

AN ALGEBRAIC EQUATION SOLUTION PROCESS FORMULATED
IN ANTICIPATION OF BANDED LINEAR EQUATIONS

by

Frank L. Endres
Hudson Matlock

Research Report Number 56-19

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

Research Project 3-5-63-56

conducted for

The Texas Highway Department

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

by the

CENTER FOR HIGHWAY RESEARCH
THE UNIVERSITY OF TEXAS AT AUSTIN

January 1971

The opinions, findings, and conclusions expressed in this publication are those of the authors and not necessarily those of the Federal Highway Administration.

PREFACE

Computer storage and time requirements for structural problems will often determine whether or not a particular program is feasible to use on an extensive basis. For multi-dimensioned problems requiring fine mesh spacing and involving nonlinear or time-dependent behavior, careful attention must be given to the efficiency of the solution process, even with the largest and fastest computers in use today.

This report describes a system of equation solving routines that may be applied to a wide variety of problems by utilizing them within appropriate programs. The routines will not be directly apparent to the structural or pavement design engineer in routine work; instead, it is the one who develops the program or the one who is concerned with fitting a program to run on a particular computer who will be directly involved with the material in this report.

The routines have been incorporated in many of the programs described in other reports of the current project. Because they are potentially very useful in optimizing solution processes in future developments, it was decided to document the routines separately by means of this report.

This page replaces an intentionally blank page in the original.

-- CTR Library Digitization Team

LIST OF REPORTS

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Report No. 56-18, "Discrete-Element Analysis for Anisotropic Skew Plates and Grids" by Mahendrakumar R. Vora and Hudson Matlock, describes a tridirectional model and a computer program for the analysis of anisotropic skew plates or slabs with grid-beams.

Report No. 56-19, "An Algebraic Equation Solution Process Formulated in Anticipation of Banded Linear Equations" by Frank L. Endres and Hudson Matlock, describes a system of equation-solving routines that may be applied to a wide variety of problems by utilizing them within appropriate programs.

Report No. 56-20, "Finite-Element Method of Analysis for Plane Curved Girders" by William P. Dawkins, presents a method of analysis that may be applied to plane-curved highway bridge girders and other structural members composed of straight and curved sections.

Report No. 56-21, "Linearly Elastic Analysis of Plane Frames Subjected to Complex Loading Condition" by Clifford O. Hays and Hudson Matlock, presents a design-oriented computer solution of plane frame structures that has the capability to economically analyze skewed frames and trusses with variable cross-section members randomly loaded and supported for a large number of loading conditions.

ABSTRACT

A general method for the solution of large, sparsely banded, positive-definite, coefficient matrices is presented. The goal in developing the method was to produce an efficient and reliable solution process and to provide the user-programmer with a package which is problem-independent, efficient, and easy to use, so that program development time can be spent in problem analysis rather than on solution technique.

The procedures have been developed specifically to deal with matrices generated by three and five-wide difference operators, whether symmetrical or unsymmetrical.

KEY WORDS: structural analysis, numerical analysis, computers, mathematics, banded equations, finite differences.

This page replaces an intentionally blank page in the original.

-- CTR Library Digitization Team

SUMMARY

We see that for systems having numerous constant vectors or coefficient matrices which are large, sparse, or possess a moderate degree of bandedness direct methods are generally preferable to iterative methods.

Since we have considered both symmetric and nonsymmetric problems as well as three and five-wide difference operators, four separate routines have been programmed. For the nonsymmetric case having a three-wide difference operator TRIP 3 (three-wide recursion inversion procedure) was developed, TRIP 4 is for the symmetric case. For the five-wide nonsymmetric case there is FRIP 3, and for the symmetric case FRIP 4. The above four routines are flow charted and listed in Appendix A.

Each of these routine drives (or calls upon) a group of secondary matrix manipulation routines referred to as the SUMP pack (Submatrix Manipulation Package). The particular group used by these four routines is SUMP 6. For a more complete description of these secondary routines see Appendix B.

The user of the solution procedure must in some way transmit his stiffness matrix to it, and since for the solution procedure only one partitioned level is needed at each recurrence of the algorithm, a shuttle routine which is called at each step must be provided by the user. It is here that he computes or fetches the appropriate level or partition of his stiffness matrix. A complete description of this routine is given in Appendix C while an example of the use of the entire package is presented in Appendix D.

This page replaces an intentionally blank page in the original.

-- CTR Library Digitization Team

IMPLEMENTATION STATEMENT

The routines developed and explained herein are of immediate benefit to the engineer who is developing a structural analysis program. The main body of the report is concerned with the theoretical development of the procedures and the appendices pertain to their implementation. These routines provide the engineer-programmer not only with a ready-to-use, efficient solution package, but with one that requires a minimum of input and reorganization of the natural form of the stiffness matrix.

This page replaces an intentionally blank page in the original.

-- CTR Library Digitization Team

TABLE OF CONTENTS

PREFACE	iii
LIST OF REPORTS	v
ABSTRACT AND KEY WORDS	vii
SUMMARY	ix
IMPLEMENTATION STATEMENT	xi
NOMENCLATURE	xv
 CHAPTER 1. INTRODUCTION	
Recursion Technique	1
Requirements	1
Use	2
 CHAPTER 2. RECURSION INVERSION	
The General Problem	3
Nonsymmetric Case	5
Symmetric Case	10
Multiple Constant Vectors	17
 CHAPTER 3. EFFICIENCY, SPEED, AND SIZE	
Computations	19
Accuracy	20
Storage	20
Timing	23
 REFERENCES	 27
 APPENDICES	
Appendix A. Flow Charts and Listings of Main Routines	31
Appendix B. Flow Charts, Listings, and Comments on Subordinate Subprograms for Matrix Manipulations	55

Appendix C. Use of Subroutines FSUB 3 and FSUB 5	87
Example Using FSUB 32 for TRIP 4 (for problem with symmetric coefficient matrix with a three-wide partitioning)	87
Example Using FSUB 52 for FRIP 4	91
Appendix D. Use of Overall Package	95
THE AUTHORS	103

NOMENCLATURE

<u>Symbol</u>	<u>Definition</u>
a_i	Submatrix element of coefficient matrix
A	Coefficient matrix
A_i	Preliminary unknown vector
b_i	Submatrix element of coefficient matrix
B_i	Recursion coefficient
c_i	Submatrix element of coefficient matrix
C_i	Recursion coefficient
d_i	Submatrix element of coefficient matrix
D_i	Recursion multiplier
e_i	Submatrix element of coefficient matrix
E_i	Recursion multiplier
f_i	Subvector element of constant vector
F, f_l	Family of constant vectors
K	Order of partitioned submatrix
L	N/K
N	Order of coefficient matrix
N_1, N_2, \dots, N_5	Band widths, respectively, of a_i, b_i, \dots, e_i
w_i	Subvector element of unknown vector

<u>Symbol</u>	<u>Definition</u>
W, w_ℓ	Family of unknown vectors
X	Coordinate in short direction of grid
Y	Coordinate in long direction of grid
Z^{-1}	Inverse of the matrix Z (any matrix)
Z^T	Transpose of the matrix Z (any matrix)
ϕ	A zero vector, matrix, or region

CHAPTER 1. INTRODUCTION

In physical problems, closed-form solutions are not always available; therefore, discrete methods must be employed to approximate the true solution. One can think of representing a continuous problem by a satisfactory discrete-element model and solve directly for the solution of the model and thereby obtain an approximate solution to the real problem.

In either case, the solution technique boils down to the solving of a system of linear algebraic equations that exhibit a high degree of sparseness and whose non-zero elements tend to cluster about the main diagonal of the coefficient matrix. Most physical systems may be ordered so that this banding phenomena will be observed, but, when care is taken, the width of the band may be reduced, thereby compacting the band and allowing for a more efficient solution. Furthermore, if we enforce a preset method of ordering the grid points, then the form of our finite-difference operator will produce banding within the main band.

It is the purpose of this report to present an efficient, general method for the solution of this type of problem.

Recursion Technique

The general problem can be subdivided in several ways. It can be symmetric or non-symmetric, or it can have three subbands or five. These different considerations are taken up in Chapter 2, along with the associated proofs that verify the method.

Requirements

In Chapter 3 the amount of work (computations) along with the amount of storage required is taken into consideration. Graphs showing actual times involved for a wide range of problems are presented.

Use

The total package can be visualized as three main parts. The first part contains the recursion algorithm and is listed and flow charted in Appendix A. The next part drives a secondary group of matrix manipulation routines, more commonly referred to as the SUMP package. These are located and flow charted in Appendix B. The third and most crucial part to the user is the FSUB routine, where the coefficient matrix is defined. A description of what is necessary for this routine is included in Appendix C, and the relationship of the complete group to the user's main program is discussed in Appendix D.

CHAPTER 2. RECURSION INVERSION

The General Problem

While the type of coefficient matrix we will concern ourselves with generally arises in physical problems where a discrete-element model of the real problem is assumed and a finite difference operator is applied, the solution technique is in no way bound to problems of this type. Any coefficient matrix which is positive definite and has the form of Fig 2.1 can be solved using this method. Although this method can be extended, in this discussion we will consider only three and five-wide type problems, respectively resulting from three and five-wide difference operators. We will also consider both the symmetrical and nonsymmetrical cases. These limits were selected since the one-wide case is trivial and the five-wide is extensive enough for most two-dimensional problems.

There is some disagreement as to whether one should consider direct or iterative solution techniques to solve this problem. The main advantages of the iterative processes are that they require less storage and are not as susceptible to round-off error. The main advantages of the direct methods are that for problems that give rise to coefficient matrices with "small" band widths they require less work. Also when it is known that in general the problems have the same coefficient matrix A and differ only in the constant vector f ($AW_{\ell} = f_{\ell}$ or $AW = F$), which is quite often the case, the direct methods are capable of solving the succeeding problems for as little as 5 percent of the effort required to solve the initial problem.

In addition, if the problems are of the type that produces systems of heterogeneous coefficients, there may be complications in the iterative methods, whereas the direct methods are not as sensitive.

The power of this method is its ability to handle efficiently matrices exhibiting second level banding which lends itself to partitioning into submatrices which in turn are also banded.

Therefore, in the methods to be presented, we are assuming the widths of the second level bands (N_1, N_2, \dots) are small relative to K , the order of the submatrices, and the overall band width is small and problems of the type $AW = F$ are expected.

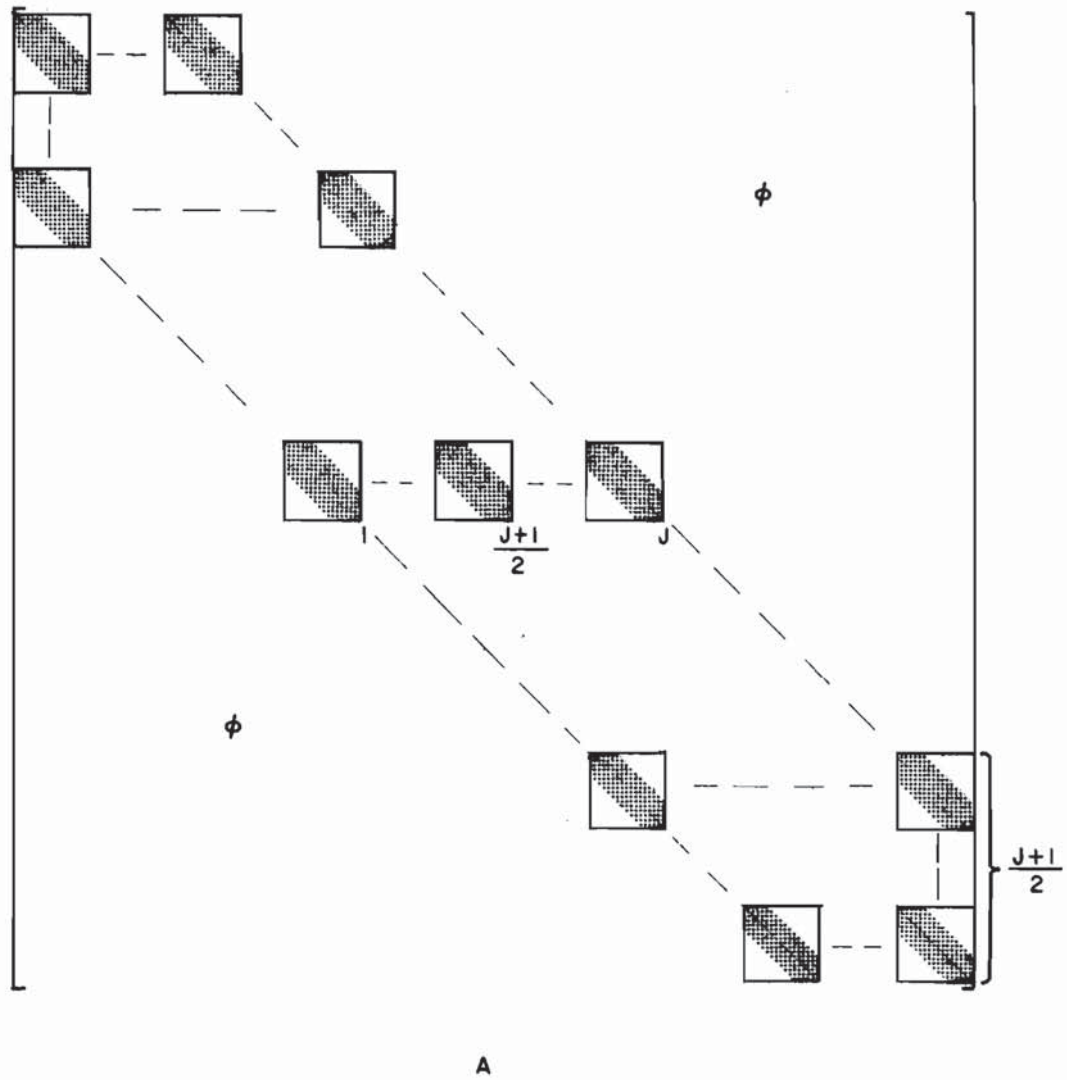


Fig 2.1. Coefficient matrix for a general J-wide operator.

Figure 2.1 shows a general J-wide partitioned matrix. Figures 2.2, 2.3, and 2.4 show a specific instance of a coefficient matrix resulting from the application of a five-wide difference operator to a real grid.

In most instances, we also know that the coefficient matrix is symmetric in addition to being positive definite. Since we can take advantage of this additional knowledge, we will develop two procedures: one for the symmetric case, one for the nonsymmetric case.

We have stated that we will limit ourselves to coefficient matrices which can be partitioned into three and five-wide bands. In terms of our difference operator, this restricts its vertical width but leaves the horizontal dimensions completely general. If we think of our coefficient matrix without reference to a difference operator, this merely means the respective widths of the bands are arbitrary up to the width of the submatrix itself. Of course, for the symmetric case, the bands must be symmetric about the main diagonal, e.g., $N_1 = N_5$ and $N_2 = N_4$ (refer to Figs 2.5 through 2.10).

Nonsymmetric Case

In the case where the operator is applied once and only once at each grid point, the respective widths of the subbands will be N_1 , N_2 , and N_3 for the three-wide case and N_1 , N_2 , N_3 , N_4 , and N_5 for the five-wide.

To insure this form and the narrowest band width, we must enforce the ordering of the mesh points as shown in Fig 2.11.

If we think of applying our operator, this tells us we apply it first to the bottom row of points from left to right. Then we go to the second row and so on, moving upward. If we now look at the partitioned matrix of Fig 2.4, we see that each submatrix has order K ($K \times K$ elements) and the stiffness matrix is composed of $L \times L$ submatrices. If we look at that portion of the coefficient matrix formed by applying the operator to the i^{th} row and for convenience assume we have a five-wide operator, we have the nonsymmetric case shown in Fig 2.12. Referring to the figure, we have

$$a_i w_{i-2} + b_i w_{i-1} + c_i w_i + d_i w_{i+1} + e_i w_{i+2} = f_i \quad (1)$$

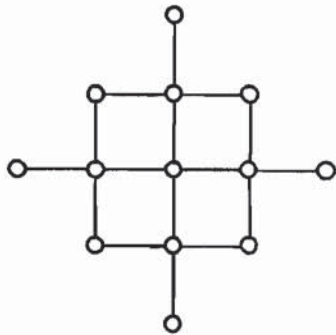


Fig 2.2. Typical five-wide difference operator.

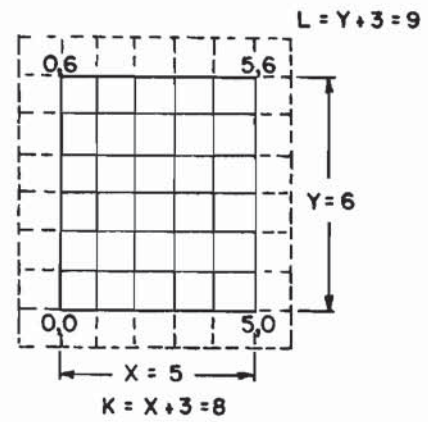


Fig 2.3. A specific discrete model.

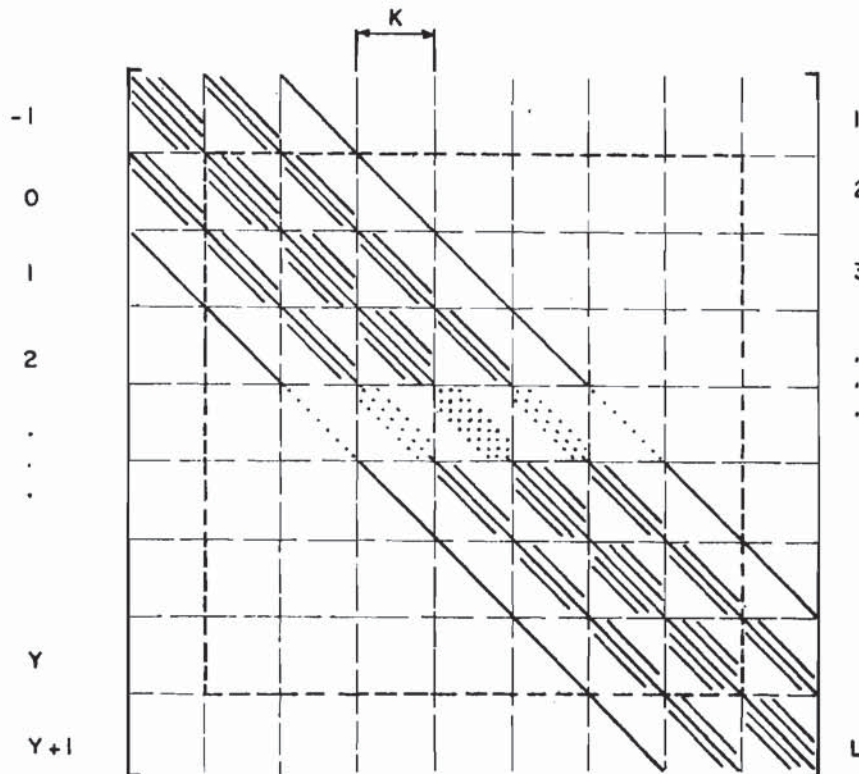


Fig 2.4. Coefficient matrix resulting from the application of the difference operator of Fig 2.2 to the discrete model of Fig 2.3.



Fig 2.5. A typical three-wide difference operator.



Fig 2.6. A typical five-wide difference operator.

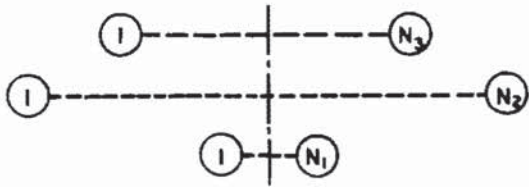


Fig 2.7. General three-wide operator.

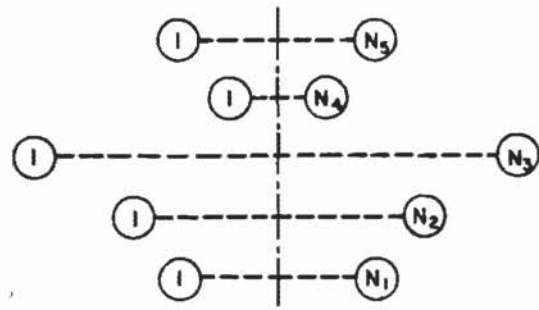


Fig 2.8. General five-wide operator.

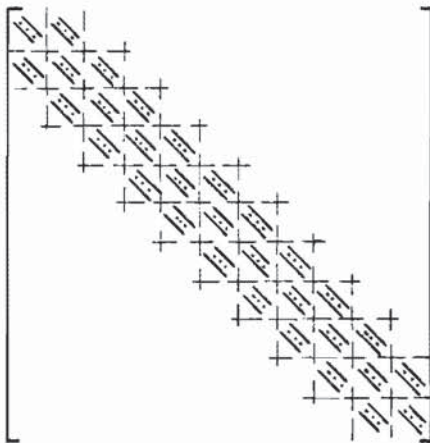


Fig 2.9. Coefficient matrix resulting from a general three-wide difference operator.

