it gives an exact replica of the binary number used by a digital computer. A decimal representation will not give an exact replica. The matrix on punched cards is to be used only as an emergency backup for the matrix written on a storage tape.

Because of the emergency backup nature of input data to this Subroutine YREADO, only cards are read. No tape reading or writing options are available.

The information entered on the data cards is given below. Required entries are denoted by an * symbol. Any other entry is optional.

	Card Columns	Format Type (1)	Entry
First Card	1-6	A	*Matrix Name. (Will appear in printout.)
	7-10	I	*Matrix Row Size.
	11-15	I	*Matrix Column Size.
	16-21	A	*Matrix Shape.
	22-69	A	Any remarks to further identify the input matrix.
Middle Cards	1-5	I	*Row Number of matrix elements on card.
	6-10	I	*Column Number of matrix element in first data field.
	14-25	O	*First data field with matrix elements. (2)
	29-40	O	*Second data field with matrix elements. (2)
	44-55	O	*Third data field with matrix elements. (2)
Last Card	1-10	I	*Ten Zerces.

Note (1) Format Type A allows any keypunch symbol.
Format Type I allows only integer numbers right justified in the field. Format Type O allows only octal numbers.

Note (2) Only non-zero elements need be entered.

No examples of input are given because data would not be keypunched for input to Subroutine YREADO but rather obtained from Subroutine YPNCHO.

Subroutine YREVAD rearranges (revises) the rows and columns of a matrix [A] , multiplies [A] by a scalar alpha, and adds the result to a previously defined matrix [Z in] . [A] , [Z in] , and [Z out] are stored on NUTA and NUTZ in FORMA sparse notation. The revision of [A] is specified by two vectors. The first vector {IVEC} gives the new row location of each row of [A] in [Z] . The second vector {JVEC} gives the new column location of each column of [A] in [Z] . The [Z] matrix must be defined before the use of this subroutine. For instance, if [Z] is to be originally defined as all zeros, Subroutine YZERO could be used. The YREVAD operation can be thought of in subscript notation as

$$z_{k\ell}\ (\text{out}) = z_{k\ell}\ (\text{in}) + \alpha\, a_{ij} \qquad \begin{pmatrix} i = 1, \text{NRA} \\ j = 1, \text{NCA} \end{pmatrix}$$

where

$k = \text{IVEC}(i)$,

$\ell = \text{JVEC}(j)$.

NRA is the number of rows of [A] , and NCA is the number of columns of [A] .

Values in {IVEC} and {JVEC} may be positive, negative or zero. A negative value changes the sign of the corresponding row or column of [A] in [Z] . [A]zero value omits the corresponding row or column of [A] from [Z]. The values are integer numbers.

This subroutine may be called repeatedly to form [Z] from the revision/addition of several [A] matrices.

An important use of Subroutine YREVAD is to revise and add the stiffness matrix of a structural component to the stiffness matrix of the complete structure to account for the difference in coordinate systems.

EXAMPLES

The first example to illustrate the YREVAD operation is as follows. Matrix [Z] has been previously defined as

$$[Z]_{4x5} = \begin{bmatrix} 1. & 0. & 0. & 0. & 0. \\ 0. & 2. & 0. & 0. & 0. \\ 0. & 0. & 3. & 4. & 5. \\ 0. & 0. & 0. & 0. & 6. \end{bmatrix}$$

Let [A] be defined as

$$[A]_{3x2} = \begin{bmatrix} 1. & 2. \\ 3. & 4. \\ 5. & 6. \end{bmatrix}$$

The first row of [A] is to be added to the third row of [Z] , the second row of [A] is to be omitted from [Z] , the third row of [A] is to be added to the first row of [Z] , with the sign of each element reversed.

$$\text{Thus } \{IVEC\}_{3x1} = \begin{bmatrix} 3 \\ 0 \\ -1 \end{bmatrix}$$

The first column of [A] is to be added to the second column of [Z] with the sign of each element reversed, the second column of [A] is to be added to the fifth column of [Z] .