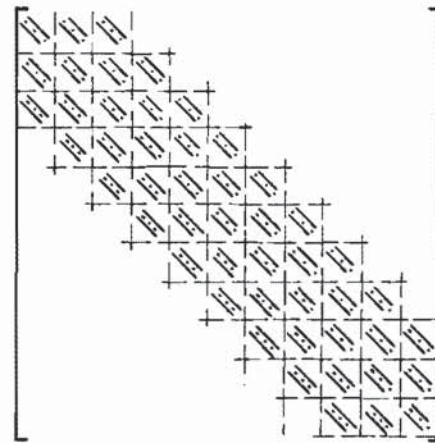


Fig 2.10. Coefficient matrix resulting from a general five-wide difference operator.

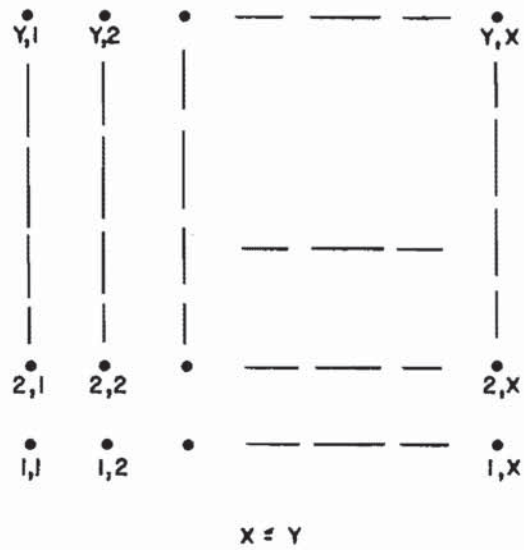


Fig 2.11. Imposed ordering of grid points.

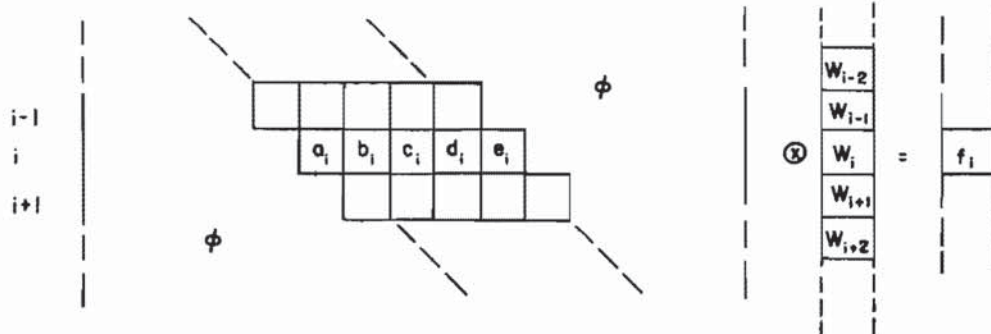


Fig 2.12. i^{th} partition of five-wide stiffness matrix.

Assume

$$w_i = A_i + B_i w_{i+1} + C_i w_{i+2}$$

Then

$$w_{i-1} = A_{i-1} + B_{i-1} w_i + C_{i-1} w_{i+1}$$

and

$$w_{i-2} = A_{i-2} + B_{i-2} w_{i-1} + C_{i-2} w_i \quad (2)$$

Substitution into Eq 1 to eliminate the variables w_{i-2} and w_{i-1} gives rise to an equation with the form of Eq 2 where

$$A_i = D_i (E_i A_{i-1} + a_i A_{i-2} - f_i)$$

$$B_i = D_i (E_i C_{i-1} + d_i)$$

$$C_i = D_i e_i$$

and where

$$D_i = -(a_i C_{i-2} + E_i B_{i-1} + c_i)^{-1}$$

$$E_i = a_i B_{i-2} + b_i$$

A , B , and C are referred to as recursion coefficients and D and E as recursion multipliers for the five-wide case.

Now with the appropriate starting values, we have an efficient two-pass, matrix elimination procedure.

For the three-wide band, we have a similar situation where by referring to Fig 2.13 we see that

$$b_i w_{i-1} + c_i w_i + d_i w_{i+1} = f_i \quad (3)$$

Assume

$$w_i = A_i + C_i w_{i+1}$$

Then

$$w_{i-1} = A_{i-1} + C_{i-1} w_i \quad (4)$$

Substitution into Eq 3 to eliminate w_{i-1} gives rise to an equation with the form of Eq 4 where

$$A_i = D_i (b_i A_{i-1} - f_i)$$

$$C_i = D_i d_i$$

and

$$D_i = (b_i C_{i-1} + c_i)^{-1}$$

In the preceding derivations, the sign (-) represents matrix negation, the sign (+) represents matrix addition, $(^{-1})$ represents matrix inversion, and concatenation indicates matrix multiplication.

Symmetric Case

For the symmetric five-wide case, if we look at the i^{th} partition we have what is shown in Fig 2.14. Applying the same type of analysis as done for the nonsymmetric case gives rise to the identical recursion equations with a_i

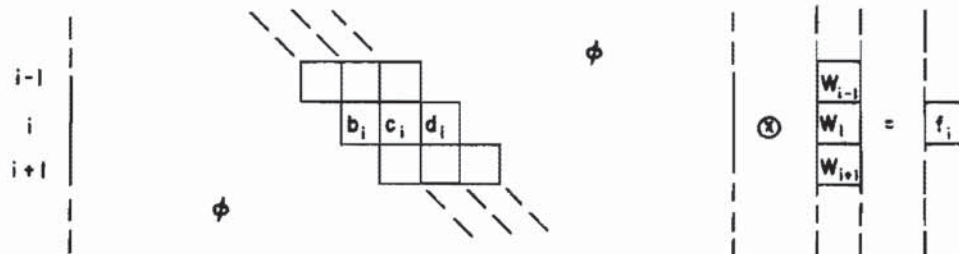


Fig 2.13. i^{th} partition of three-wide stiffness matrix.

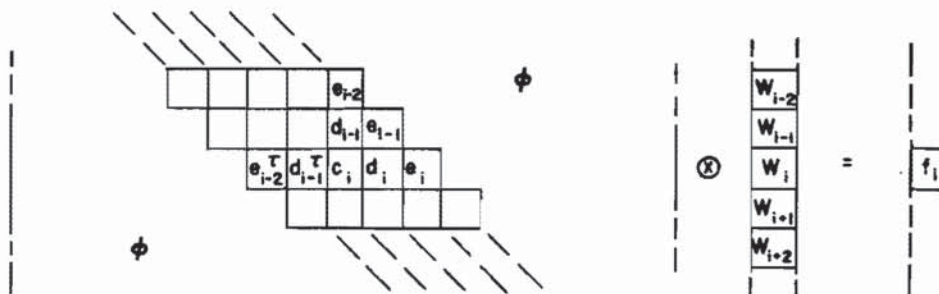


Fig 2.14. i^{th} partition of five-wide symmetric stiffness matrix.

replaced by e_{i-2}^\top and b_i by d_{i-1}^\top . This yields several important results for the five-wide case, namely, that $E_i B_{i-1}$, and therefore D_i , are symmetric and $B_i = D_i E_{i+1}^\top$. We shall later see that this leads to approximately a 50 percent reduction in time for both the three and five-wide cases.

Making the substitutions in Eq 3 gives us

$$A_i = D_i (E_i A_{i-1} + e_{i-2}^\top A_{i-2} - f_i)$$

$$B_i = D_i (E_i C_{i-1} + d_i)$$

$$C_i = D_i e_i$$

and

$$D_i = -(e_{i-2}^\top C_{i-2} + E_i B_{i-1} + c_i)^{-1}$$

$$E_i = e_{i-2}^\top B_{i-2} + d_{i-1}^\top \tag{5}$$

In the following definitions and theorems A and B are square matrices and x is a vector.

Definitions

- I. The statement that A is symmetric means that for all a , $a_{ij} = a_{ji}$.
- II. The statement that A is positive definite means that for any non-zero vector x , $x^\top A x > 0$.

Theorems

- I. If A and B are symmetric, then $A + B$ is symmetric.
- II. If A is symmetric, then for all scalars c , cA is symmetric.
- III. $A + A^\top$ is symmetric.
- IV. If A is symmetric, then BAB^\top is symmetric.

$$V. (A_1 A_2 \dots A_N)^\top = A_N^\top \dots A_2^\top A_1^\top .$$

VI. If A is symmetric, then A^{-1} is symmetric.

$$VII. (A_1 + A_2 + \dots + A_N)^\top = A_1^\top + A_2^\top + \dots + A_N^\top .$$

Proof of the symmetry of D for the five-wide case:

Given Eq 5 and the boundary conditions

$$e_{-1}^\top = d_0^\top = e_0^\top = B_{-1} = C_{-1} = B_0 = C_0 = E_1 = \phi$$

and that the coefficient matrix S and its diagonal partition c_i are symmetric and positive definite.

$$\text{Let } \bar{D}_i = -D_i^{-1}$$

For $i = 1$ and $i = 2$

$$\begin{aligned} \bar{D}_1 &= E_1 B_0 + e_{-1}^\top C_{-1} + c_1 \\ &= c_1 \text{ which is symmetric} \end{aligned}$$

Therefore D_1 is symmetric from VI and II

$$\begin{aligned} \bar{D}_2 &= E_2 B_1 + e_0^\top C_0 + c_2 \\ &= d_1^\top D_1 d_1 + c_2 \end{aligned}$$

$d_1^\top D_1 d_1$ is symmetric because of IV and finally \bar{D}_2 is symmetric because of I. Therefore D_2 is symmetric from VI and II.

Assume D_i is symmetric for $i = k-2$ and $i = k-1$

We now need to show that $B_k = D_k E_{k+1}^\top$.

To do this we will use an inductive proof within our main induction.

For $j = 1$

$$B_1 = D_1 (E_1 C_0 + d_1) = D_1 d_1$$

$$E_2 = e_0^\top B_0 + d_1^\top = d_1^\top$$

$$E_2^\top = d_1$$

Therefore

$$B_1 = D_1 E_2^\top$$

Assume for $j = k-1$

$$B_{k-1} = D_{k-1} E_k^\top$$

$$B_{k-1}^\top = E_k D_{k-1}^\top$$

Therefore

$$D_k (B_{k-1}^\top e_{k-1} + d_k) = D_k (E_k D_{k-1}^\top e_{k-1} + d_k) \quad (6)$$

From Eq 5

$$E_{k+1} = e_{k-1}^\top B_{k-1} + d_k^\top$$

Therefore

$$E_{k+1}^\top = B_{k-1}^\top e_{k-1} + d_k \quad (7)$$

From Eq 5

$$B_k = D_k (E_k C_{k-1} + d_k) \quad (8)$$

and

$$C_{k-1} = D_{k-1} e_{k-1} = D_{k-1}^\top e_{k-1} \quad (9)$$

Substituting Eq 9 into Eq 8 gives

$$B_k = D_k (E_k D_{k-1}^\top e_{k-1} + d_k) \quad (10)$$

and substituting Eq 7 and Eq 10 into Eq 6 gives

$$B_k = D_k E_{k+1}^\top$$

Therefore

$$B_j = D_j E_{j+1}^\top$$

$$j = 1, 2, \dots, k$$

and

$$\begin{aligned} \bar{D}_k &= E_k B_{k-1} + e_{k-2}^\top C_{k-2} + c_k \\ &= E_k D_{k-1} E_k^\top + e_{k-2}^\top D_{k-2} e_{k-2} + c_k \end{aligned} \quad (11)$$

Since D_{k-1} and D_{k-2} have been assumed to be symmetric we have that $E_k D_{k-1} E_k^\top$ and $e_{k-2}^\top D_{k-2}^\top e_{k-2}$ are both symmetric because of IV and finally \bar{D}_k is symmetric from the successive application of I. Therefore D_k is symmetric from VI and II and by induction we conclude that D_i is symmetric for all $i = 1, 2, \dots$. Given this we can now conclude that

$$B_i = D_i E_{i+1}^\top \quad \text{for all } i = 1, 2, \dots \quad (12)$$

and therefore $E_i B_{i-1}$ is symmetric for all $i = 1, 2, \dots$.

The proof is similar but less involved for the three-wide case.

Making these substitutions into Eq 5 gives us our final set of equations for the symmetric case.

Five-wide case

$$A_i = D_i (E_i A_{i-1} + e_{i-2}^\top A_{i-2} - f_i)$$

$$B_i = D_i E_{i+1}^\top$$

$$C_i = D_i e_i$$

$$D_i = -(e_{i-2}^\top C_{i-2} + E_i B_{i-1} + c_i)^{-1}$$

$$E_{i+1} = e_{i-1}^\top B_{i-1} + d_i^\top \quad (13)$$

Three-wide case

$$A_i = D_i (d_{i-1}^\top A_{i-1} - f_i)$$

$$C_i = D_i d_i$$

$$D_i = -(d_{i-1}^T C_{i-1} + c_i)^{-1} \quad (14)$$

If we compare the recursion inversion algorithm with the elimination (a la Gauss) methods, we see analogous patterns. The first phase, in which the recursion coefficients and multipliers are calculated, is analogous to the triangularization phase of the elimination methods. The second phase, in which the recursion equation is solved for the unknown vectors, is analogous to the back-substitution phase.

In the following section we will see that there is also a similar pattern in the way multiple constant vectors can be handled.

Multiple Constant Vectors

A very important part of any solution scheme is its ability to handle problems where for the same coefficient matrix there are numerous constant vectors. We represent this by $Aw = F$ where $F = (f_1, f_2, \dots, f_\ell)$. This actually represents ℓ individual problems,

$$Aw_1 = f_1, \quad Aw_2 = f_2, \quad \text{and so forth.}$$

One way to handle this problem, which seems appealing at first glance, is the classical solution $w_i = A^{-1}f_i$, $i = 1, 2, \dots, \ell$. This requires N^2 multiplications for each vector, which is approximately the same amount required for back substitution, given an upper triangularized matrix or some analogous factorization. But since it requires approximately N^3 multiplications to compute A^{-1} and only $N^3/3$ for the elimination or decomposition methods, we see that the latter are preferable. The above analysis was done with the assumption that A had no special properties. In the case where A is banded, the comparison between the two alternatives becomes even more striking, because the decomposition methods can readily take advantage of the banding whereas the inverse of a banded matrix is not necessarily banded and therefore in general cannot take advantage of this additional information.

If we now consider the recursion inversion approach, we see that the introduction of multiple f 's requires only that for each vector we need to recalculate recursion coefficient A and circumvent the calculations of B , C , D , and E , all of which is approximately equal to the work required for

banded decomposition. In addition, we see that the work required to calculate A is approximately equal to that needed for the transformation of the constant vector F and banded back substitution.

For the sake of discussion and for descriptive purposes in the flow charts, we categorize problems in the following manner:

$$\text{Given } AW = F, \text{ where } F = f_1$$

We refer to this as a standard problem

$$\text{when } AW = F, \text{ where } F = (f_1, f_2, \dots, f_\ell)$$

f_1 is referred to as the parent problem and $f_i, i = 2, 3, \dots, \ell$ are referred to as offspring problems.

While it is true that f_1 through f_ℓ are all of equal importance from the solution standpoint, the distinction between f_1 and the other f 's was made because in the recursion inversion procedure, the original vector is operated on concurrently with the calculation of the recursion coefficients.

CHAPTER 3. EFFICIENCY, SPEED, AND SIZE

To program the recursion inversion procedure in the most efficient manner, it is necessary that the implied matrix operations not be done explicitly, but by banded matrix operation routines which do only the necessary operations. Also, to save storage as well as time, we pack these submatrices, thereby eliminating the zero elements which lie between the bands. Also, to minimize internal core memory requirements, only one level (horizontal partition) will actually be generated at any given time. The packing procedures necessary are explained fully in Appendix C.

Computations

To compute the amount of work necessary, let us first consider in particular the five-wide nonsymmetric case. Let us also assume N_1 and N_5 are small relative to K (the submatrix order), where N_1 and N_5 are the respective band widths of the outside bands. Therefore, the overall band width is $\geq 4K + 1$, and for simplicity and because of the above assumption, let us use $4K$.

If we look back at the recursion coefficients, we see that the dominant operations are the two full matrix multiplications in B and the one multiplication and inversion to get D . Referring to Fig 2.4 for the definition of K and L , we define N as K times L . Each of these requires K^3 multiplications. These must be repeated L times giving us an approximate estimate of

$$\begin{aligned} & 4K^3L \\ = & 4K^2N \\ = & (2K)^2N \end{aligned}$$

which is the amount needed for a comparable form of banded Gaussian elimination.

If we consider back substitution or the necessary work required for additional constant vectors, we see that this involves only the computation of the recursion coefficient A and then the application of the recursion equation (Eq 2, p 4). The former requires roughly $(2K + 1)N$ multiplications while the latter takes $2KN$ making the total $(4K + 1)N$, which also compares almost identically with banded Gaussian back-substitution.

We have used banded Gaussian elimination as a yardstick because, as was mentioned in Chapter 1, it is much more expedient than taking the actual inverse and in fact it has been proven that no direct method can take less computation than Gaussian elimination for a nonsymmetric matrix.

We find that for the three-wide nonsymmetric case, the recursion inversion method also is approximately equivalent to banded Gaussian elimination from the standpoint of computations involved. As was mentioned earlier, the additional information of symmetry reduces both solutions by approximately 50 percent.

Accuracy

Error analysis indicates that for well-conditioned problems at least four significant digits will remain for up to 100,000 equations, where a 60-bit word length is used. In both the nonsymmetric and symmetric case, the routines are such that one can take advantage of the accurate accumulation of inner-products, which can be very helpful for problems that are ill-conditioned. Greater advantage of this can be taken in the symmetric case because a compacted inversion routine is employed which also takes advantage of the accurate accumulation of inner-products.

Storage

For all practical purposes the storage requirements are consumed by the recursion coefficients (Table 3.1). Although it is not necessary, the entire solution vector has been retained in the core for convenience.

For a single vector problem, the auxiliary storage is reduced to $N(2K + 1)$ as one would expect. For the symmetric case we no longer need B_{i-2} but its space is replaced by E_{i+1} . Also, only C_{i-2} actually occurs in the revised formulas, but C_{i-1} is needed as temporary storage. However, by judicious arrangement, we have been able to use B_{i-1} as a temporary and

TABLE 3.1. STORAGE FOR FIVE-WIDE CASE (NONSYMMETRIC)

	Total Storage	Core	Auxiliary
A_i	$K \cdot L$	K	$K \cdot L$
A_{i-1}	K	K	0
A_{i-2}	K	K	0
B_i	$K^2 L = K \cdot N$	K^2	$K^2 \cdot L$
B_{i-1}	K^2	K^2	0
B_{i-2}	K^2	K^2	0
C_i	$K^2 L = K \cdot N$	K^2	$K^2 \cdot L$
C_{i-1}	K^2	K^2	0
C_{i-2}	K^2	K^2	0
D	$K^2 \cdot L$	K^2	$K^2 \cdot L^*$
E	$K^2 \cdot L$	K^2	$K^2 \cdot L^*$
W_i	$K \cdot L$	$K \cdot L$	0

* (This storage is necessary for multiple vector problems only.)

$$\text{Total core storage} = 8K^2 + 3K + K \cdot L$$

$$\text{Total auxiliary storage} = 4K^2 L + K \cdot L = N(4K + 1)$$

delete the need for a storage slot for C_{i-2} . Therefore, the core storage for the symmetric five-wide case is reduced by K^2 .

Table 3.2 describes an analogous situation.

For a single vector problem, the auxiliary storage is reduced to $N(K + 1)$. It should now be obvious why K is chosen to be the smaller of the pair K, L . Although the order of the stiffness matrix ($N = K \cdot L$) is not changed, the band width is directly proportional to K ; therefore it should be the smaller to reduce both the core storage and computational requirements.

TABLE 3.2. STORAGE FOR THREE-WIDE CASE

	Total Storage	Core	Auxiliary
A_i	$K \cdot L$	K	$K \cdot L$
A_{i-1}	K	K	0
C_i	$K^2 L$	K^2	$K^2 L$
D_i	$K^2 L$	K^2	$K^2 L^*$
W_i	$K \cdot L$	$K \cdot L$	0

* (This storage is necessary only for multiple vector problems.)

$$\text{Total core storage } (T_c) = 2K^2 + 2K + K \cdot L$$

$$\text{Total auxiliary storage} = 2K^2 L + KL = N(2K + 1)$$

Let us assume we had a problem that yielded a five-wide problem in which $K = 10$ and $L = 20$:

$$T_c = 8 \cdot 10^2 + 3 \cdot 10 + 10 \cdot 20 = 1030$$

In these computations we have been neglecting the storage necessary for the coefficient matrix itself, but, since we only need one partition at a time, this storage is negligible. If, in our example, we assume a typical 1-3-5-3-1 operator (refer to Fig 2.2), the additional storage required would be $(1 + 3 + 5 + 3 + 1)10 = 130$, which is not only small compared to 1030 but, as L increases, will not change since it is dependent only on K . Also as K increases, this number increases linearly, whereas our total core storage requirement (T_c) increases quadratically.

Timing

Figures 3.1 and 3.2 show the time in seconds for the solutions to problems having square grids ($L = K$). For rectangular grids ($L > K$) the time increases in direct proportion to L and may be estimated by multiplying the time from Fig 3.1 or Fig 3.2 by $\frac{L}{K}$.*

* All runs were made on a CDC 6600, with the SCOPE 2.0 RUN compiler.

Basic execution time in minor cycles:

division - 29 m.c.

multiplication - 10 m.c.

addition - 4 m.c.

1 m.c. = 250 nanoseconds.

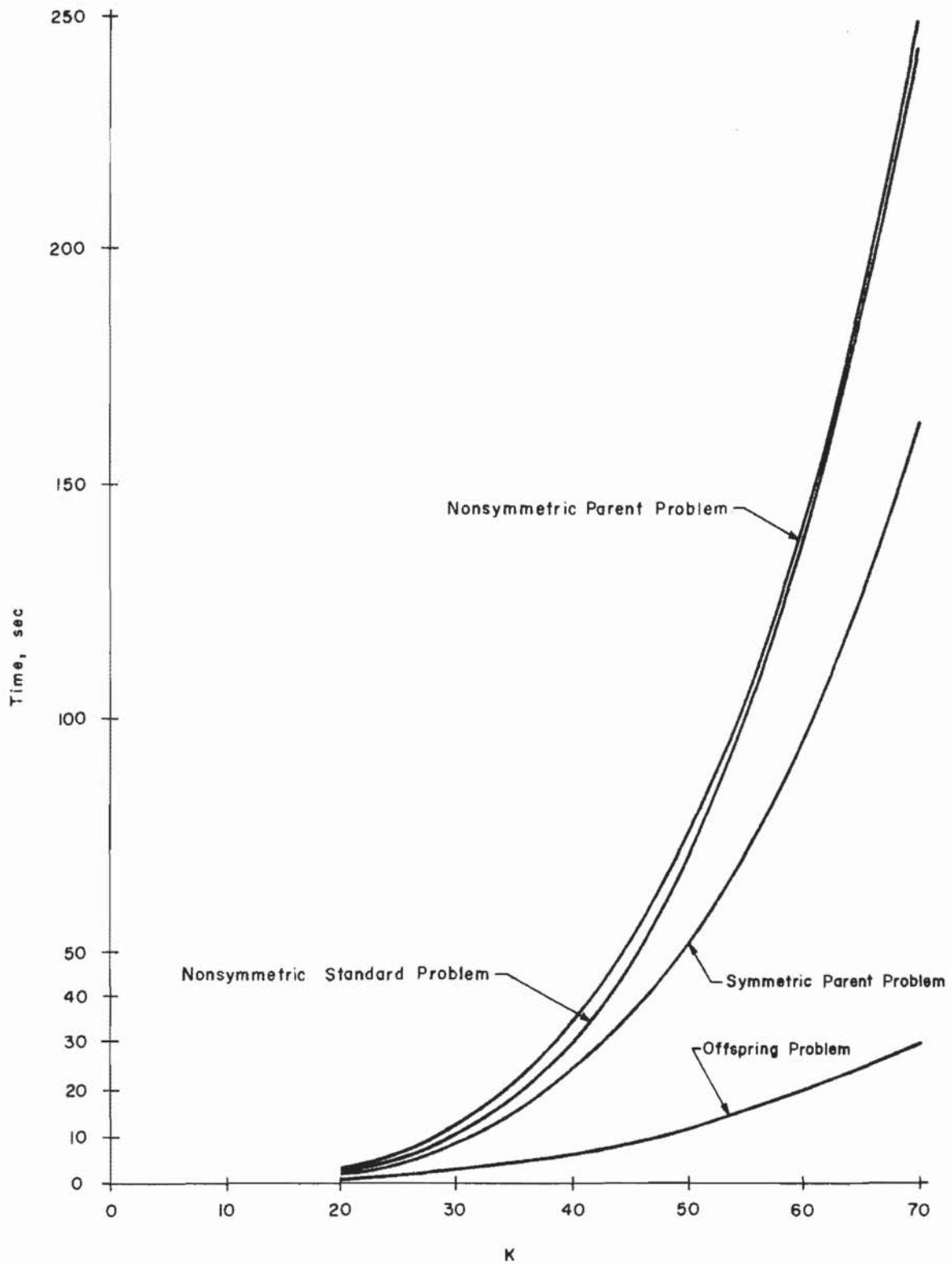


Fig 3.1. Time graph for three-wide solver.

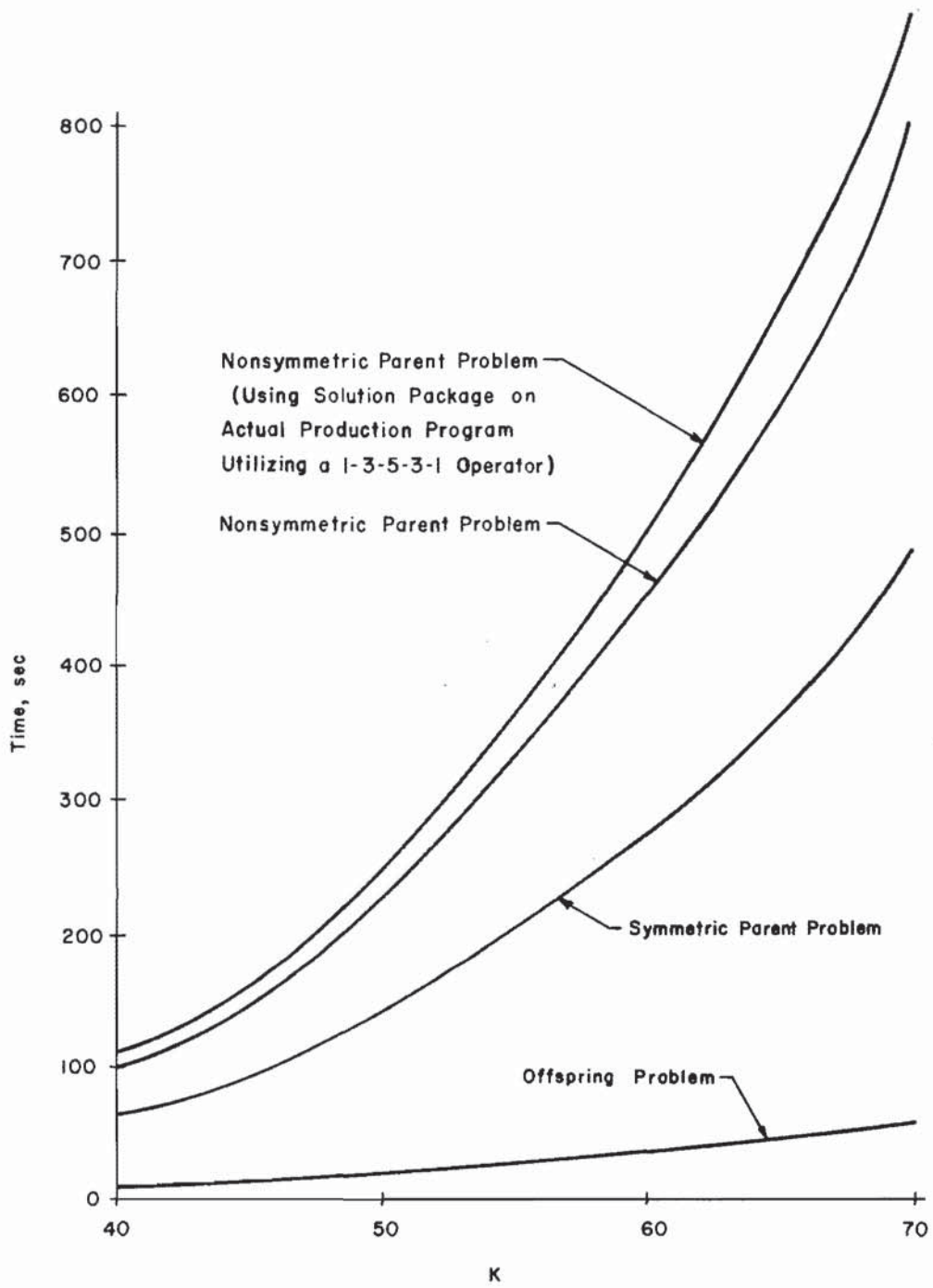


Fig 3.2. Time graph for five-wide solver.

This page replaces an intentionally blank page in the original.

-- CTR Library Digitization Team

REFERENCES

1. The American Standards Association, "American Standard FORTRAN," New York, 1966.
2. Panak, John J., and Hudson Matlock, "A Discrete-Element Method of Multiple-Loading Analysis for Two-Way Bridge Floor Slabs," Research Report No. 56-13, Center for Highway Research, The University of Texas at Austin, January 1970.
3. Stelzer, C. Fred, Jr., and W. Ronald Hudson, "A Direct Computer Solution for Plates and Pavement Slabs," Research Report No. 56-9, Center for Highway Research, The University of Texas, Austin, October 1967.

This page replaces an intentionally blank page in the original.

-- CTR Library Digitization Team

APPENDIX A

FLOW CHARTS AND LISTINGS OF MAIN ROUTINES

This page replaces an intentionally blank page in the original.

-- CTR Library Digitization Team

APPENDIX A. FLOW CHARTS AND LISTINGS OF MAIN ROUTINES

TRIP 3	-	Listing
TRIP 4	-	Listing
TRIP 3 & 4	-	Flow chart
FRIP 3	-	Listing
FRIP 4	-	Listing
FRIP 3 & 4	-	Flow chart

TRIP: Three-wide Recursion Inversion Procedure

FRIP: Five-wide Recursion Inversion Procedure


```

      SUBROUTINE TRIP3 ( L1,L2,L3,ML,A,AM1,ATM,C,D,          20MY8
1          BB,CC,DD,FF,W,N1,N2,N3 )                      20MY8
C*****          THE LATEST REVISION DATE FOR THIS PROGRAM IS - - - 01AG8
C***** THIS GROUP OF 10 SUBROUTINES PROVIDES THE USER WITH AN    23MR8
C          EFFICIENT GENERAL SPARSELY BANDED EQUATION SOLVER      11JA8
C          (THE MATRIX IS ASSUMED TO BE POSITIVE DEFINITE)        12MR8
C          WHICH CAN HANDLE UP TO 3 GROUPS OF BANDS , EACH        11JA8
C          OF ARBITRARY WIDTH                                     11JA8
C***** THIS ROUTINE SUPERVISES 9 SUBROUTINES , 8 OF WHICH        23MR8
C          ARE SELF-SUFFICIENT AND COME AS A PACKAGE( SUMP 6 ), THE 01AG8
C          REMAINING ONE GENERATES AND PACKS THE STIFFNESS        11JA8
C          MATRIX AS OUTLINED IN THE APPENDIX OF THE RELATED REPORT 23MR8
C          THIS ROUTINE MUST BE SUPPLIED BY THE USER SINCE       11JA8
C          IT DESCRIBES HIS PARTICULAR PROBLEM                   11JA8
C***** IN THE MAIN PROGRAM THE FOLLOWING CAN BE EQUIVALENCED    20MY8
C          ( ATM , CC )                                          20MY8
C***** SCRATCH TAPES SHOULD BE REQUESTED FOR TAPES 1 AND 2      05MR8
C          TAPE 3 WORKS APPROPRIATELY AS A DISK FILE , BUT A SCRATCH 20MY8
C          TAPE CAN BE USED IF NECESSARY OR DESIRED.            20MY8
      DIMENSION A(L1 ) , AM1(L1 ) , ATM(L1 ) ,              20MY8
1          C(L1,L1) , D(L1,L1) , W(L2,L3) ,                20MY8
2          BB(L1,N1) , CC(L1,N2) , DD(N3,L1) , FF( L1 )    08AP8
      COMMON /R1/ NK , NL , NF                               01FE8
      REWIND 1                                               11JA8
      REWIND 2                                               11JA8
      REWIND 3                                               18JA8
      IF( ML ) 140, 100, 100                                  11JA8
C          SET INITIAL CONDITIONS                               11JA8
100      DO 135 I = 1 , NK                                     01FE8
          DO 130 J = 1 , NK                                     01FE8
              C(I,J) = 0.0                                     12MR8
130      CONTINUE                                             11JA8
135      CONTINUE                                             11JA8
140      DO 150 I = 1 , NK                                     01FE8
          A(I) = 0.0                                          20MY8
150      CONTINUE                                             11JA8

```



```

C*****
C      BEGIN BACKWARD PASS -- COMPUTE RECURSION EQUATION
C*****
      BACKSPACE 1                                20MY8
      BACKSPACE 2                                20MY8
      CALL RFV ( W(NF,NL), A , L1 , 1 , NK )      01AG8
              NLM1 = NL - 1                       20MY8
C
      DO 2000 L = NF , NLM1                       20MY8 .
              J = NLM1 + NF - L                   20MY8 .
      BACKSPACE 1                                11JA8 .
      BACKSPACE 2                                18JA8 .
C      READ A COEFFICIENT FROM TAPE 1            11JA8 .
      READ (1) ( A(I), I = 1,NK )                01FE8 .
C      READ C COEFFICIENT FROM TAPE 2            12MR8 .
      READ (2) ( ( C(I,K) , I = 1,NK) , K = 1,NK ) 12MR8 .
      BACKSPACE 1                                11JA8 .
      BACKSPACE 2                                18JA8 .
      CALL MFFV ( C , W(NF,J+1),AM1, L1 , 1 , NK ) 01AG8 .
      CALL ASFV ( A , AM1, W(NF,J) , L1 , 1 , NK , +1 ) 01AG8 .
2000      CONTINUE                                11JA8 .
C
      RETURN . . . . .                          11JA8
      END                                          11JA8

```



```

SUBROUTINE TRIP4 ( L1,L2,L3,ML,A,AM1,ATM,C,D,          20MY8
1              DT1,CC,DD,FF,W,N1,N2 )                20MY8
C*****      THE LATEST REVISION DATE FOR THIS PROGRAM IS - - - 01AG8
C***** THIS GROUP OF 13 SUBROUTINES PROVIDES THE USER WITH AN 23MR8
C          EFFICIENT GENERAL SPARSELY BANDED EQUATION SOLVER    11JA8
C          (THE MATRIX IS ASSUMED TO BE SYMMETRIC AND POSITIVE DEFINITE) 12MR8
C          WHICH CAN HANDLE UP TO 3 GROUPS OF BANDS , EACH      11JA8
C          OF ARBITRARY WIDTH                                    11JA8
C***** THIS ROUTINE SUPERVISES 12 SUBROUTINES , 11 OF WHICH    23MR8
C          ARE SELF-SUFFICIENT AND COME AS A PACKAGE( SUMP 6 ), THE 01AG8
C          REMAINING ONE GENERATES AND PACKS THE STIFFNESS      11JA8
C          MATRIX AS OUTLINED IN THE APPENDIX OF THE RELATED REPORT 23MR8
C          THIS ROUTINE MUST BE SUPPLIED BY THE USER SINCE     11JA8
C          IT DESCRIBES HIS PARTICULAR PROBLEM                 11JA8
C***** IN THE MAIN PROGRAM THE FOLLOWING CAN BE EQUIVALENCED 20MY8
C          ( ATM , CC )                                         20MY8
C***** SCRATCH TAPLS SHOULD BE REQUESTED FOR TAPES 1 AND 2    05MR8
C          TAPE 3 WORKS APPROPRIATELY AS A DISK FILE , BUT A SCRATCH 20MY8
C          TAPE CAN BE USED IF NECESSARY OR DESIRED.           20MY8
          DIMENSION A(L1 ) , AM1(L1 ) , ATM(L1 ) ,
1              C(L1,L1) , D(L1,L1) , W(L2,L3) ,
2              DT1(L1,N1) , CC(L1,N2) , DD(N1,L1) , FF( L1 )
          COMMON /RI/ NK , NL , NF
          REWIND 1
          REWIND 2
          REWIND 3
          IF( ML ) 140, 100, 100
C          SET INITIAL CONDITIONS
100          DO 135 I = 1 , NK
              DO 130 J = 1 , NK
                  C(I,J) = 0.0
130          CONTINUE
135          CONTINUE
140          DO 150 I = 1 , NK
              A(I) = 0.0
150          CONTINUE

```

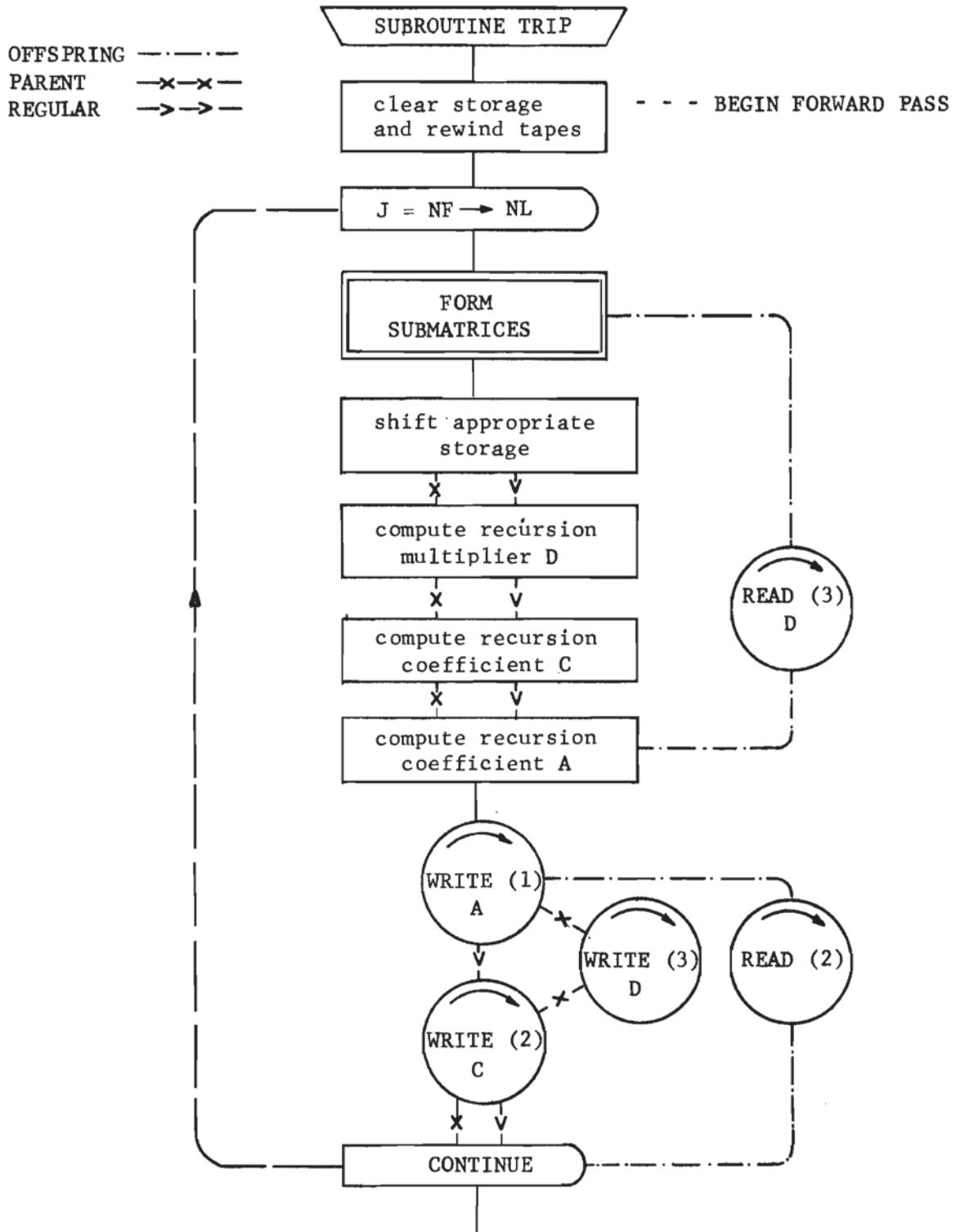


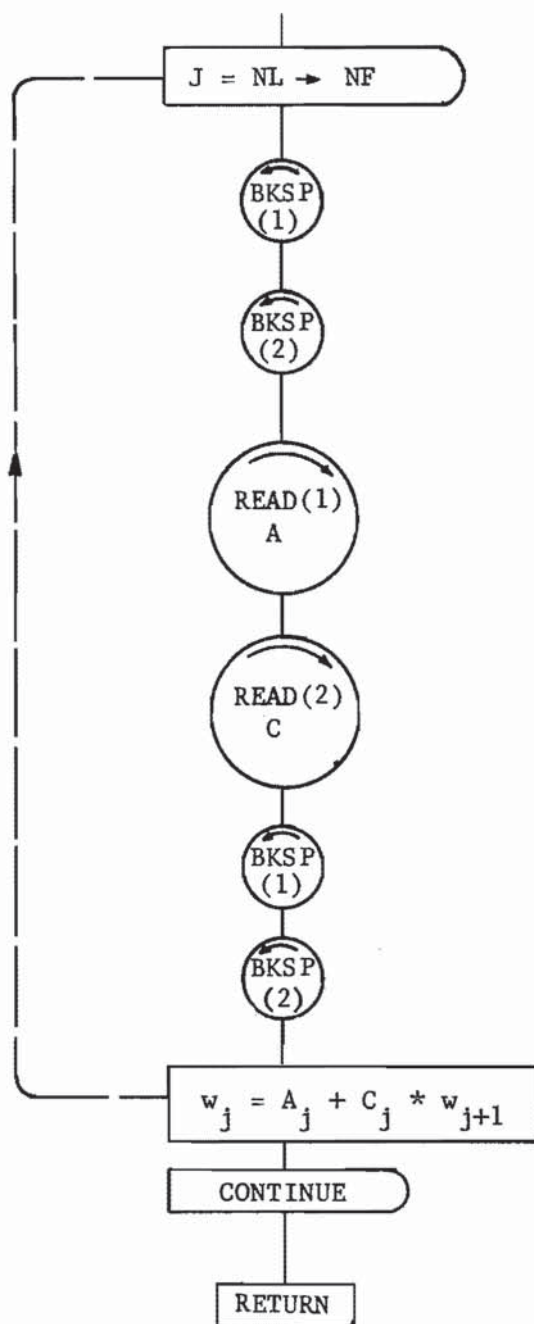
```

C*****
C      BEGIN BACKWARD PASS -- COMPUTE RECURSION EQUATION
C*****
      BACKSPACE 1                                20MY8
      BACKSPACE 2                                20MY8
      CALL RFV ( W(NF,NL), A , L1 , 1 , NK )      01AG8
              NLM1 = NL - 1                      20MY8
C      . . . . .
      DO 2000 L = NF , NLM1                       20MY8 .
              J = NLM1 + NF - L                  20MY8 .
      BACKSPACE 1                                11JA8 .
      BACKSPACE 2                                18JA8 .
C      READ A COEFFICIENT FROM TAPE 1            11JA8 .
      READ (1) ( A(1), I = 1,NK )                01FE8 .
C      READ C COEFFICIENT FROM TAPE 2            12MR8 .
      READ (2) ( ( C(1,K) , I = 1,NK) , K = 1,NK ) 12MR8 .
      BACKSPACE 1                                11JA8 .
      BACKSPACE 2                                18JA8 .
      CALL MFFV ( C , W(NF,J+1),AM1, L1 , 1 , NK ) 01AG8 .
      CALL ASFV ( A , AM1, W(NF,J) , L1 , 1 , NK , +1 ) 01AG8 .
2000 CONTINUE                                    11JA8 .
C      . . . . .
      RETURN                                     11JA8
      END                                         11JA8