$$\text{Thus } \{JVEC\}_{2x1} = \begin{bmatrix} -2 \\ 5 \end{bmatrix}$$

Then, assuming $\alpha = 1.0$ and placing $\{IVEC\}$ and $\{JVEC\}$ adjacent to [A] to aid in visualizing the revision of [A] , we have

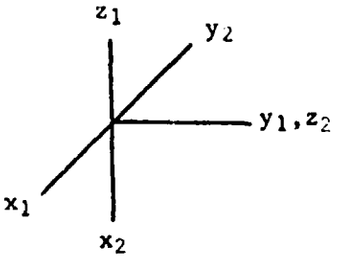
$$[Z]_{4x5} = \begin{bmatrix} 1. & 0. & 0. & 0. & 0. \\ 0. & 2. & 0. & 0. & 0. \\ 0. & 0. & 3. & 4. & 5. \\ 0. & 0. & 0. & 0. & 6. \end{bmatrix} + 1.0 \begin{bmatrix} 3 \\ 0 \\ -1 \end{bmatrix} \begin{bmatrix} -2 & 5 \\ 1. & 2. \\ 3. & 4. \\ 5. & 6. \end{bmatrix}$$

$$= \begin{bmatrix} 1. & 0. & 0. & 0. & 0. \\ 0. & 2. & 0. & 0. & 0. \\ 0. & 0. & 3. & 4. & 5. \\ 0. & 0. & 0. & 0. & 6. \end{bmatrix} + 1.0 \begin{bmatrix} 3 \\ 0 \\ -1 \end{bmatrix} \begin{bmatrix} 0. & -1. & 0. & 0. & 2. \\ 0. & -3. & 0. & 0. & 4. \\ 0. & -5. & 0. & 0. & 6. \end{bmatrix}$$

$$= \begin{bmatrix} 1. & 0. & 0. & 0. & 0. \\ 0. & 2. & 0. & 0. & 0. \\ 0. & 0. & 3. & 4. & 5. \\ 0. & 0. & 0. & 0. & 6. \end{bmatrix} + 1.0 \begin{bmatrix} 0. & 5. & 0. & 0. & -6. \\ 0. & 0. & 0. & 0. & 0. \\ 0. & -1. & 0. & 0. & 2. \\ 0. & 0. & 0. & 0. & 0. \end{bmatrix}$$

$$= \begin{bmatrix} 1. & 5. & 0. & 0. & -6. \\ 0. & 2. & 0. & 0. & 0. \\ 0. & -1. & 3. & 4. & 7. \\ 0. & 0. & 0. & 0. & 6. \end{bmatrix}.$$

A second example of the use of this subroutine demonstrates the coordinate transformation concept which is a very important application of **YREVAD**. It is desired to transform a stiffness matrix from an original coordinate system (subscript 1) to a final coordinate system (subscript 2) as shown in the sketch below.



For this example, the stiffness matrix in the original coordinate system is assumed to be

$$\begin{bmatrix} 1. & 2. & 4. \\ 2. & 3. & 5. \\ 4. & 5. & 6. \end{bmatrix}.$$

From inspection of the coordinate system axes in the above sketch,

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} -z_1 \\ -x_1 \\ y_1 \end{bmatrix}.$$

To obtain the vector on the right hand side of the equation:

x_1 of the original coordinate system is to be the second row of the final coordinate system with the sign reversed.

y_1 of the original coordinate system is to be the third row of the final coordinate system.

z_1 of the original coordinate system is to be the first row of the final coordinate system with the sign reversed.