```

Flow Chart for Three-Wide Recursion Inversion Process
 TRIP 3 , TRIP 4





C***** NOTATION FOR FRIP 3

C	A	-	RECURSION COEFFICIENT (A(I))	
C	AM1	-	RECURSION COEFFICIENT (A(I-1))	
C	AM2	-	RECURSION COEFFICIENT (A(I-2))	
C	ATM	-	TEMPORARY VECTOR	
C	B	-	RECURSION COEFFICIENT (B(I))	
C	BM1	-	RECURSION COEFFICIENT (B(I-1))	
C	BM2	-	RECURSION COEFFICIENT (B(I-2))	
C	C	-	RECURSION COEFFICIENT (C(I))	
C	CM1	-	RECURSION COEFFICIENT (C(I-1))	
C	CM2	-	RECURSION COEFFICIENT (C(I-2))	
C	D	-	RECURSION MULTIPLIER (D(I))	
C	E	-	RECURSION MULTIPLIER (E(I))	
C	AA	-	SUB-MATRIX (LITTLE A(I))	
C	BB	-	SUB-MATRIX (LITTLE B(I))	
C	CC	-	SUB-MATRIX (LITTLE C(I))	
C	DD	-	SUB-MATRIX (LITTLE D(I))	
C	EE	-	SUB-MATRIX (LITTLE E(I))	
C	FF	-	SUB-MATRIX VECTOR (LITTLE F(I))	
C	W	-	SOLUTION VECTOR (STORED AS TWO-DIMENSIONAL)	
C	N1	-	BAND WIDTH OF AA	
C	N2	-	BAND WIDTH OF BB	
C	N3	-	BAND WIDTH OF CC	
C	N4	-	BAND WIDTH OF DD	
C	N5	-	BAND WIDTH OF EE	
C	ML	-	PROBLEM TYPE SWITCH	NEGATIVE FOR OFFSPRING
C				ZERO FOR STANDARD
C				POSITIVE FOR PARENT
C	NK	-	ORDER OF SUBMATRICES	
C	NL	-	MATRIX ORDER OF OVERALL COEFFICIENT MATRIX	
C	NF	-	STARTING VALUE FOR MAIN DO LOOP	
C	L1	-	VARIABLE DIMENSION PARAMETER (REQUIRED)	
C	L2	-	VARIABLE DIMENSION PARAMETER (OPTIONAL)	
C	L3	-	VARIABLE DIMENSION PARAMETER (OPTIONAL)	

```

SUBROUTINE FRIP3 ( L1,L2,L3,ML,A,AM1,AM2,ATM,B,BM1,BM2,C,      20MY8
1          CM1,CM2,D,E,AA,BB,CC,DD,EE,FF,W,N1,N2,N3,N4,N5 ) 20MY8
C***** THE LATEST REVISION DATE FOR THIS PROGRAM IS - - - - 01AG8
C***** THIS GROUP OF 10 SUBROUTINES PROVIDES THE USER WITH AN 20MY8
C          EFFICIENT GENERAL SPARSELY BANDED EQUATION SOLVER    04JA8
C          (THE MATRIX IS ASSUMED TO BE POSITIVE DEFINITE)     12MR8
C          WHICH CAN HANDLE UP TO 5 GROUPS OF BANDS , EACH      04JA8
C          OF ARBITRARY WIDTH                                   04JA8
C***** THIS ROUTINE SUPERVISES 9 SUBROUTINES , 8 OF WHICH     20MY8
C          ARE SELF-SUFFICIENT AND COME AS A PACKAGE( SUMP 6 ), THE 01AG8
C          REMAINING ONE GENERATES AND PACKS THE STIFFNESS      04JA8
C          MATRIX AS OUTLINED IN THE APPENDIX OF THE RELATED REPORT 23MR8
C          THIS ROUTINE MUST BE SUPPLIED BY THE USER SINCE    04JA8
C          IT DESCRIBES HIS PARTICULAR PROBLEM                 04JA8
C***** IN THE MAIN PROGRAM THE FOLLOWING CAN BE EQUIVALENCED 20MY8
C          ( ATM , BB )                                         20MY8
C***** SCRATCH TAPES SHOULD BE REQUESTED FOR TAPES 1 AND 2   05MR8
C          TAPE 3 WORKS APPROPRIATELY AS A DISK FILE , BUT A SCRATCH 20MY8
C          TAPE CAN BE USED IF NECESSARY OR DESIRED.           20MY8
          DIMENSION A(L1 ) , AM1(L1 ) , AM2(L1 ) ,             20MY8
1          B(L1,L1) , BM1(L1,L1) , BM2(L1,L1) , ATM(L1 ) ,    20MY8
2          C(L1,L1) , CM1(L1,L1) , CM2(L1,L1) ,               20MY8
3          D(L1,L1) , E(L1,L1) , W(L2,L3) ,                   20MY8
4          AA(L1,N1) , BB(L1,N2) , CC(L1,N3) , DD(L1,N4) ,    04JA8
5          EE(N5,L1) , FF(L1) ,                                04JA8
          COMMON /RI/ NK , NL , NF                             02FE8
          REWIND 1                                             04JA8
          REWIND 2                                             04JA8
          REWIND 3                                             17JA8
          IF( ML ) 140, 100, 100                               04JA8
C          SET INITIAL CONDITIONS                               04JA8
100         DO 135 J = 1 , NK                                  01FE8
           DO 130 I = 1 , NK                                  01FE8
             BM1(I,J) = 0.0                                    04JA8
             CM1(I,J) = 0.0                                    04JA8
             B(I,J) = 0.0                                      04JA8
             C(I,J) = 0.0                                      04JA8
130         CONTINUE                                          04JA8
135         CONTINUE                                          04JA8
140         DO 150 I = 1 , NK                                  01FE8
           A(I) = 0.0                                          20MY8
           AM1(I) = 0.0                                        20MY8
150         CONTINUE                                          04JA8

```

```

C*****
C      BEGIN FORWARD PASS  --  SOLVE FOR RECURSION COEFFICIENTS      04JAB
C*****
C      . . . . .
C      DO 1000 J = NF , NL      01FE8 .
C          JJ = J      04JAB .
C      FORM SUB-MATRICES      04JAB .
C      CALL FSUB51 (L1,L2,L3,AA,BB,CC,DD,EE,FF,ML,JJ,N1,N2,N3,N4,N5 ) 12MR8 .
C      CALL RFV ( AM2, AM1, L1 , 1 , NK )      20MY8 .
C      CALL RFV ( AM1, A , L1 , 1 , NK )      20MY8 .
C          IF( ML ) 210, 180, 180      04JAB .
180 CALL RFV ( BM2, BM1, L1 , L1 , NK )      20MY8 .
C      CALL RFV ( BM1, B , L1 , L1 , NK )      20MY8 .
C      CALL RFV ( CM2, CM1, L1 , L1 , NK )      20MY8 .
C      CALL RFV ( CM1, C , L1 , L1 , NK )      20MY8 .
C          GO TO 220      04JAB .
C      READ D AND E MULTIPLIERS FROM TAPE 3      17JAB .
210 READ (3) (( D(I,K) , E(I,K) , I = 1,NK) , K = 1,NK )      01FE8 .
C          GO TO 280      04JAB .
C      CALCULATE RECURSION MULTIPLIER E      04JAB .
220 CALL MBFV ( AA , BM2, E , L1 , L1 , NK , N1 )      20MY8 .
C      CALL ABF ( BB , E , E , L1 , NK , N2 )      01FE8 .
C      CALCULATE RECURSION MULTIPLIER D      04JAB .
C      CALL MFFV ( E , BM1, D , L1 , L1 , NK )      20MY8 .
C      CALL MBFV ( AA , CM2, C , L1 , L1 , NK , N1 )      20MY8 .
C      CALL ASFV ( D , C , D , L1 , L1 , NK , +1 )      01AG8 .
C      CALL ABF ( CC , D , D , L1 , NK , N3 )      01FE8 .
C      CALL INVR5 ( D , L1 , NK )      01FE8 .
C      CALL CFV ( D , L1 , L1 , NK , -1. )      20MY8 .
C      CALCULATE RECURSION COEFFICIENT C      04JAB .
C      CALL MFB ( D , EE , C , L1 , NK , N5 )      01FE8 .
C      CALCULATE RECURSION COEFFICIENT B      04JAB .
C      CALL MFFV ( E , CM1, B , L1 , L1 , NK )      20MY8 .
C      CALL ABF ( DD , B , BM2, L1 , NK , N4 )      01FE8 .
C      CALL MFFV ( D , BM2, B , L1 , L1 , NK )      20MY8 .
C      CALCULATE RECURSION COEFFICIENT A      04JAB .
280 CALL MFFV ( E , AM1, A , L1 , 1 , NK )      20MY8 .
C      CALL MBFV ( AA , AM2, ATM, L1 , 1 , NK , N1 )      20MY8 .
C      CALL ASFV ( A , ATM, AM2, L1 , 1 , NK , +1 )      20MY8 .
C      CALL ASFV ( AM2, FF , ATM, L1 , 1 , NK , -1 )      20MY8 .
C      CALL MFFV ( D , ATM, A , L1 , 1 , NK )      20MY8 .
C      SAVE A COEFFICIENT ON TAPE 1      04JAB .
C      WRITE (1) ( A(I) , I = 1,NK )      02FE8 .
C          IF( ML ) 400,600,500      04JAB .
400 READ (2)      17JAB .
C          GO TO 1000      04JAB .
C      SAVE D AND E MULTIPLIERS ON TAPE 3      17JAB .
500 WRITE (3) (( D(I,K),E(I,K) , I=1,NK) , K=1,NK)      02FE8 .
C      SAVE B AND C COEFFICIENTS ON TAPE 2      17JAB .
600 WRITE (2) (( B(I,K),C(I,K) , I=1,NK) , K=1,NK)      02FE8 .
1000 CONTINUE      04JAB .
C      . . . . .

```

```

C*****
C      BEGIN BACKWARD PASS -- COMPUTE RECURSION EQUATION          04JAB
C*****
      BACKSPACE 1          20MY8
      BACKSPACE 2          20MY8
      CALL RFV ( W(NF,NL), A , L1 , 1 , NK )          01AG8
      BACKSPACE 1          20MY8
      BACKSPACE 2          20MY8
      READ (1) ( A(I), I = 1,NK )          20MY8
      READ (2) (( B(I,K) , C(I,K), I = 1,NK) , K = 1,NK)          20MY8
      BACKSPACE 1          20MY8
      BACKSPACE 2          20MY8
      CALL MFFV ( B , W(NF,NL), AM1, L1 , 1 , NK )          01AG8
      CALL ASFV ( A , AM1, W(NF,NL-1) , L1 , 1 , NK , +1 )          01AG8
      NLM2 = NL - 2          20MY8
C
      DO 2000 L = NF , NLM2          20MY8
      J = NLM2 + NF - L          .
      BACKSPACE 1          04JAB
      BACKSPACE 2          17JAB
C      READ A COEFFICIENT FROM TAPE 1          04JAB
      READ (1) ( A(I), I = 1,NK )          02FEB
C      READ B AND C COEFFICIENTS FROM TAPE 2          17JAB
      READ (2) (( B(I,K) , C(I,K), I = 1,NK) , K = 1,NK)          02FEB
      BACKSPACE 1          04JAB
      BACKSPACE 2          17JAB
      CALL MFFV ( B , W(NF,J+1),AM1, L1 , 1 , NK )          01AG8
      CALL MFFV ( C , W(NF,J+2),AM2, L1 , 1 , NK )          01AG8
      CALL ASFV ( AM1, AM2, AM1, L1 , 1 , NK , +1 )          20MY8
      CALL ASFV ( A , AM1, W(NF,J) , L1 , 1 , NK , +1 )          01AG8
2000 CONTINUE          04JAB
C
      RETURN          4JAB
      END          4JAB

```



```

SUBROUTINE FRIP4 ( L1,L2,L3,ML,A,AM1,AM2,ATM,B,BM1,EP1,C,CM1, 20MY8
1 D,E,ET2,ET1,CC,DT,EE,FF,W,N1,N2,N3 ) 27MY8
C***** THE LATEST REVISION DATE FOR THIS PROGRAM IS - - - - 01AG8
C***** THIS GROUP OF 15 SUBROUTINES PROVIDES THE USER WITH AN 20MY8
C EFFICIENT GENERAL SPARSELY BANDED EQUATION SOLVER 04JA8
C (THE MATRIX IS ASSUMED TO BE SYMMETRIC AND POSITIVE DEFINITE) 12MR8
C WHICH CAN HANDLE UP TO 5 GROUPS OF BANDS , EACH 04JA8
C OF ARBITRARY WIDTH 04JA8
C***** THIS ROUTINE SUPERVISES 14 SUBROUTINES , 13 OF WHICH 20MY8
C ARE SELF-SUFFICIENT AND COME AS A PACKAGE( SUMP 6 ), THE 01AG8
C REMAINING ONE GENERATES AND PACKS THE STIFFNESS 04JA8
C***** MATRIX AS OUTLINED IN IN THE APPENDIX OF THE RELATED REPORT 23MR8
C THIS ROUTINE MUST BE SUPPLIED BY THE USER SINCE 04JA8
C IT DESCRIBES HIS PARTICULAR PROBLEM 04JA8
C***** IN THE MAIN PROGRAM THE FOLLOWING CAN BE EQUIVALENCED 20MY8
C ( ATM , DT ) 20MY8
C***** SCRATCH TAPES SHOULD BE REQUESTED FOR TAPES 1 AND 2 05MR8
C TAPE 3 WORKS APPROPRIATELY AS A DISK FILE , BUT A SCRATCH 20MY8
C TAPE CAN BE USED IF NECESSARY OR DESIRED. 20MY8
DIMENSION A(L1 ) , AM1(L1 ) , AM2(L1 ) , 20MY8
1 B(L1,L1) , BM1(L1,L1) , EP1(L1,L1) , ATM(L1 ) , 20MY8
2 C(L1,L1) , CM1(L1,L1) , D(L1,L1) , 20MY8
3 E(L1,L1) , W(L2,L3) , ET2(L1,N1) , 20MY8
4 DT(L1,N2) , CC(L1,N3) , ET1(L1,N1) , EE(N1,L1) , 23MR8
5 FF(L1) 23MR8
COMMON /RI/ NK , NL , NF 02FEB
REWIND 1 04JA8
REWIND 2 04JA8
REWIND 3 17JA8
IF( ML ) 140, 100, 100 04JA8
C SET INITIAL CONDITIONS 04JA8
100 DO 135 J = 1 , NK 01FEB
DO 130 I = 1 , NK 01FEB
B(I,J) = 0.0 04JA8
C(I,J) = 0.0 04JA8
CM1(I,J) = 0.0 04JA8
EP1(I,J) = 0.0 23MR8
130 CONTINUE 04JA8
135 CONTINUE 04JA8
140 DO 150 I = 1 , NK 01FEB
A(I) = 0.0 20MY8
AM1(I) = 0.0 20MY8
150 CONTINUE 04JA8

```

```

C*****
C      BEGIN FORWARD PASS  --  SOLVE FOR RECURSION COEFFICIENTS      04JA8
C*****
C      . . . . .
C      DO 1000  J = NF , NL      01FE8 .
C          JJ = J                04JA8 .
C      FORM SUB-MATRICES      04JA8 .
C      CALL FSUB52 ( L1,L2,L3,ET2,ET1,CC,DT,EE,FF,ML,JJ,N1,N2,N3 ) 27MY8 .
C      CALL RFV   ( AM2, AM1, L1 , 1 , NK )      20MY8 .
C      CALL RFV   ( AM1, A , L1 , 1 , NK )      20MY8 .
C          IF( ML ) 210, 180, 180      04JA8 .
C      180 CALL RFV ( BM1, B , L1 , L1 , NK )    20MY8 .
C          GO TO 220      04JA8 .
C      READ D AND E MULTIPLIERS FROM TAPE 3      17JA8 .
C      210 READ (3) (( D(I,K) , E(I,K) , I = 1,NK) , K = 1,NK ) 01FE8 .
C          GO TO 280      04JA8 .
C      CALCULATE RECURSION MULTIPLIER E      04JA8 .
C      220 CALL RFV ( E , EP1, L1 , L1 , NK )    20MY8 .
C      CALCULATE RECURSION MULTIPLIER EP1    23MR8 .
C      CALL MBFV ( ET1, BM1, EP1, L1 , L1 , NK , N1 ) 20MY8 .
C      CALL ABF ( DT , EP1, EP1, L1 , NK , N2 ) 23MR8 .
C      CALCULATE RECURSION MULTIPLIER D      04JA8 .
C      CALL SMFF ( E , BM1, D , L1 , NK )      05MR8 .
C      CALL RFV ( BM1, CM1, L1 , L1 , NK )    20MY8 .
C      CALL RFV ( CM1, C , L1 , L1 , NK )    20MY8 .
C      CALL MBFV ( ET2, BM1, C , L1 , L1 , NK , N1 ) 20MY8 .
C      CALL ASFV ( D , C , D , L1 , L1 , NK , +1 ) 20MY8 .
C      CALL ABF ( CC , D , D , L1 , NK , 'N3 ) 01FE8 .
C      CALL INVR6 ( D , L1 , NK )            15MR8 .
C      CALL CFV ( D , L1 , L1 , NK , -1 )    20MY8 .
C      CALCULATE RECURSION COEFFICIENT C      04JA8 .
C      CALL MFB ( D , EE , C , L1 , NK , N1 ) 20MR8 .
C      CALCULATE RECURSION COEFFICIENT B      04JA8 .
C      CALL MFFT ( D , EP1, B , L1 , NK )    23MR8 .
C      CALCULATE RECURSION COEFFICIENT A      04JA8 .
C      280 CALL MFFV ( E , AM1, A , L1 , 1 , NK ) 20MY8 .
C      CALL MBFV ( ET2, AM2, ATM, L1 , 1 , NK , N1 ) 20MY8 .
C      CALL ASFV ( A , ATM, AM2, L1 , 1 , NK , +1 ) 20MY8 .
C      CALL ASFV ( AM2, FF , ATM, L1 , 1 , NK , -1 ) 20MY8 .
C      CALL MFFV ( D , ATM, A , L1 , 1 , NK ) 20MY8 .
C      SAVE A COEFFICIENT ON TAPE 1      04JA8 .
C      WRITE (1) ( A(I) , I = 1,NK )      02FE8 .
C          IF( ML )400,600,500      04JA8 .
C      400 READ (2)      17JA8 .
C          GO TO 1000      4JA8 .
C      SAVE D AND E MULTIPLIERS ON TAPE 3      17JA8 .
C      500 WRITE (3) (( D(I,K),E(I,K), I=1,NK), K=1,NK) 02FE8 .
C      SAVE B AND C COEFFICIENTS ON TAPE 2    17JA8 .
C      600 WRITE (2) (( B(I,K),C(I,K), I=1,NK), K=1,NK) 02FE8 .
C      1000 CONTINUE      04JA8 .
C      . . . . .