The $\{IVEC\}$ to accomplish this change is

$$\{IVEC\}_{3x1} = \begin{bmatrix} -2 \\ 3 \\ -1 \end{bmatrix}.$$

Apply this $\{IVEC\}$ as the $\{IVEC\}$ and $\{JVEC\}$ to the original stiffness matrix to obtain the final stiffness matrix. This application is analogous to the change in coordinates made in the triple matrix product procedure.

$$\begin{bmatrix} -2 \\ 3 \\ -1 \end{bmatrix} \begin{bmatrix} 1. & 2. & 4. \\ 2. & 3. & 5. \\ 4. & 5. & 6. \end{bmatrix} \begin{bmatrix} -2 & 3 & -1 \end{bmatrix} \xrightarrow{\text{YREVAD}} \begin{bmatrix} 6. & 4. & -5. \\ 4. & 1. & -2. \\ -5. & -2. & 3. \end{bmatrix}.$$

The above **YREVAD** procedure may be compared with the more conventional triple matrix product procedure below.

The equation for the coordinate transformation would be

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} & -1. & \\ & & 1. \\ -1. & & \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}. \qquad (1$$

The strain energy expression with the stiffness matrix is given by $U = \frac{1}{2} \{q\}^T [k] \{q\}$ or for coordinate system 1,

$$U = \frac{1}{2} [x_1 \; y_1 \; z_1] \begin{bmatrix} 1. & 2. & 4. \\ 2. & 3. & 5. \\ 4. & 5. & 6. \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}. \qquad (2)$$

Substituting Equation (1) into (2) gives

$$U = \frac{1}{2} [x_2 \; y_2 \; z_2] \begin{bmatrix} & & -1. \\ -1. & & \\ & 1. & \end{bmatrix} \begin{bmatrix} 1. & 2. & 4. \\ 2. & 3. & 5. \\ 4. & 5. & 6. \end{bmatrix} \begin{bmatrix} & -1. & \\ & & 1. \\ -1. & & \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix},$$

thus the inner triple matrix product gives the stiffness matrix in coordinate system 2, i.e.,

$$\begin{bmatrix} 6. & 4. & -5. \\ 4. & 1. & -2. \\ -5. & -2. & 3. \end{bmatrix}.$$

which is the same result as that obtained from the **YREVAD** procedure.

The advantages of using the **YREVAD** procedure over the triple matrix product procedure are:

1) Less computer time;
2) Less computer core is used;
3) Usually easier to code the $\{IVEC\}$ (thus $\{JVEC\}$) than to code the transformation matrix.

DESCRIPTION OF TECHNIQUE

Matrix [A] is read from NUTA and the non-zero element location numbers are replaced according to the contents of {IVEC} and {JVEC}. The sign of the non-zero elements is changed accordingly and the revised $[\bar{A}]$ is stored on NUT1. Matrix [Z in] is read from NUTZ and stored on NUT2. Subroutine YAABB is then called to sum $[\bar{A}]$ and [Z in]. The resulting matrix [Z out] is stored on NUTZ.

Subroutine YRTAPE reads partitions of a selected FORMA sparse notation matrix from a FORMA reserve tape (disc) into the computer core and stores them sequentially on a utility tape to be operated on by other FORMA subroutines. The matrix to be selected is identified by the desired run number and matrix name. This procedure is accomplished by searching the matrix headings (See Subroutine YWTAPE explanation.) until a match with the desired run number and matrix name is obtained, and then reading the matrix elements and element locations into core and storing them on a utility tape. This procedure is repeated for each matrix partition. The search starts from the current position (does not rewind) of the tape (disk) and proceeds to the EOT (end of tape defined in subroutine YWTAPE explanation). If the desired matrix was not found upon reaching the EOT, a rewind is performed and one more search to the EOT is made. If the desired matrix is again not found, (1) an error message is printed, (2) a listing of the matrix headings is printed (See Subroutine LTAPE writeup), and (3) transfer is made to Subroutine ZZBOMB where the program is terminated.

Subroutine YRVAD1 performs a special case of the function performed by FORMA Subroutines REVADD and YREVAD (see subroutine explanations for these subroutines). Subroutine YRVAD1 rearranges (revises) the rows and columns of a matrix [A] , multiplies [A] by a scalar alpha, and adds the result to a previously defined matrix [Z in] . [A] is a matrix in FORMA dense storage. [Z in] and [Z out] are stored on NUTZ in FORMA sparse notation. YRVAD1 may be used only when [A] , [Z in] , and [Z out] are all symmetric. Because of this symmetry, only one vector {IJVEC} is required for the revision of matrix [A] .

DESCRIPTION OF TECHNIQUE

Any non-zero elements in [A] are stored in workspace V. Element location numbers are formed for the corresponding elements in workspace LV. These elements of [A] are then stored on NUT1 in FORMA sparse notation. Matrix [Z in] is read from NUTZ and stored on NUT2. Subroutine YAABB is called to add sparse matrix [A] to matrix [Z in] and the resulting matrix [Z out] is stored on NUTZ.

Limitation:

The dimension size of V and LV must exceed or be equal to the number of non-zero elements in matrix [A] .

Subroutine YRVAD2 reads small dense matrices and integer vectors from a utility tape (disk) and uses the vectors to revise the matrix locations to form large sparse matrices. These sparse matrices are then summed to form one large matrix in FORMA sparse notation [Z]. Because the small matrices are assumed to be symmetric, [Z] is also symmetric and only the lower half is formed. Only matrix elements having an absolute value greater than 10^{-25} are summed in [Z]. Subroutine YRVAD2 reads matrices from the utility tape until a blank record is found.

Subroutine YRVIS1 rearranges (revises) the columns of a matrix [A] in FORMA dense storage to form matrix [Z] in FORMA sparse notation. The rows of [A] are not rearranged, thus only one integer vector {JVEC} need be input to this subroutine.

DESCRIPTION OF TECHNIQUE

The non-zero elements of [A] are stored in workspace V. Element location numbers are formed in workspace LV corresponding to the elements in V. [Z] , composed of these elements and locations, is stored on NUTZ.

Limitation:

The dimension size of V and LV must exceed or be equal to the number of non-zero elements in [A] .

Subroutine YSRED2 operates on the stiffness matrix [A] to form a reduced stiffness matrix [R] and/or the reducing transformation [T]. The relation between the stiffness matrix [A], displacements {X}, and applied forces {B} may be expressed in matrix form as

$$[A]\ \{X\} = \{B\} \qquad [1]$$

The reduction method assumes Eq [1] to be partitioned as

$$\begin{bmatrix} [A_{11}] & [A_{12}] \\ [A_{21}] & [A_{22}] \end{bmatrix} \begin{Bmatrix} \{X_1\} \\ \{X_2\} \end{Bmatrix} = \begin{Bmatrix} \{B_1\} \\ \{B_2\} \end{Bmatrix} \qquad [2]$$

where $\{X_1\}$ are the displacements to be reduced out and $\{X_2\}$ are the displacements to be retained. The applied forces acting on the coordinates to be reduced are assumed to be zero, such that

$$\{B_1\} = \{0\} \qquad [3]$$

Substituting Eq [3] into Eq [2] and expanding the upper partition, will yield the reduced displacements in terms of the retained displacements as

$$\{X_1\} = -\ [A_{11}]^{-1}\ [A_{12}]\ \{X_2\} \qquad [4]$$

Expanding the lower partitions of Eq [2] and substituting Eq [4] will yield the reduced stiffness matrix as

$$[R]\ \{X_2\} = \{B_2\} \qquad [5]$$

where [R] is the reduced stiffness matrix and is expressed as

$$[R] = [A_{22}] - [A_{21}]\ [A_{11}]^{-1}\ [A_{12}] \qquad [6]$$

The reducing transformation [T] may be expressed using Eq [4] as

$$\begin{Bmatrix} \{X_1\} \\ \{X_2\} \end{Bmatrix} = [T]\ \{X_2\} \qquad [7]$$

where

$$[T] = \begin{bmatrix} -[A_{11}]^{-1} & [A_{12}] \\ [I] & \end{bmatrix} \qquad [8]$$