```



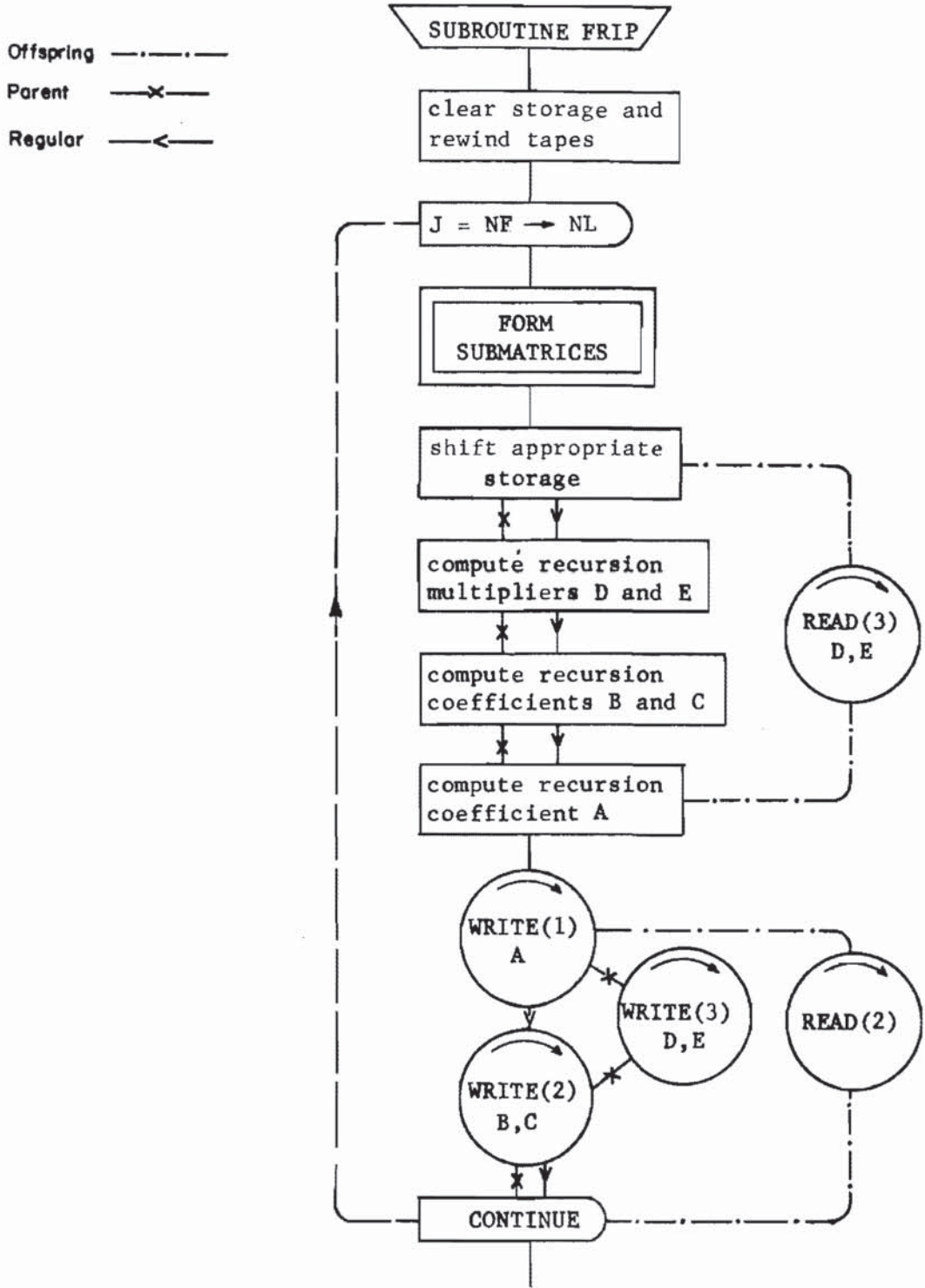
```

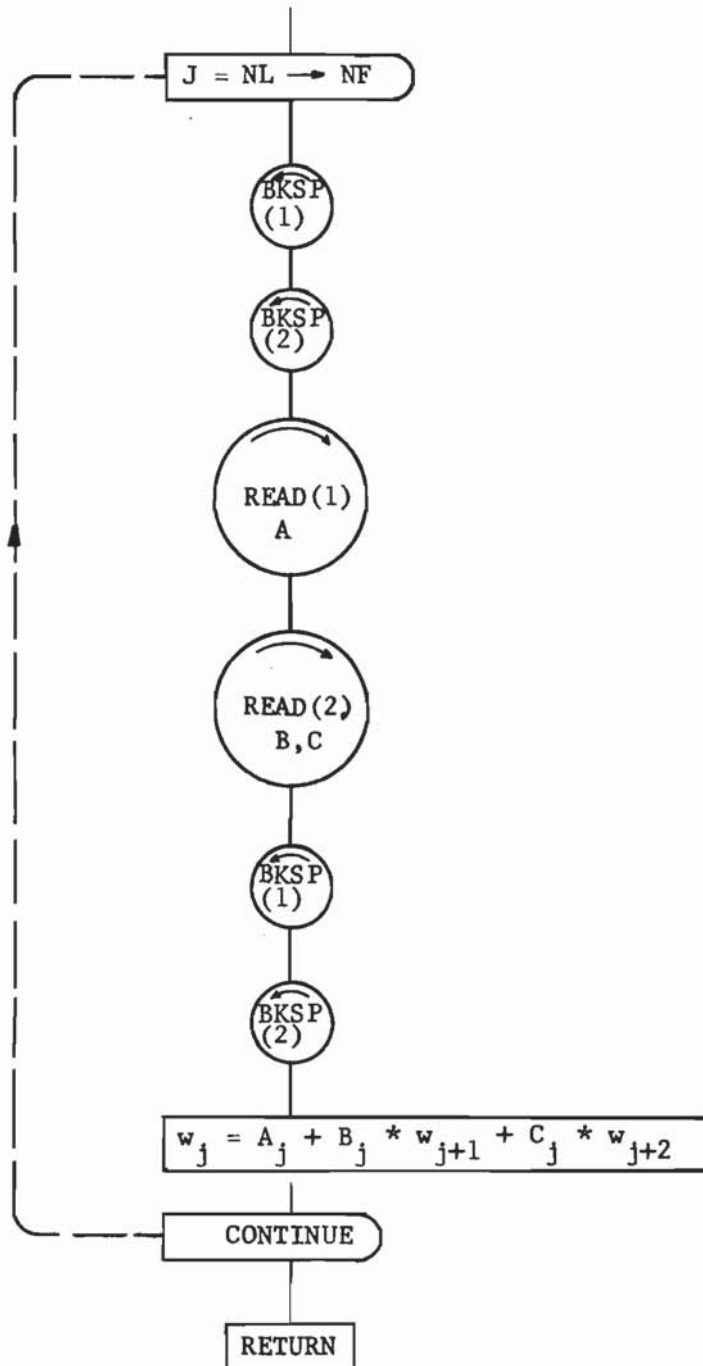
C*****
C      BEGIN BACKWARD PASS -- COMPUTE RECURSION EQUATION          04JAB
C*****
      BACKSPACE 1          20MY8
      BACKSPACE 2          20MY8
      CALL RFV ( W(NF,NL), A , L1 , 1 , NK )          01AG8
      BACKSPACE 1          20MY8
      BACKSPACE 2          20MY8
      READ (1) ( A(I), I = 1,NK )          20MY8
      READ (2) (( B(I,K) , C(I,K), I = 1,NK) , K = 1,NK)          20MY8
      BACKSPACE 1          20MY8
      BACKSPACE 2          20MY8
      CALL MFFV ( B , W(NF,NL), AM1, L1 , 1 , NK )          01AG8
      CALL ASFV ( A , AM1, W(NF,NL-1) , L1 , 1 , NK , +1 )          01AG8
      NLM2 = NL - 2          20MY8
C      . . . . .
      DO 2000 L = NF , NLM2          20MY8
      J = NLM2 + NF - L          20MY8
      BACKSPACE 1          04JAB
      BACKSPACE 2          17JAB
C      READ A COEFFICIENT FROM TAPE 1          04JAB
      READ (1) ( A(I), I = 1,NK )          02FE8
C      READ B AND C COEFFICIENTS FROM TAPE 2          17JAB
      READ (2) (( B(I,K) , C(I,K), I = 1,NK) , K = 1,NK)          02FE8
      BACKSPACE 1          04JAB
      BACKSPACE 2          17JAB
      CALL MFFV ( B , W(NF,J+1),AM1, L1 , 1 , NK )          01AG8
      CALL MFFV ( C , W(NF,J+2),AM2, L1 , 1 , NK )          01AG8
      CALL ASFV ( AM1, AM2, AM1, L1 , 1 , NK , +1 )          20MY8
      CALL ASFV ( A , AM1, W(NF,J) , L1 , 1 , NK , +1 )          01AG8
2000 CONTINUE          04JAB
C      . . . . .
      RETURN          4JAB
      END          4JAB

```

Flow Chart for Five-Wide Recursion Inversion Process

FRIP 3 , FRIP 4





This page replaces an intentionally blank page in the original.

-- CTR Library Digitization Team

APPENDIX B

FLOW CHARTS, LISTINGS, AND COMMENTS ON SUBORDINATE
ROUTINES FOR MATRIX MANIPULATIONS

This page replaces an intentionally blank page in the original.

-- CTR Library Digitization Team

APPENDIX B. FLOW CHARTS, LISTINGS, AND COMMENTS ON SUBORDINATE
SUBPROGRAMS FOR MATRIX MANIPULATIONS

Listings and Flow Charts of Subordinate Subroutines*

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

*In all the above subroutines, all matrices are either square or column vectors, except in the banded routines where banded square matrices are represented by packed matrices that are $K \times J$ where K is the order of the square matrix, and J is the band width.

```

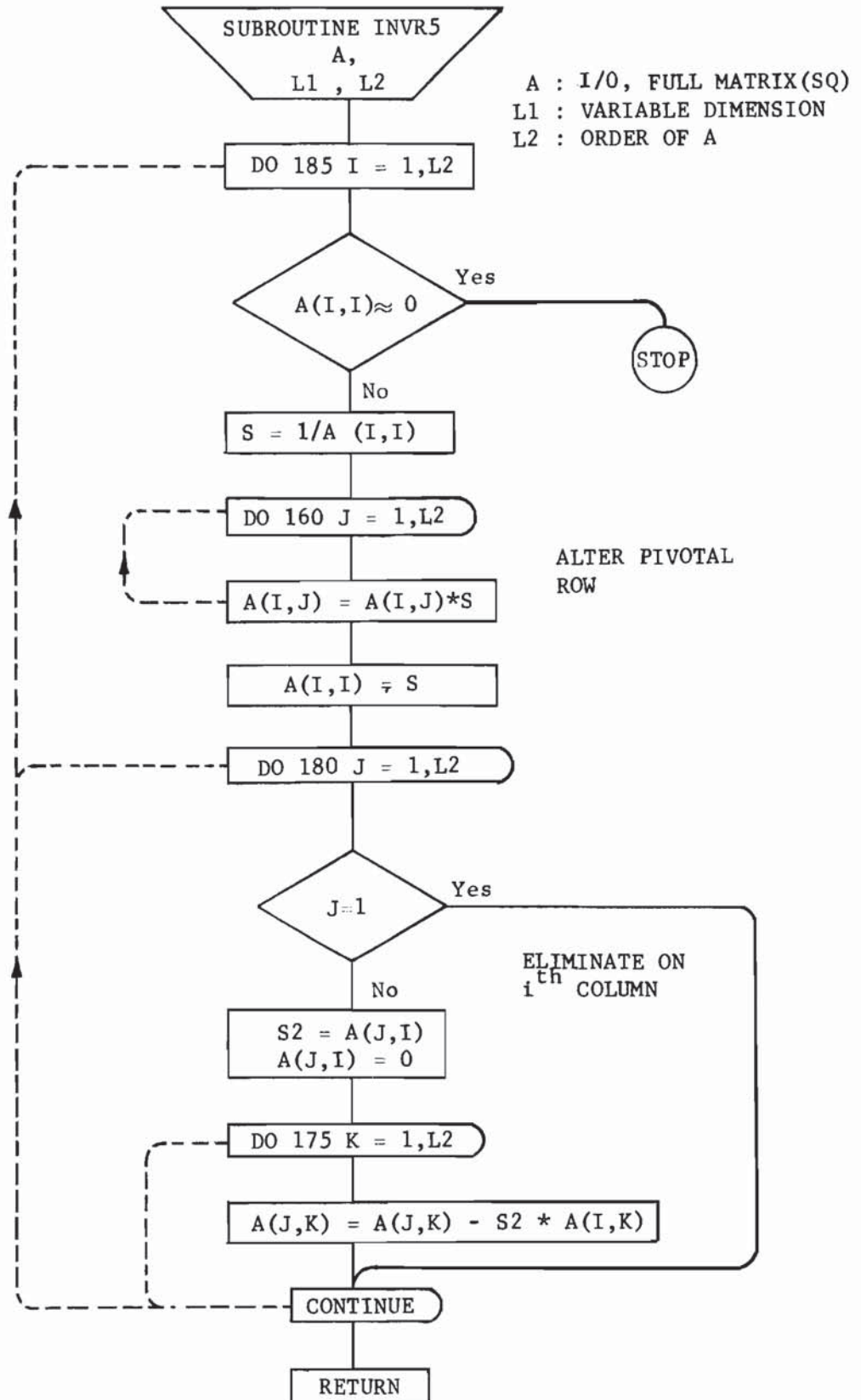
SUBROUTINE INVR5 ( A , L1 , L2 )
C***** THIS ROUTINE TAKES THE INVERSE OF A GENERAL POSTIVE
C          DEFINITE MATRIX , A SUPERIMPOSED AUGMENTED MATRIX METHOD
C          IS EMPLOYED
          DIMENSION A(L1,L1)
          20 FORMAT ( /85X,* NON-POSITIVE DEFINITE MATRIX ENCOUNTERED * )
          30 FORMAT ( 1X,10E10.3 )
          EP = 1.0E-10
C          .....
          DO 185 I = 1 , L2
          IF ( ABS(A(I,I) ) - EP ) 990, 990, 150
150         S = 1.0 / A(I,I)
          DO 160 J = 1 , L2
          A(I,J) = A(I,J) * S
160        CONTINUE
          A(I,I) = S
          DO 180 J = 1 , L2
          IF ( J-I ) 170, 180, 170
170         S = A(J,I)
          A(J,I) = 0.0
          DO 175 K = 1 , L2
          A(J,K) = A(J,K) - S * A(I,K)
175        CONTINUE
180        CONTINUE
185        CONTINUE
C          .....
          RETURN
          990 PRINT 20
          PRINT 30,(( A(I,J),J=1,L2),I=1,L2)
          END

```

```

31JA8
05MR8
05MR8
05MR8
28DE7
08AP8
100C7
100C7
.....
100C7 ..
31JA8 .
06FE8 .
28DE7 ..
100C7 ..
100C7 ..
28DE7 .
100C7 ..
100C7 ..
06FE8 ..
28DE7 ..
28DE7...
06FE8...
100C7...
100C7 ..
100C7 .
.....
100C7
100C7
28DE7
100C7

```


```

SUBROUTINE DCOM1 ( X , L1 , L2 )
DIMENSION X(L1,L1) , T(100)
DOUBLE PRECISION S , S1
C*****
C
C***** CAUTION - THE ACCURATE ACCUMULATION OF INNER PRODUCTS IS
C          BEING ATTEMPTED THRU THE DOUBLE PRECISIONING OF THE VARIABLES
C          S AND S1 . CARE SHOULD BE TAKEN TO INSURE THIS IS DONE PROPERLY
C*****
10 FORMAT ( /85X,* NON-POSITIVE DEFINITE MATRIX ENCOUNTERED * )
15 FORMAT ( /,5X,13E10.3 )
DO 20 I = 1 , L2
    T(I) = X(I,I)
20 CONTINUE
    IF ( X(1,1) .LE. 0.0 ) GO TO 4000
    S1 = X(1,1)
    S1 = DSQRT( S1 )
    X(1,1) = S1
    S1 = 1.0 / S1
DO 50 I = 2 , L2
    X(I,1) = X(I,1) * S1
50 CONTINUE
    L2M1 = L2 - 1
C
    DO 200 J = 2 , L2M1
        S = 0.0
        JM1 = J - 1
DO 120 K = 1 , JM1
    S = S + X(J,K) * X(J,K) * 1.0D0
120 CONTINUE
    IF ( X(J,J) .LE. S ) GO TO 4000
    S1 = X(J,J) - S
    S1 = DSQRT( S1 )
    X(J,J) = S1
    S1 = 1.0 / S1
    JP1 = J + 1
DO 190 I = JP1 , L2
    S = 0.0
DO 180 K = 1 , JM1
    S = S + X(I,K) * X(J,K) * 1.0D0
180 CONTINUE
    X(I,J) = ( X(I,J) - S ) * S1
190 CONTINUE
200 CONTINUE
C
    S = 0.0
DO 250 K = 1 , L2M1
    S = S + X(L2,K) * X(L2,K) * 1.0D0
250 CONTINUE
    S = X(L2,L2) - S
    IF ( S .LE. 0.0 ) GO TO 4000
    X(L2,L2) = DSQRT( S )
RETURN
4000 PRINT 10
    X(1,1) = T(1)

```

```
DO 400 I = 2 , L2
      K = I - 1
      X(I,I) = Y(I)
DO 350 J = 1 , K
      X(I,J) = X(J,I)
350  CONTINUE
400  CONTINUE
DO 500 I = 1 , L2
500 PRINT 15, ( X(I,J), J = 1,L2 )
      END
```

```
09AP8
12MR8
12MR8
12MR8
12MR8
12MR8
12MR8
20MR9
20MR9
19FE8
```

```

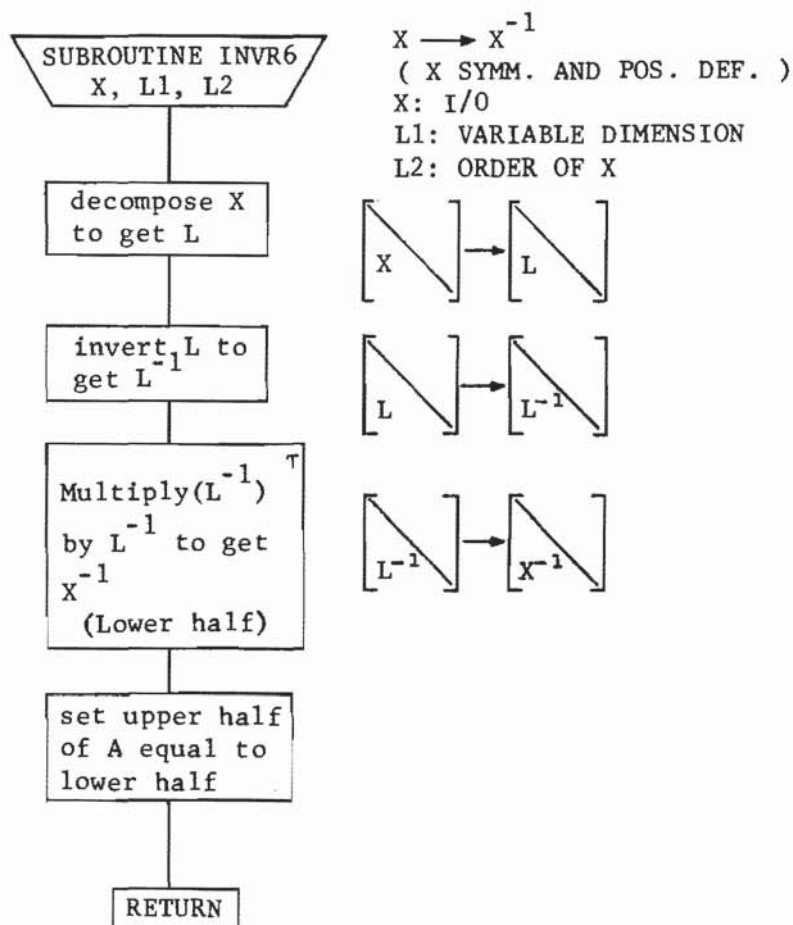
SUBROUTINE INVL1 ( X , L1 , L2 )                19FE8
DIMENSION X(L1,L1)                            19FE8
DOURLE PRECISION SUM                          29MY8
C*****B
C
C***** CAUTION - THE ACCURATE ACCUMULATION OF INNER PRODUCTS IS 29MY8 E
C BEING ATTEMPTED THRU THE DOUBLE PRECISIONING OF THE VARIABLE 29MY8 W
C SUM . CARE SHOULD BE TAKEN TO INSURE THIS IS DONE PROPERLY . 29MY8 A
C*****E
DO 50 I = 1 , L2                               19FE8
    X(I,I) = 1.0 / X(I,I)                       19FF8
50 CONTINUE                                    19FE8
    L2M1 = L2 - 1                               19FE8
DO 200 J = 1 , L2M1                            19FE8
    JP1 = J+1                                   19FE8
DO 150 I = JP1 , L2                            19FE8
    IM1 = I-1                                   19FE8
    SUM = 0.0                                   19FE8
DO 120 K = J , IM1                             19FE8
    SUM = SUM - X(I,K) * X(K,J) * 1.0D0        24JF8
120 CONTINUE                                    19FE8
    X(I,J) = X(I,I) * SUM                      19FF8
150 CONTINUE                                    19FE8
200 CONTINUE                                    19FE8
RETURN                                          19FE8
END                                              19FE8