Also

$$[R] = [T]^T [A] [T] \qquad [9]$$

DESCRIPTION OF TECHNIQUE

This subroutine uses Gauss reduction partially completed to form matrix [R] and [T] from stiffness matrix [A]. As an example of the method, consider three simultaneous equations of the following form

$$\begin{aligned} a_{11} x_1 + a_{12} x_2 + a_{13} x_3 &= b_1 \\ a_{21} x_1 + a_{22} x_2 + a_{23} x_3 &= b_2 \\ a_{31} x_1 + a_{32} x_2 + a_{33} x_3 &= b_3 \end{aligned} \qquad [10]$$

These equations may be written in matrix form as

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \end{Bmatrix} \qquad [11]$$

Solve the first Equation for x_1 as

$$x_1 = - \frac{a_{12}}{a_{11}} x_2 - \frac{a_{13}}{a_{11}} x_3 + \frac{b_1}{a_{11}} \qquad [12]$$

Substituting Eq [12] into the second and third equations in Eq [11] and divide the first by a_{11} results in

$$\begin{bmatrix} 1 & a^*_{12} & a^*_{13} \\ 0 & a^*_{22} & a^*_{23} \\ 0 & a^*_{32} & a^*_{33} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} b^*_1 \\ b^*_2 \\ b^*_3 \end{Bmatrix} \qquad [13]$$

where

$$a_{12}^{*} = \frac{a_{12}}{a_{11}} \qquad [14]$$

$$a_{13}^{*} = \frac{a_{13}}{a_{11}} \qquad [15]$$

$$a_{22}^{*} = a_{22} - \frac{a_{12}\, a_{21}}{a_{11}} \qquad [16]$$

$$a_{23}^{*} = a_{23} - \frac{a_{13}\, a_{21}}{a_{11}} \qquad [17]$$

$$a_{32}^{*} = a_{32} - \frac{a_{12}\, a_{31}}{a_{11}} \qquad [18]$$

$$a_{33}^{*} = a_{33} - \frac{a_{13}\, a_{31}}{a_{11}} \qquad [19]$$

$$b_{1}^{*} = \frac{b_{1}}{a_{11}} \qquad [20]$$

$$b_{2}^{*} = b_{2} - \frac{b_{1}\, a_{21}}{a_{11}} \qquad [21]$$

$$b_{3}^{*} = b_{3} - \frac{b_{1}\, a_{31}}{a_{11}} \qquad [22]$$

Solve the second equation for x_2 which will yield

$$x_{2} = -\frac{a_{23}^{*}}{a_{22}^{*}}\, x_{3} + \frac{b_{2}^{*}}{a_{22}^{*}} \qquad [23]$$

Substitute Eq [23] into the third equation in Eq [13] and divide the second equation by a_{22}^{*}. This results in

$$\begin{bmatrix} 1 & a_{12}^{*} & a_{13}^{*} \\ 0 & 1 & a_{23}^{**} \\ 0 & 0 & a_{33}^{**} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} b_1^{*} \\ b_2^{**} \\ b_3^{**} \end{Bmatrix} \qquad [24]$$

where

$$a_{23}^{**} = \frac{a_{23}^{*}}{a_{22}^{*}} \quad [25]$$

$$a_{33}^{**} = a_{33}^{*} - \frac{a_{23}^{*} a_{32}^{*}}{a_{22}^{*}} \quad [26]$$

$$b_{2}^{**} = \frac{b_{2}^{*}}{a_{22}^{*}} \quad [27]$$

$$b_{3}^{**} = b_{3}^{*} - \frac{b_{2}^{*} a_{32}^{*}}{a_{22}^{*}} \quad [28]$$

The reduced stiffness matrix has been formed and is contained as the a_{33}^{**} element. This can be shown if in Eq [2] we let