```

```

SUBROUTINE MLTXL ( X , L1 , L2 )                19FE8
DIMENSION X(L1,L1)                             19FE8
DOUBLE PRECISION SUM                           29MY8
C*****B
C
C***** CAUTION - THE ACCURATE ACCUMULATION OF INNER PRODUCTS IS 29MY8 E
C      BEING ATTEMPTED THRU THE DOUBLE PRECISIONING OF THE VARIABLE 29MY8 W
C      SUM . CARE SHOULD BE TAKED TO INSURE THIS IS DONE PROPERLY . 29MY8 A
C*****E
      DO 200 I = 1 , L2                          19FE8
      DO 150 J = 1 , I                            19FE8
          SUM = 0.0                               19FE8
      DO 100 K = I , L2                          19FE8
          SUM = SUM + X(K,I) * X(K,J) * 1.0D0    24JE8
100    CONTINUE                                  19FE8
          X(I,J) = SUM                            19FE8
150    CONTINUE                                  19FE8
200    CONTINUE                                  19FE8
      RETURN                                     19FE8
      END                                         19FE8

```



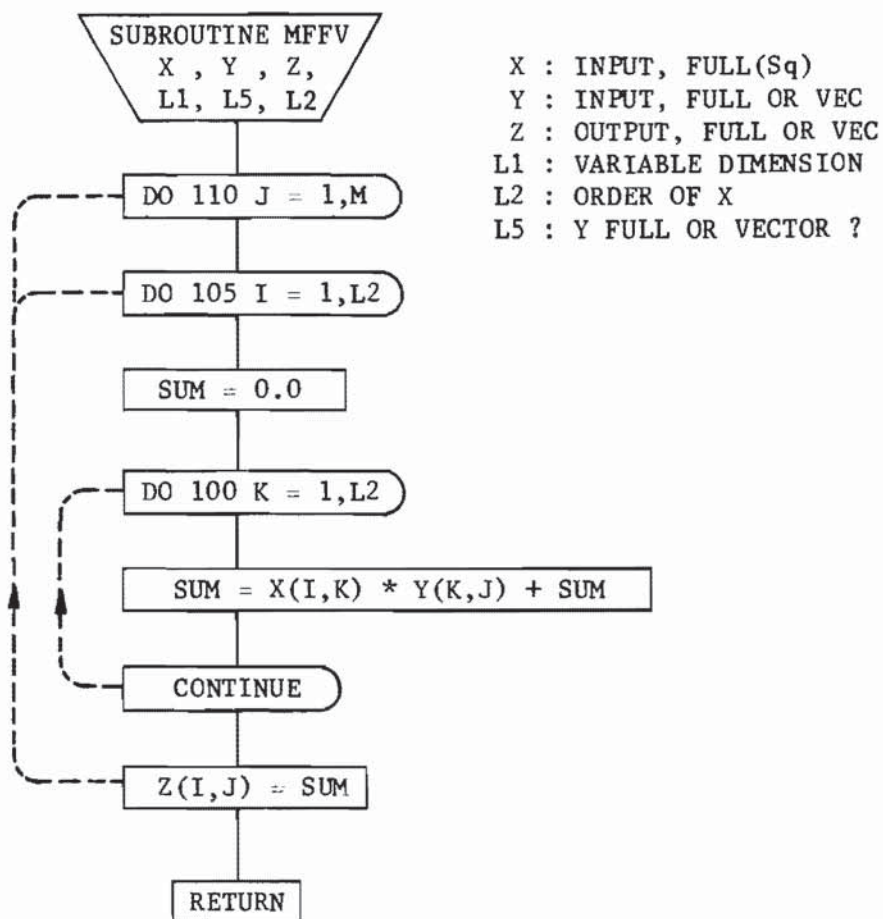
$$X = L \cdot L^T$$

$$X^{-1} = (L \cdot L^T)^{-1} = (L^T)^{-1} \cdot L^{-1} = (L^{-1})^T \cdot L^{-1}$$

```

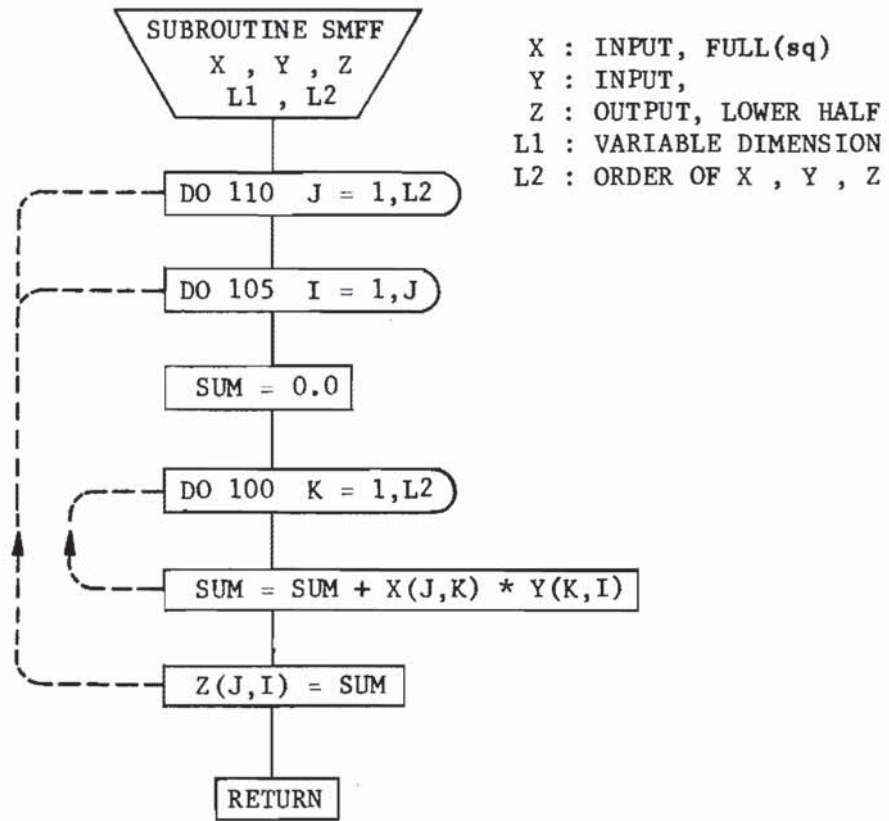
SUBROUTINE MFFV ( X , Y , Z , L1 , L5 , L2 )           03MR8
C***** THIS ROUTINE MULTIPLIES A FULL MATRIX       13DF7
C          TIMES A FULL MATRIX OR A VECTOR          13DF7
C          ( X * Y = Z )                             13DE7
          DIMENSION X(L1,L1) , Y(L1,L5) , Z(L1,L5)  13DE7
          DOUBLE PRECISION SUM                       29MY8
C*****B
C          29MY8 E
C***** CAUTION - THE ACCURATE ACCUMULATION OF INNER PRODUCTS IS 29MY8 W
C          BEING ATTEMPTED THRU THE DOUBLE PRECISIONING OF THE VARIABLE 29MY8 A
C          SUM . CARE SHOULD BE TAKED TO INSURE THIS IS DONE PROPERLY . 29MY8 R
C*****E
          M = 1                                       20MY8
          IF( L1 .EQ. L5 ) M = L2                     20MY8
          DO 110 J = 1,M                               13DF7
          DO 105 I = 1,L2                              13DE7
              SUM = 0.0                                03MR8
          DO 100 K = 1,L2                              13DE7
              SUM = SUM + X(I,K) * Y(K,J) * 1.0D0     24JF8
100      CONTINUE                                     13DE7
              Z(I,J) = SUM                             03MR8
105      CONTINUE                                     13DE7
110      CONTINUE                                     13DE7
          RETURN                                       13DF7
          END                                          13DE7

```

$$\begin{bmatrix} X \end{bmatrix} \cdot \begin{bmatrix} Y \end{bmatrix} = \begin{bmatrix} Z \end{bmatrix} \quad L5 = L1$$

$$\begin{bmatrix} X \end{bmatrix} \cdot \begin{matrix} \text{OR} \\ \begin{bmatrix} Y \end{bmatrix} \end{matrix} = \begin{bmatrix} Z \end{bmatrix} \quad L5 = 1$$

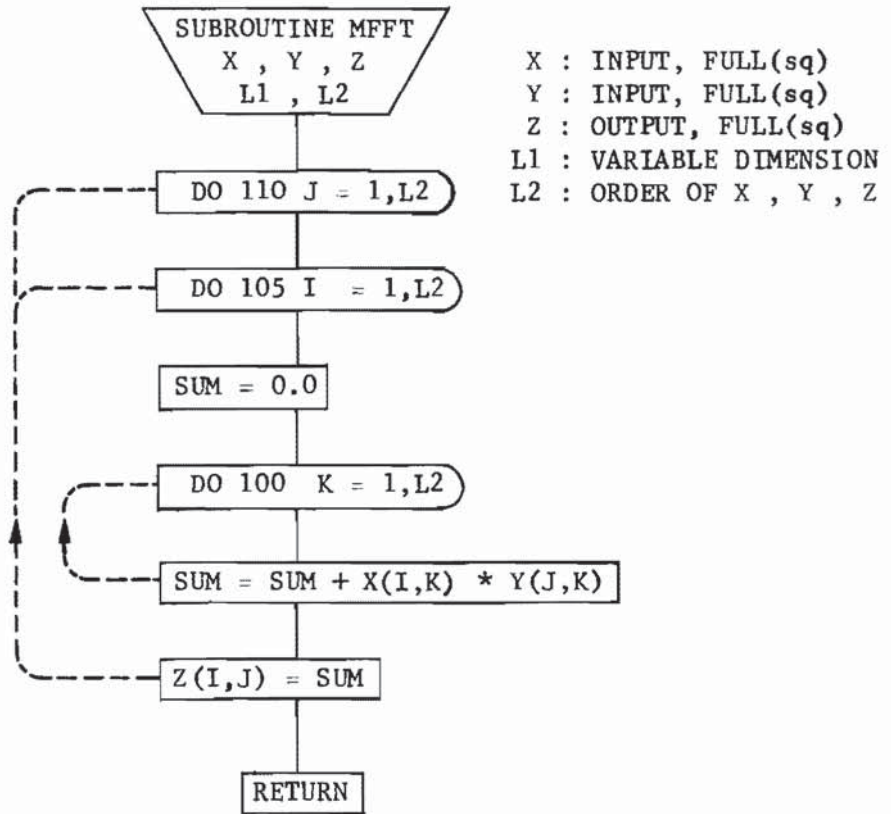


$$\begin{bmatrix} X \end{bmatrix} \cdot \begin{bmatrix} Y \end{bmatrix} = \begin{bmatrix} Z \end{bmatrix}$$

```

SUBROUTINE MFFT ( X , Y , Z , L1 , L2 )
C***** THIS ROUTINE MULTIPLIES A FULL MATRIX
C      TIMES THE TRANSPOSE OF A SECOND FULL MATRIX
C      ( X * YT = Z )
      DIMENSION X(L1,L1) , Y(L1,L1) , Z(L1,L1)
      DOUBLE PRECISION SUM
C*****B
C
C***** CAUTION - THE ACCURATE ACCUMULATION OF INNER PRODUCTS IS
C      BEING ATTEMPTED THRU THE DOUBLE PRECISIONING OF THE VARIABLE
C      SUM . CARE SHOULD BE TAKED TO INSURE THIS IS DONE PROPERLY .
C*****E
      DO 110 J = 1 , L2
      DO 105 I = 1 , L2
          SUM = 0.0
      DO 100 K = 1 , L2
          SUM = SUM + X(I,K) * Y(J,K) * 1.0D0
100      CONTINUE
          Z(I,J) = SUM
105      CONTINUE
110      CONTINUE
      RETURN
      END

```

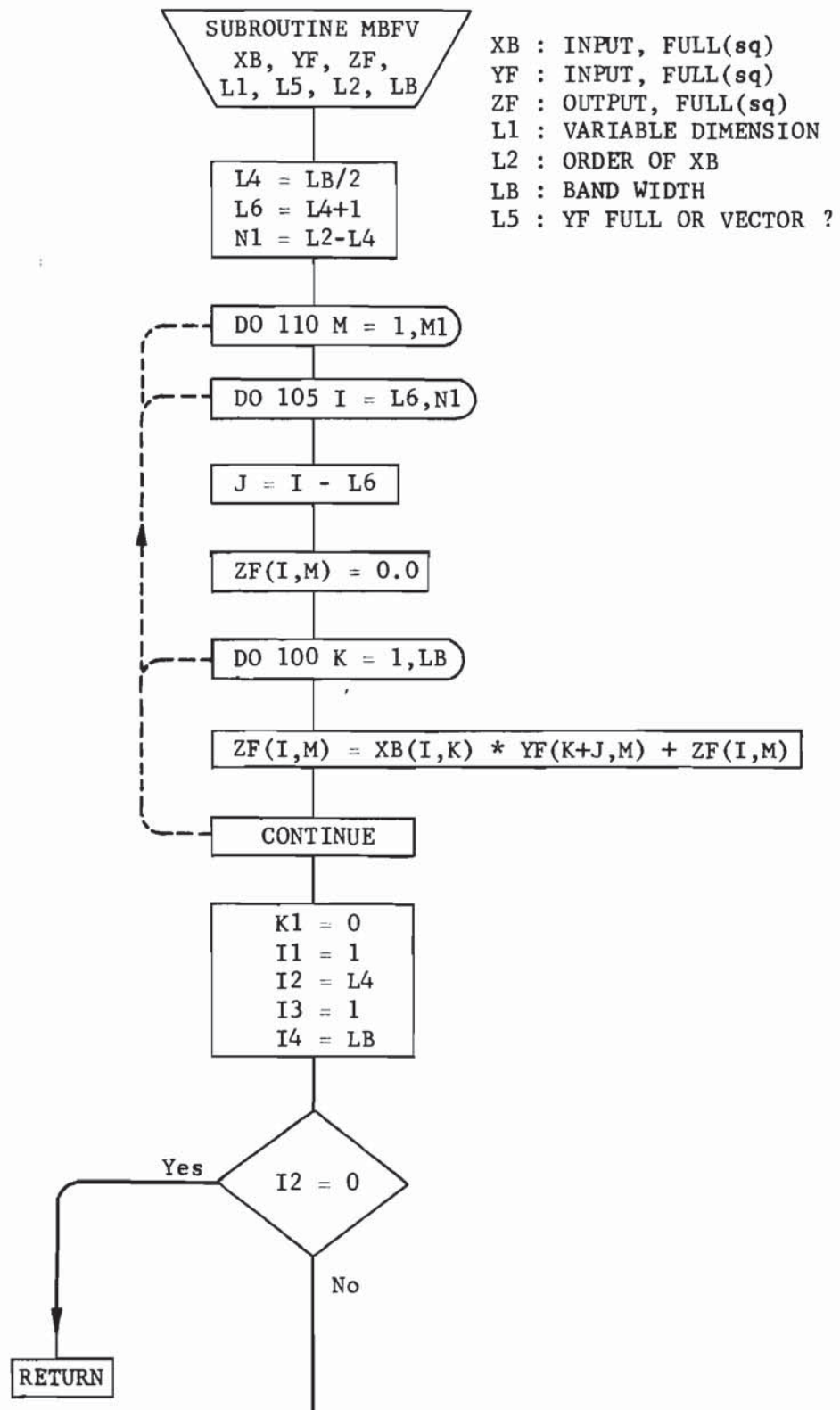


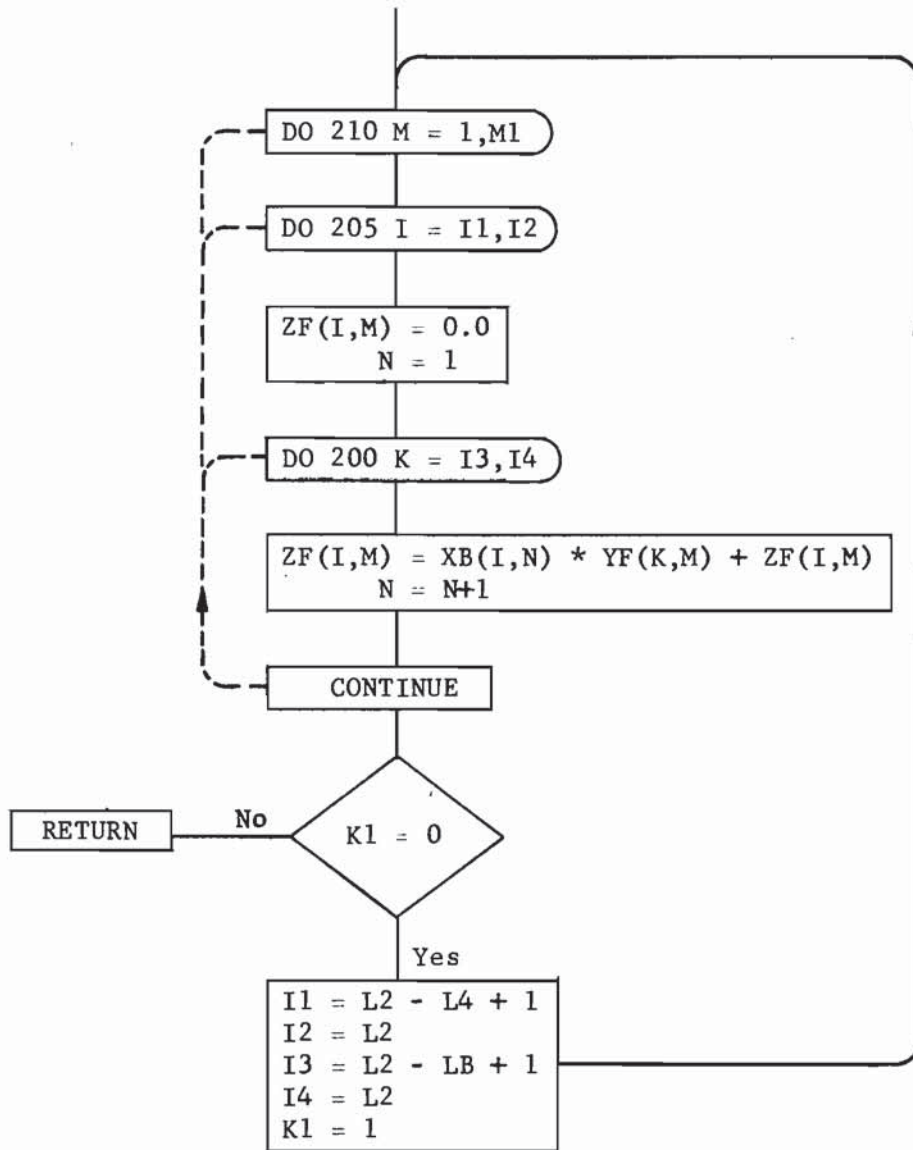
$$\begin{bmatrix} X \end{bmatrix} \cdot \begin{bmatrix} Y \end{bmatrix}^T = \begin{bmatrix} Z \end{bmatrix}$$

```

          SUBROUTINE MBFV ( XB , YF , ZF , L1 , L5 , L2 , LB )
C***** THIS ROUTINE MULTIPLIES A BANDED MATRIX
C          TIMES A FULL MATRIX OR A VECTOR
C          ( XB * YF = ZF )
          DIMENSION XB( L1,LB ) , YF( L1,L5 ) , ZF( L1,L5 )
          DOUBLE PRECISION SUM
C*****B
C          29MY8 E
C***** CAUTION - THE ACCURATE ACCUMULATION OF INNER PRODUCTS IS
C          BEING ATTEMPTED THRU THE DOUBLE PRECISIONING OF THE VARIABLE
C          SUM . CARE SHOULD BE TAKED TO INSURE THIS IS DONE PROPERLY .
C          29MY8 W
C          29MY8 A
C          29MY8 R
C*****E
          M1 = 1
          IF( L1 .EQ. L5 ) M1 = L2
          L4 = LB/2
          L6 = L4 + 1
          N1 = L2 - L4
          DO 110 M = 1,M1
          DO 105 I = L6,N1
            J = I - L6
            SUM = 0.0
          DO 100 K = 1,LB
            SUM = SUM + XB(I,K) * YF(K+J,M) * 1.0D0
          100 CONTINUE
            ZF(I,M) = SUM
          105 CONTINUE
          110 CONTINUE
            K1 = 0
            I1 = 1
            I2 = L4
            I3 = 1
            I4 = LB
          IF( I2 ) 150, 900, 150
          150 DO 210 M = 1,M1
            DO 205 I = I1,I2
              SUM = 0.0
              N = 1
            DO 200 K = I3, I4
              SUM = SUM + XB(I,N) * YF(K,M) * 1.0D0
              N = N + 1
            200 CONTINUE
              ZF(I,M) = SUM
            205 CONTINUE
            210 CONTINUE
              IF( K1 ) 900,300,900
          300 I1 = L2 - L4 + 1
            I2 = L2
            I3 = L2 - LB + 1
            I4 = L2
            K1 = 1
          GO TO 150
          900 RETURN
          END

```



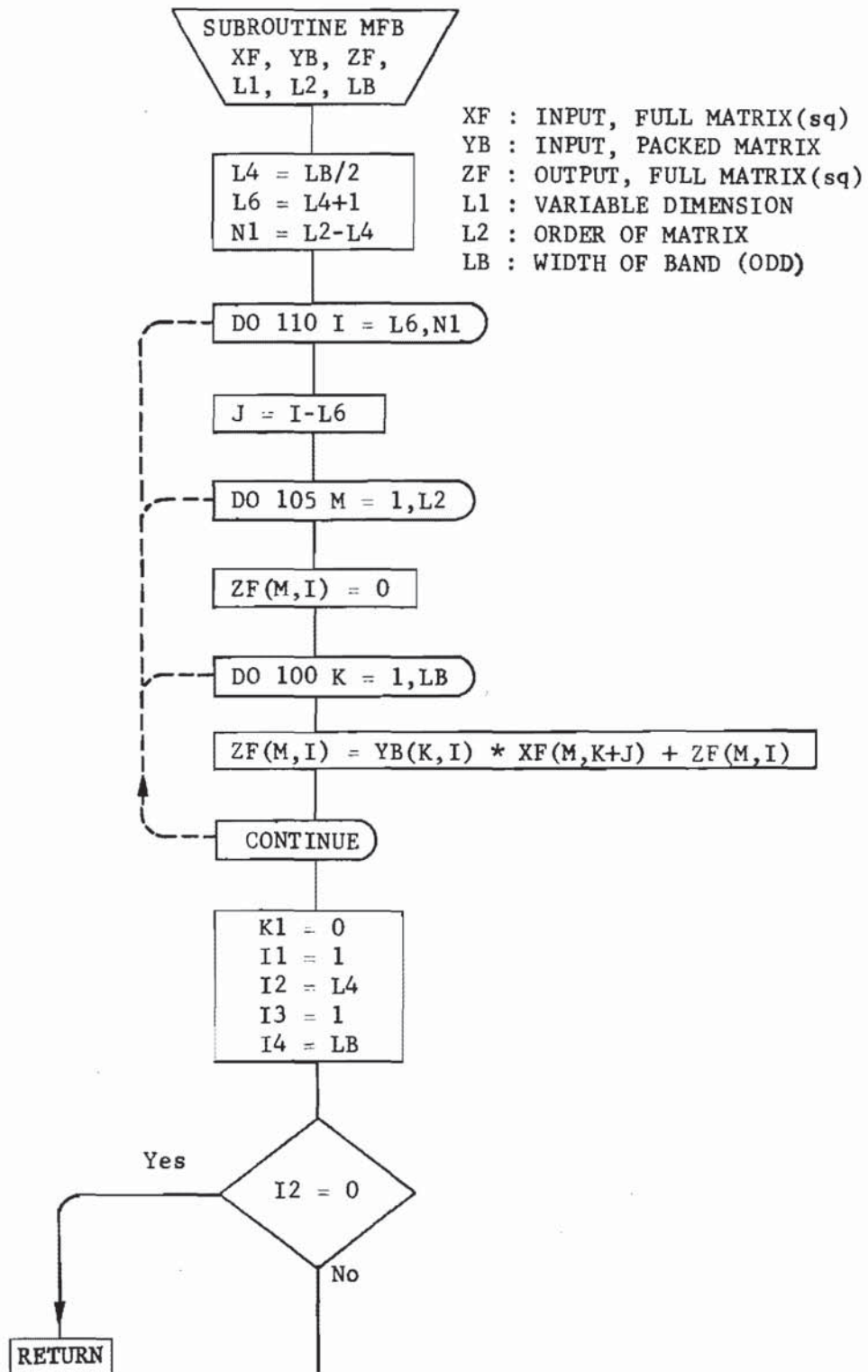


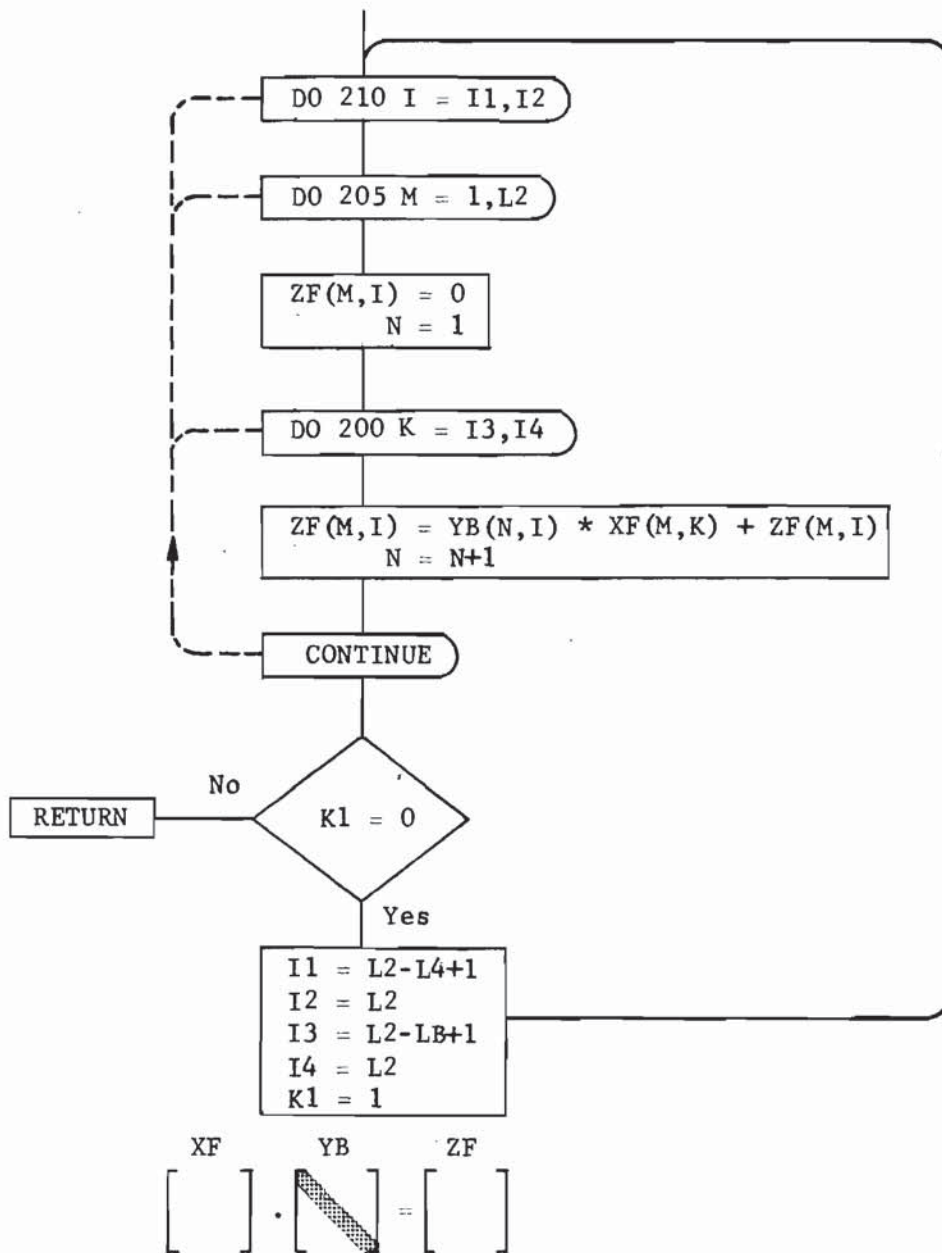
$$\begin{array}{ccc}
 \text{XB} & \text{YF} & \text{ZF} \\
 \left[\begin{array}{c} \diagdown \\ \diagup \end{array} \right] & \cdot \left[\begin{array}{c} \\ \\ \\ \end{array} \right] & = \left[\begin{array}{c} \\ \\ \\ \end{array} \right] \\
 & & L5 = L1 \\
 & \text{OR} & \\
 \left[\begin{array}{c} \diagdown \\ \diagup \end{array} \right] & \cdot \left[\begin{array}{c} \\ \\ \\ \end{array} \right] & = \left[\begin{array}{c} \\ \\ \\ \end{array} \right] \\
 \text{XB} & \text{YF} & \text{ZF} \\
 & & L5 = 1
 \end{array}$$


```

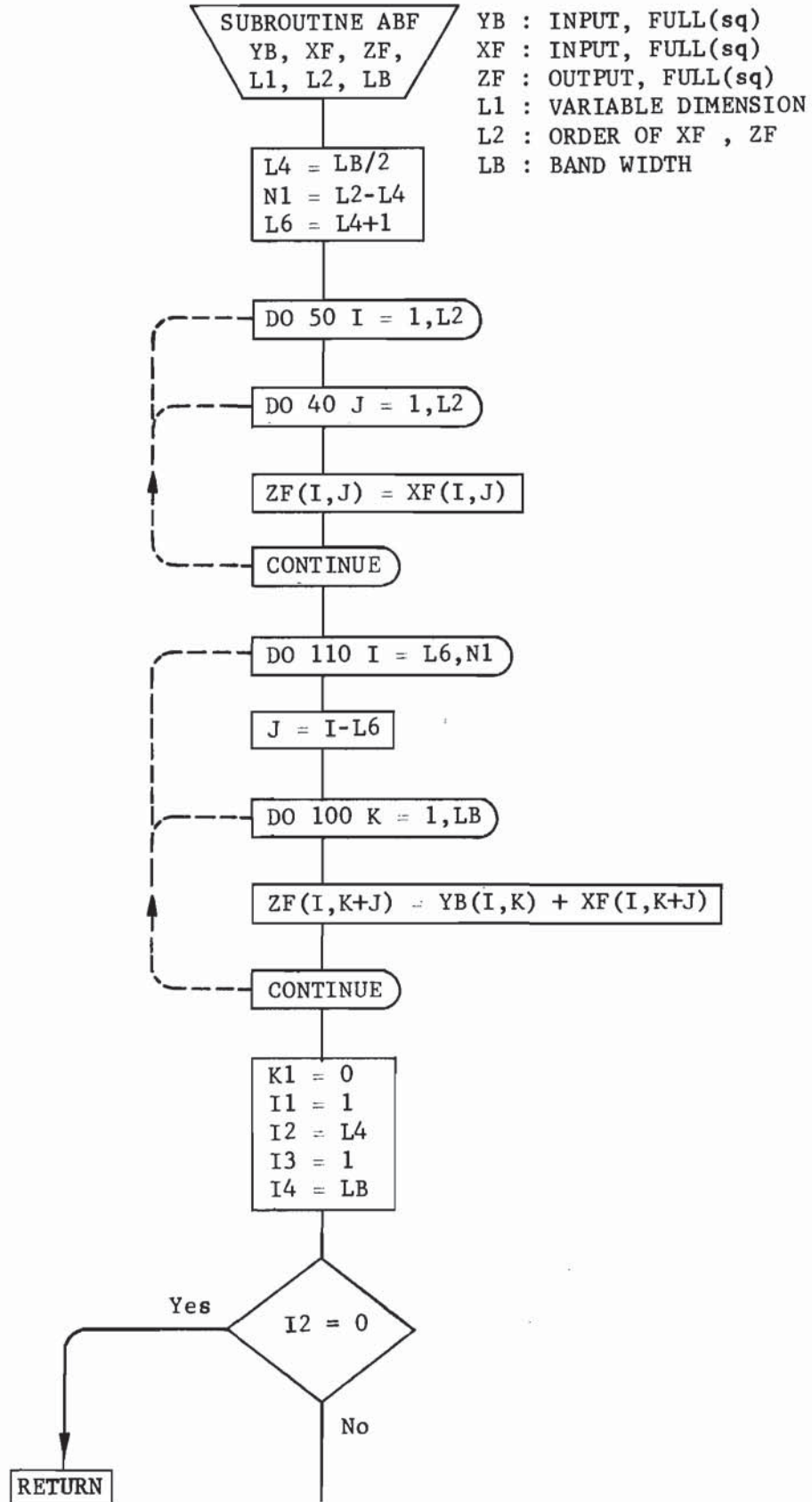
SUBROUTINE MFB ( XF , YB , ZF , L1 , L2 , LB )           07DE7
C***** THIS ROUTINE MULTIPLIES A FULL MATRIX         07DE7
C      TIMES A BANDED MATRIX                          07DE7
C      ( XF * YB = ZF )                               07DE7
C      DIMENSION XF( L1,L1 ) , YB( LB,L1 ) , ZF( L1,L1 ) 07DE7
C      DOUBLE PRECISION SUM                           29MY8
C*****B
C      29MY8 E
C***** CAUTION - THE ACCURATE ACCUMULATION OF INNER PRODUCTS IS 29MY8 W
C      BEING ATTEMPTED THRU THE DOUBLE PRECISIONING OF THE VARIABLE 29MY8 A
C      SUM . CARE SHOULD BE TAKED TO INSURE THIS IS DONE PROPERLY . 29MY8 R
C*****F
      L4 = LB/2                                         07DE7
      L6 = L4 + 1                                       07DE7
      N1 = L2 - L4                                       07DE7
DO 110 I = L6,N1                                       07DE7
      J = I - L6                                         07DE7
DO 105 M = 1,L2                                         07DE7
      SUM = 0.0                                         06MY8
DO 100 K = 1,LB                                         07DE7
      SUM = SUM + YB(K,I) * XF(M,K+J) * 1.0D0         24JE8
100  CONTINUE                                           10N07
      ZF(M,I) = SUM                                     06MY8
105  CONTINUE                                           10N07
110  CONTINUE                                           10N07
      K1 = 0                                           10N07
      I1 = 1                                           10N07
      I2 = L4                                           07DE7
      I3 = 1                                           08DE7
      I4 = LB                                           07DE7
      IF( I2 ) 150, 900, 150                            07DE7
150  DO 210 I = I1, I2                                  10N07
      DO 205 M = 1,L2                                  07DE7
      SUM = 0.0                                         06MY8
      N = 1                                           08DE7
DO 200 K = I3, I4                                       08DE7
      SUM = SUM + YB(N,I) * XF(M,K) * 1.0D0         24JF8
      N = N + 1                                       08DE7
200  CONTINUE                                           10N07
      ZF(M,I) = SUM                                     06MY8
205  CONTINUE                                           10N07
210  CONTINUE                                           10N07
      IF( K1 ) 900,300,900                              10N07
300  I1 = L2 - L4 + 1                                  07DE7
      I2 = L2                                           07DE7
      I3 = L2 - LB + 1                                  07DE7
      I4 = L2                                           07DE7
      K1 = 1                                           10N07
      GO TO 150-                                       10N07
900  RETURN                                           10N07
      END                                             10N07

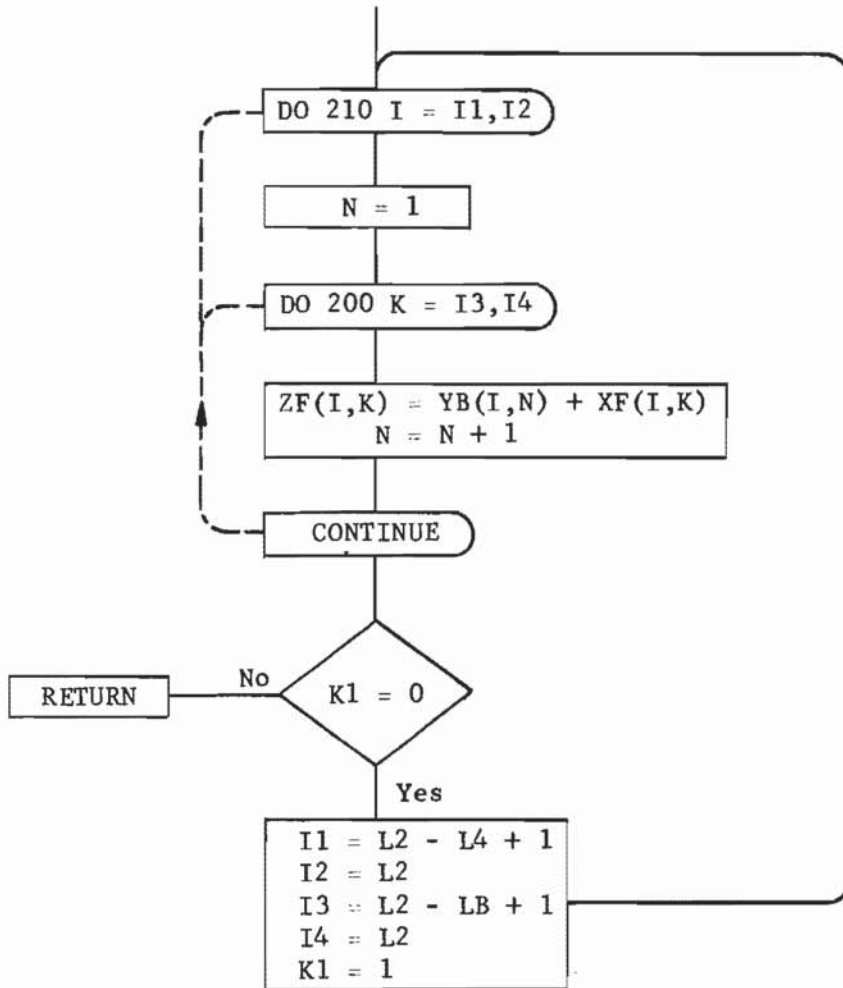
```





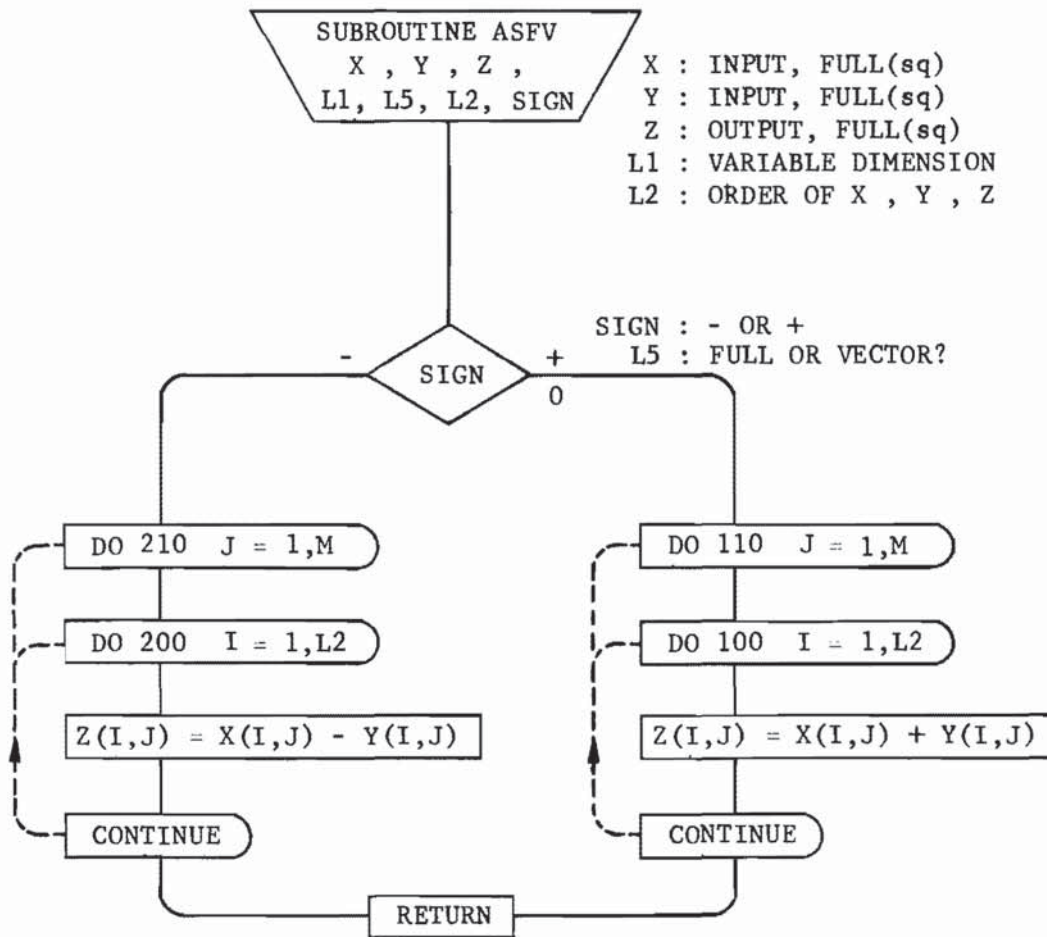
	SUBROUTINE ABF (YB , XF , ZF , L1 , L2 , LB)	07DE7
C*****	THIS ROUTINE ADDS A BANDED MATRIX	07DE7
C	TO A FULL MATRIX	07DE7
C	(YB + XF = ZF OR XF + YB = ZF)	07DE7
	DIMENSION YB(L1, LB) , XF(L1, L1) , ZF(L1, L1)	07DE7
	L4 = LB/2	07DE7
	N1 = L2 - L4	07DE7
	L6 = L4 + 1	07DE7
	DO 50 I = 1, L2	07DE7
	DO 40 J = 1, L2	07DE7
	ZF(I, J) = XF(I, J)	07DE7
40	CONTINUE	07DE7
50	CONTINUE	07DE7
	DO 110 I = L6, N1	07DE7
	J = I - L6	07DE7
	DO 100 K = 1, LB	07DE7
	ZF(I, K+J) = YB(I, K) + XF(I, K+J)	11DE7
100	CONTINUE	10N07
110	CONTINUE	10N07
	K1 = 0	10N07
	I1 = 1	10N07
	I2 = L4	07DE7
	I3 = 1	08DE7
	I4 = LB	07DE7
	IF(I2) 150, 900, 150	07DE7
150	DO 210 I = I1, I2	10N07
	N = 1	08DE7
	DO 200 K = I3, I4	08DF7
	ZF(I, K) = YB(I, N) + XF(I, K)	11DE7
	N = N + 1	08DE7
200	CONTINUE	10N07
210	CONTINUE	10N07
	IF(K1) 900, 300, 900	10N07
300	I1 = L2 - L4 + 1	07DE7
	I2 = L2	07DE7
	I3 = L2 - LB + 1	07DE7
	I4 = L2	07DE7
	K1 = 1	07DE7
	GO TO 150	10N07
900	RETURN	10N07
	END	10N07