$$\{X_1\} = \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} \quad [29]$$

$$\{X_2\} = \{x_3\} \quad [30]$$

$$[A_{11}] = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad [31]$$

$$[A_{12}] = \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix} \quad [32]$$

$$[A_{21}] = [a_{31} \; a_{32}] \quad [33]$$

$$[A_{22}] = a_{33} \quad [34]$$

$$\{B_1\} = \begin{Bmatrix} b_1 \\ b_2 \end{Bmatrix} \quad [35]$$

$$\{B_2\} = b_3 \quad [36]$$

Substituting Eq [31] through [34] into Eq [6] results in a reduced stiffness matrix of the form

$$[R] = a_{33} + \frac{a_{21}\, a_{23}\, a_{31} + a_{12}\, a_{13}\, a_{32} - a_{13}\, a_{22}\, a_{31} - a_{11}\, a_{23}\, a_{32}}{a_{11}\, a_{22} - a_{12}\, a_{21}} \qquad [37]$$

Equation [37] is identical to the result obtained by expanding Eq [26]. Thus, Gauss reduction partially completed yields the reduced stiffness matrix.

The reducing transformation may also be obtained using Gauss reduction if additional operations are performed. From Eq [24], solve the second equation for x_2

$$x_2 = - a_{23}^{**}\ x_3 + b_2^{**} \qquad [38]$$

Substitute Eq [38] into the first equation in Eq [24], which will yield

$$\begin{bmatrix} 1 & 0 & a_{13}^{***} \\ 0 & 1 & a_{23}^{**} \\ 0 & 0 & a_{33}^{**} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} b_1^{***} \\ b_2^{**} \\ b_3^{**} \end{Bmatrix} \qquad [39]$$

where

$$a_{13}^{***} = a_{13}^{*} - a_{12}^{*}\, a_{23}^{**} \qquad [40]$$

$$b_1^{***} = b_1^{*} - a_{12}^{*}\, b_2^{**} \qquad [41]$$

Inspection of Eq [39] shows that we have formed a unity matrix partition where the row-columns were reduced. The a_{13}^{***} and the a_{23}^{**} elements contain the necessary information to form the reducing transformation. To show this, substitute Eq [31] and [32] into Eq [8] to yield

$$[T] = \begin{bmatrix} \dfrac{a_{21}\,a_{23} - a_{13}\,a_{22}}{a_{11}\,a_{22} - a_{21}\,a_{12}} \\ \\ \dfrac{a_{12}\,a_{13} - a_{11}\,a_{23}}{a_{11}\,a_{22} - a_{21}\,a_{12}} \\ \\ 1 \end{bmatrix} \qquad [42]$$

The second and third rows of Eq [42] are equal to the negative of elements a_{13}^{***} and a_{23}^{**} in Eq [39]. Thus, Gauss reduction also yields the reducing transformation.

DESCRIPTION OF TECHNIQUE

The stiffness matrix [A] is read from NUTA in FORMA sparse notation, converted to banded notation, and stored on NUT1. This banded notation stores groups of columns of the upper half of symmetric matrix [A] . The reducing procedure is similar to the decomposition procedure of subroutines YDCOM3 and YDCM3A. The reduced stiffness matrix [R] is stored on NUTR in FORMA sparse notation. If input argument "IFT" is equal to one, the reducing transformation is formed and stored on NUTT in FORMA sparse notation.

Subroutine YDTOS reads a FORMA sparse matrix from the peripheral device on which the matrix is written and stores it in a dimensional work space in core. The work space is zeroed before the nonzero terms of the sparse matrix are inserted into the work space.

DESCRIPTION OF TECHNIQUE

Dense matrix [Z] is formed from the nonzero terms of FORMA sparse matrix [A]. After all of [Z] is zeroed, each partition of [A] is sequentially read into work vectors {V} and {LV}. For each nonzero element location, LV(k), i and j are found:

i = LV(k)/100000

j = LV(k)-100000 * i

then

z_{ij} = V(k) for all elements of [A].

Subroutine YSYMUH symmetrizes a square FORMA sparse matrix [A]. Existing elements above the diagonal are set to zero and new elements are added above the diagonal to reflect elements below the diagonal. The matrix is then searched and any zero elements are removed. Finally, the elements and their locations are ordered. This result is [Z] where:

$$z_{ij} = a_{ij} \qquad i \geq j$$

and

$$z_{ij} = a_{ji} \qquad i < j \ .$$

Subroutine YTRANS calculates the transpose (interchange of rows and columns) of a matrix in FORMA sparse notation. If $[A]_{NRA \times NCA}$ is the matrix to be transposed, then the result is

$$[Z]_{NCA \times NRA} = [A]^T$$

where

$$z_{ji} = a_{ij} \qquad \begin{pmatrix} i = 1, NRA \\ j = 1, NCA \end{pmatrix}$$

NRA is the number of rows of [A], and NCA is the number of columns of [A].

DESCRIPTION OF TECHNIQUE

The location of each a_{ij} (100000 * i + j) is transposed to 100000 * j + i and the matrix is reordered.

Subroutine YUNITY generates a square matrix [Z] with diagonal elements equal to one and all off-diagonal elements equal to zero. That is,

$$z_{ij} = 1. \qquad (i = j)$$

$$z_{ij} = 0. \qquad (i \neq j)$$

In matrix notation,

$$[Z]_{NxN} = \begin{bmatrix} 1. & & 0. \\ & \ddots & \\ 0. & & 1. \end{bmatrix}$$

where N is the size of [Z] (square). A synonym for the unity matrix is the identity matrix, thus the usual designation as [I] .

A matrix is unaltered when multiplied by the unity matrix and the process is commutative. In matrix notation,

$$[Z]\,[I] = [I]\,[Z] = [Z] .$$

DESCRIPTION OF TECHNIQUE

N ones are stored in workspace V and N diagonal element location numbers are stored in workspace LV. [Z] , composed of these ones and location numbers is stored on NUTA.

Subroutine YWRITE writes a FORMA sparse notation matrix of real numbers (a FORTRAN term for numbers with a decimal point) on paper. A group of up to ten consecutive elements from a row of the matrix is printed on each line. If all of the elements of a group are zero, printing of this line is suppressed.

Each matrix printed begins on a new page. On each page of printout is the page heading given by Subroutine PAGEHD, the name of the matrix, the row size and column size of the matrix, the number on nonzero elements in the matrix, and the number of partitions in the matrix.

This is followed by the matrix data. On any line of matrix data the first integer number is the row number of the matrix elements on that line. The second integer number is the column number of the matrix element in the first data field. The next group of real numbers (up to ten) are the values of the matrix elements. This group of matrix elements is given in consecutive column order.

Subroutine YWTAPE writes FORMA sparse matrix data at the end of existing written matrix data on a FORMA tape (disk is preferred, see below). Each set of matrix data consists of two logical records. The first record contains the matrix heading (tape identification, location number, run number of rows of matrix, number of columns of matrix, date, the acronym "spart," the number of nonzeroes in the partition, the number of the partition, and the number of partitions in the matrix. The second record consists of the matrix elements of the partition and the element locations. Subroutine YWTAPE is compatible with FORMA subroutine WTAPE and dense and sparse matrices may be stored on the same FORMA reserve tape.

A schematic representation of the tape (disk) is given by the following sketch.

Beginning
of
tape (disk)

where

H_i = Matrix heading of the i^{th} written matrix partition,

E_i = Matrix elements of the i^{th} written matrix partition,

EOT = End of Tape. Data written by Subroutine WTAPE, YWTAPE, or INTAPE that all FORMA tape subroutines recognize as being the end of written data.

Each vertical line is an end of logical record put on by computer system's routines. The tape is written in binary form as opposed to binary coded decimal (BCD) form.

To find the end of written matrix data, a search is made of the matrix headings until the EOT is found. For this reason, a "new" tape (disk) must be initialized with Subroutine INTAPE so that the tape (disk) contains an EOT. A "new" tape (disk) is defined to be a tape (disk) for which it is desired to start writing matrix data at the front of the tape (disk). Thus, a "new" tape (disk) could be one with obsolete FORMA matrix data on it as well as one that has never been written on by the FORMA system. When the EOT is found, a backspace operation is done over the EOT, and then the current matrix heading, current matrix elements, and a new EOT is written.

A disk is preferred to a tape because the physical separation of the read and write heads on most tape drives may cause tape tolerance problems; thus back-spacing over the EOT is usually not successful. Instead of ending up positioned in front of the EOT, the write head is often positioned in front of the previous matrix elements (E_n is the above sketch). The current matrix heading will be written over previous matrix elements. This causes problems later when trying to read the records written on the tape. To alleviate this problem, it is strongly recommended that all FORMA tape subroutines use an intermediate device such as a disk. At the start of a computer run, the existing tape should be copied on the disk by using computer control cards. Likewise, at the end of the run, the disk should be copied back on tape by using computer control cards.

Subroutine YZERLH eliminates the non-zero elements below the diagonal of a matrix [A] stored on NUTA in FORMA sparse notation. That is,

$$a_{ij} = 0. \qquad (i > j)$$

EXAMPLE

If [A] is input to Subroutine YZERLH as

$$[A]_{3x3} = \begin{bmatrix} 1. & 2. & 3. \\ 4. & 5. & 6. \\ 7. & 8. & 9. \end{bmatrix}$$

the matrix output from this subroutine will be

$$[A]_{3x3} = \begin{bmatrix} 1. & 2. & 3. \\ 0. & 5. & 6. \\ 0. & 0. & 9. \end{bmatrix}$$

Matrix [A] is read into workspaces V and LV. The non-zero element locations are searched for element locations below the diagonal. These locations and the corresponding elements are eliminated.

Subroutine YZERUH eliminates the non-zero elements above the diagonal of a matrix [A] stored on NUTA in Forma sparse notation. That is,

$$a_{ij} = 0. \qquad (i < j)$$

EXAMPLE

If [A] is input to Subroutine YZERUH as

$$[A]_{3x3} = \begin{bmatrix} 1. & 2. & 3. \\ 4. & 5. & 6. \\ 7. & 8. & 9. \end{bmatrix}$$

the matrix output from this subroutine will be

$$[A]_{3x3} = \begin{bmatrix} 1. & 0. & 0. \\ 4. & 5. & 0. \\ 7. & 8. & 9. \end{bmatrix}$$

Matrix [A] is read into workspaces V and LV. The non-zero element locations are searched for element locations above the diagonal. These locations and the corresponding elements are eliminated.

Subroutine **YZERO** generates a matrix with each element equal to zero. That is,

$$z_{ij} = 0. \qquad \begin{pmatrix} i = 1, NR \\ j = 1, NC \end{pmatrix}$$

In matrix notation,

$$[Z]_{NRxNC} = \begin{bmatrix} 0. & 0. & \cdots & 0. \\ 0. & 0. & & \vdots \\ \vdots & & & \vdots \\ 0. & . & . & 0. \end{bmatrix}$$

where NR is the number of rows of [Z], and NC is the number of columns of [Z].

This subroutine is useful in setting a matrix array to zero before performing subsequent operations such as matrix assembly (**ASSEM**) or revision/addition of one matrix into another (**YREVAD**).

Matrix [Z] is stored on NUTA in FORMA sparse notation. Because matrix elements equal to zero are not stored in this notation, only matrix headers are stored on NUTA.