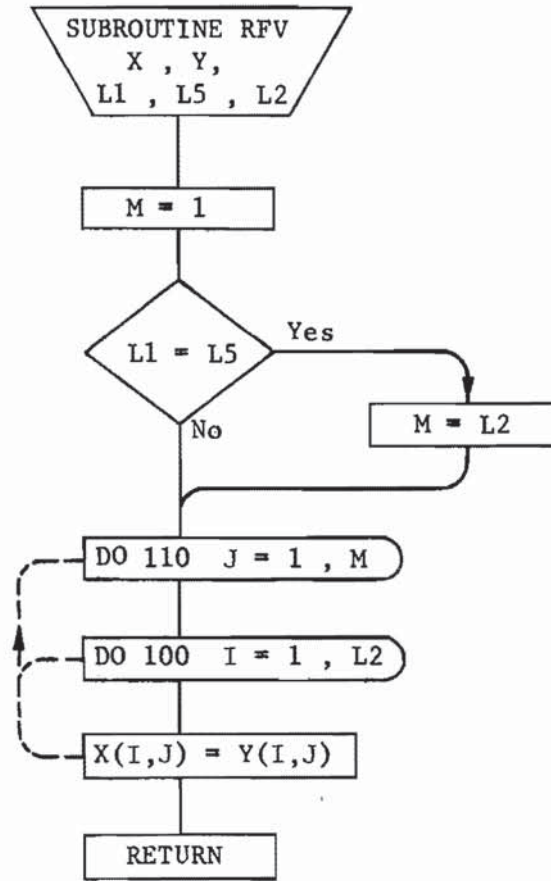
$$\begin{matrix} \text{YB} \\ \text{[} \begin{matrix} \text{N} \\ \text{ } \end{matrix} \text{]} \end{matrix} + \begin{matrix} \text{XF} \\ \text{[} \text{]} \end{matrix} = \begin{matrix} \text{ZF} \\ \text{[} \text{]} \end{matrix}$$

	SUBROUTINE ASFV (X , Y , Z , L1 , L5 , L2 , SIGN)	20MY8
C*****	THIS ROUTINE ADDS OR SUBTRACTS 2 FULL MATRICES OR 2 VECTORS	20MY8
C	(X - Y = Z OR X + Y = Z)	13DE7
	DIMENSION X(L1,L5) , Y(L1,L5) , Z(L1,L5)	13DE7
	M = 1	20MY8
	IF(L1 .EQ. L5) M = L2	20MY8
	IF (SIGN) 190, 50, 50	13DE7
50	DO 110 J = 1,M	13DE7
	DO 100 I = 1,L2	13DE7
	Z(I,J) = X(I,J) + Y(I,J)	13DE7
100	CONTINUE	13DE7
110	CONTINUE	13DE7
	GO TO 300	13DE7
190	DO 210 J = 1,M	13DE7
	DO 200 I = 1,L2	13DE7
	Z(I,J) = X(I,J) - Y(I,J)	13DE7
200	CONTINUE	13DE7
210	CONTINUE	13DE7
300	RETURN	13DE7
	END	13DE7



$$\begin{bmatrix} X \\ - \\ Y \end{bmatrix} + \begin{bmatrix} Y \\ - \\ Z \end{bmatrix} = \begin{bmatrix} Z \end{bmatrix}$$

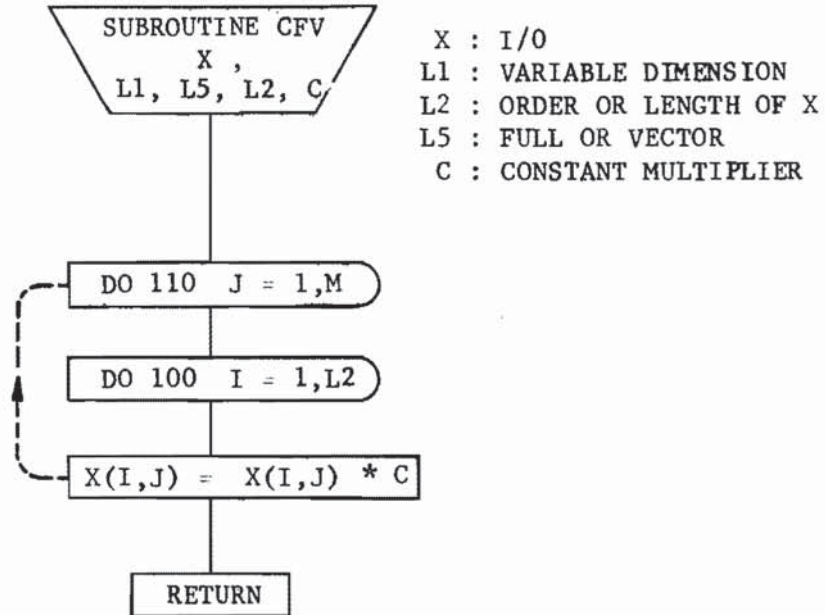

```
      SUBROUTINE RFV ( X , Y , L1 , L5 , L2 )           23MR8
C***** THIS ROUTINE REPLACES A FULL MATRIX OR A VECTOR 23MR8
C      ( X = Y )                                       23MR8
      DIMENSION X(L1,L5) , Y(L1,L5)                 23MR8
              M = 1                                    20MY8
              IF( L1 .EQ. L5 ) M = L2                 20MY8
              DO 110 J = 1,M                           23MR8
              DO 100 I = 1 , L2                         23MR8
                  X(I,J) = Y(I,J)                     23MR8
100          CONTINUE                                   23MR8
110          CONTINUE                                   23MR8
      RETURN                                           20MY8
      END                                               23MR8
```



$$\begin{bmatrix} X \end{bmatrix} = \begin{bmatrix} Y \end{bmatrix} \quad L5 = L1$$

OR

$$\begin{matrix} X \\ \left[\right] \end{matrix} = \begin{matrix} Y \\ \left[\right] \end{matrix} \quad L5 = 1$$



$$\begin{bmatrix} X \end{bmatrix} = C \cdot \begin{bmatrix} X \end{bmatrix}$$

OR

$$\begin{bmatrix} X \\ \end{bmatrix} = C \cdot \begin{bmatrix} X \\ \end{bmatrix}$$

APPENDIX C

USE OF SUBROUTINES FSUB 3 AND FSUB 5

This page replaces an intentionally blank page in the original.

-- CTR Library Digitization Team

APPENDIX C. USE OF SUBROUTINES FSUB 3 AND FSUB 5

Example Using FSUB 32 for TRIP 4 (for problem with symmetric coefficient matrix with a three-wide partitioning)

The call statement requires that FSUB 32 contain the following variables in its parameter list:

SUBROUTINE FSUB 32 (L1, L2, L3, BB, CC, DD, FF, ML, JJ, N2, N3)

where

L1 - variable dimension,

L2, L3 - variable dimension parameters for data, necessary for coefficient matrix generation if they are to be handled as such. If so, they too must be included in the parameter list.

BB, CC, DD, FF - dummy parameters representing the packed submatrices

$$d_{i-1}^T, c_i, d_i, \text{ and } f_i,$$

ML - switch indicating whether problem is parent, regular, or offspring (+, 0, -1). For this routine we are concerned only with whether it is an offspring or not. Remember for the symmetric case

$$b_i = d_{i-1}^T,$$

JJ - index of what partition we are at, JJ = 1, 2, ..., L,

N2 - band width of BB and DD, and

N3 - band width of CC

The dimension statement will be as follows (if generation data are added and variable dimensioning is used, then the additional variables should be added on):

```
DIMENSION BB(L1, N2) , CC(L1, N3) , DD(N1, L1) , FF(L1)
```

Common block RI is needed and appears as follows:

```
COMMON /RI/ NK , NL , NF
```

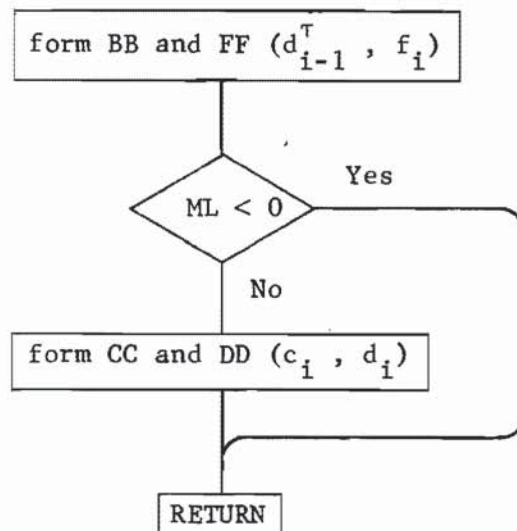
This block is common to both the solution driver (TRIP 4) and the main driving routine utilizing the solution package and therefore will be explained in the following appendix.

In general FSUB 32 will have the following form:

```
SUBROUTINE FSUB 32 (L1, L2, L3, BB, CC, DD, FF, ML, JJ, N2, N3)
```

```
DIMENSION BB(L1, N1) , CC(L1, N2) , DD(N1, L1) , FF(L1)
```

```
COMMON /RI/ NK , NL , NF
```



As it has been mentioned before, the submatrices are packed, and it is here that the packing must be done. Figures C.1 and C.2 explain the packing procedure for the specific routines involved. For FSUB 31 and FSUB 32, BB and CC (b_i, c_i) will be packed according to Fig C.1 and DD (d_i) according to Fig C.2. For FSUB 51 and FSUB 52, AA, BB, CC, and DD (a_i, b_i, c_i, d_i) are done as in Fig C.1 and EE (e_i) as in Fig C.2. In both cases FF (f_i) is a vector. Figures C.3 and C.4 demonstrate a specific application of the packing procedure.

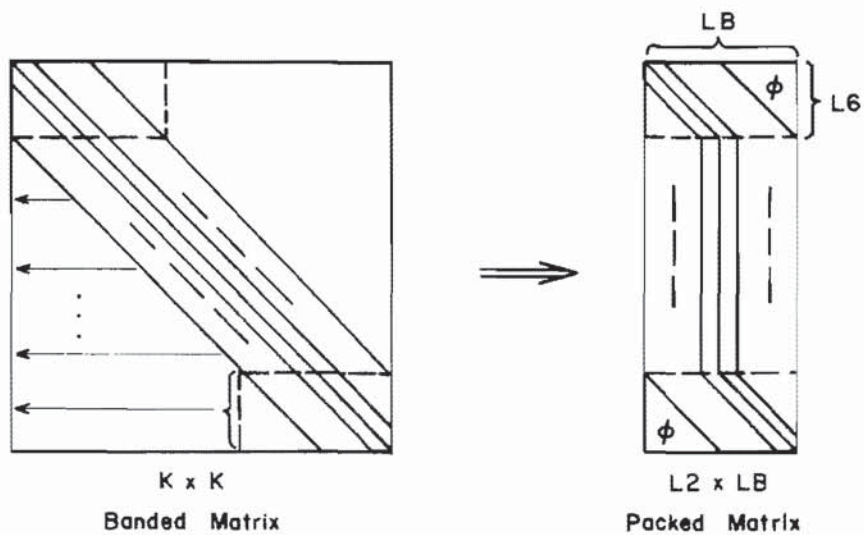


Fig C.1. Packing of the banded matrices that are multiplied by a full matrix.

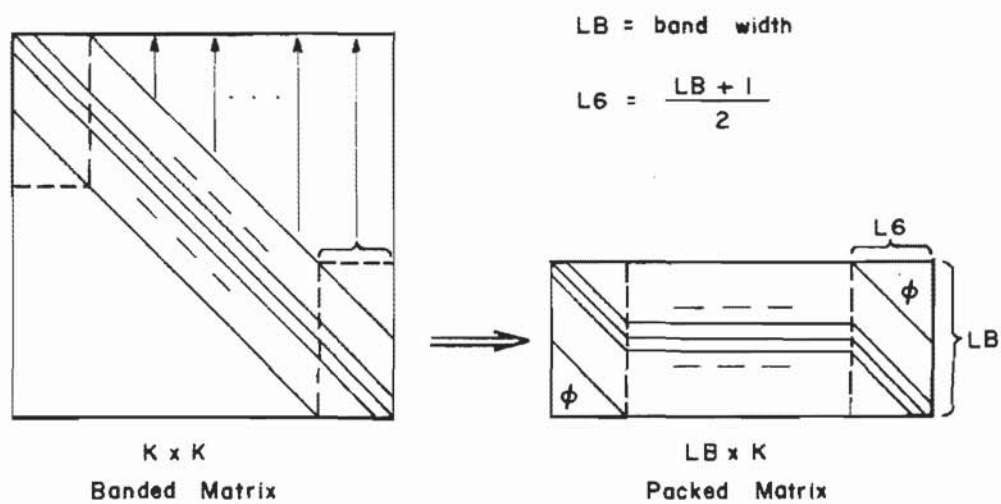


Fig C.2. Packing of the banded matrices that are multiplied by a full matrix.

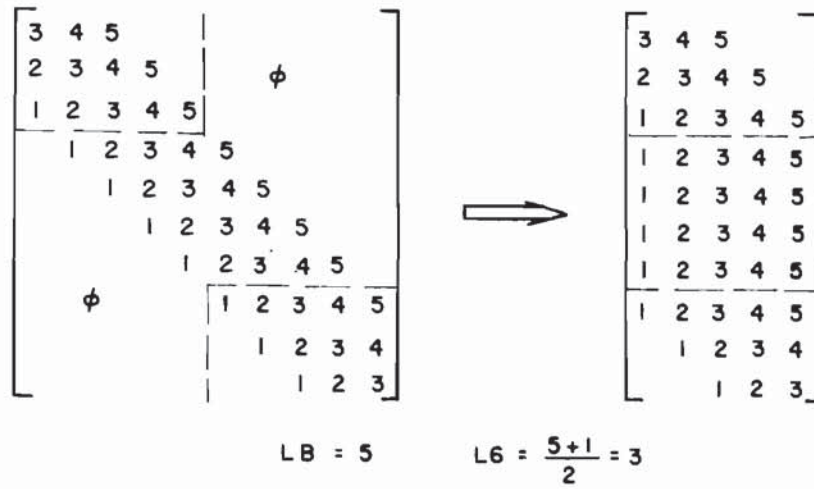


Fig C.3. Example of packing for subroutines ABF and MBFV.

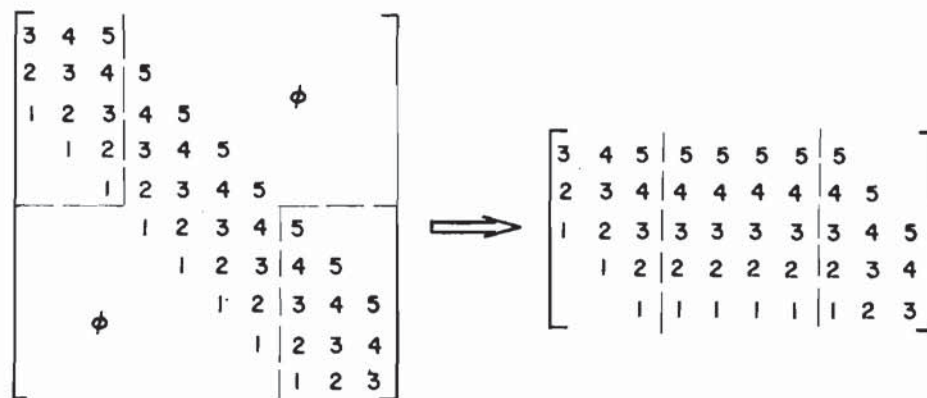


Fig C.4. Example of packing for subroutine MFB.

Example Using FSUB 52 for FRIP 4

The call statement requires that FSUB 52 contain the following variables in its parameter list:

```
SUBROUTINE FSUB 52 (L1, L2, L3, AA, BB, CC, DD, EE, FF, ML, JJ,
1                  N1, N2, N3)
```

where

L1, L2, L3 - as in FSUB 32,

AA, BB, CC, DD, EE, FF - dummy parameters representing the packed matrices e_{i-2}^τ , e_{i-1}^τ , c_i , d_i^τ , e_i , f_i .

For the symmetric case $a_i = e_{i-2}^\tau$, $b_i = e_{i-1}^\tau$, and we need the transpose of d_i .

ML and JJ - as in FSUB 32,

N1 - band width of AA and EE,

N2 - band width of BB and DD, and

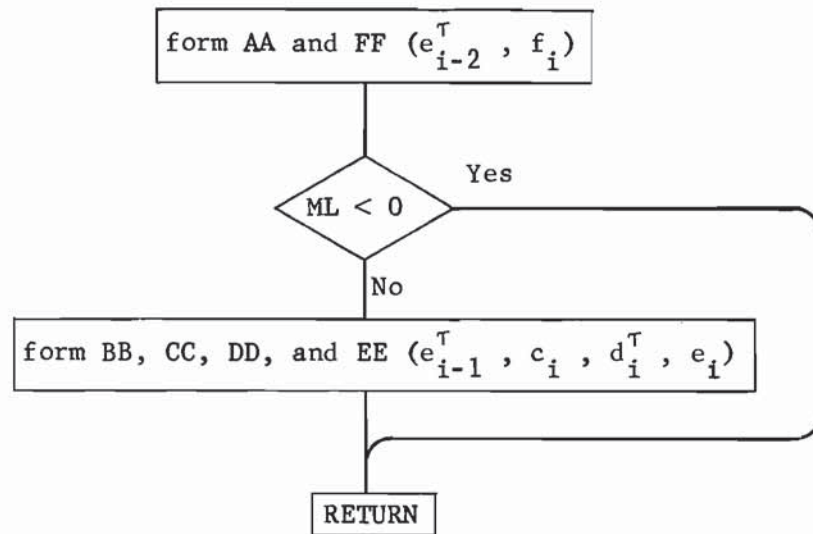
N3 - band width of CC.

In general FSUB 52 will have the following form:

```
SUBROUTINE FSUB 52 (L1, L2, L3, AA, BB, CC, DD, EE, FF, ML, JJ,
1                  N1, N2, N3)

DIMENSION AA(L1, N1) , BB(L1, N2) , CC(L1, N3) , BD(L1, N2) ,
1          EE(N1, L1) , FF(L1)

COMMON /RI/ NK , NL , NF
```



For the nonsymmetric cases either TRIP3 or FRIP3 would be used and therefore FSUB 31 or FSUB 51 would have to be created. The only difference in FSUB 31 as opposed to FSUB 32 would be that b_i is not necessarily equal to d_{i-1}^T ; therefore, it would have to be formed independently and therefore the additional band width parameter $N4$ is needed, and the calling would be as follows:

```
CALL FSUB 31 (L1, L2, L3, BB, CC, DD, FF, ML, JJ, N2, N3, N4)
```

The analogous situation exists for the five-wide nonsymmetric case. The five parameters AA through EE would now stand for a_i through e_i , respectively and again the band width parameters must be added to the calling statement and the subroutine parameter list.

```
CALL FSUB 51 (L1, L2, L3, AA, BB, CC, DD, EE, FF, ML, JJ,
1           N1, N2, N3, N4, N5)
```

This means we would now dimension DD with the variable $N4$ and EE with $N5$ whereas in the symmetric, since it is necessary that $N5 = N1$ and $N4 = N2$, they were not needed.

APPENDIX D

USE OF OVERALL PACKAGE

This page replaces an intentionally blank page in the original.

-- CTR Library Digitization Team

APPENDIX D. USE OF OVERALL PACKAGE

To demonstrate what is necessary to use the solution package, we will take TRIP 4 to use as an example.

First of all, the user's main program must dimension the following variables thusly:

```
DIMENSION A( $\alpha$ ) , AM1( $\alpha$ ) , ATM( $\alpha$ ) , C( $\alpha$ ,  $\alpha$ ) , D( $\alpha$ ,  $\alpha$ ) ,  
1          W( $\beta$ ,  $\rho$ ) , DT1( $\alpha$ , 1) , CC( $\alpha$ , 1) , DD(1,  $\alpha$ ) , FF( $\alpha$ ) (1)
```

For all four routines the dimension statement necessary in the main program is identical with that in the appropriate routine, with the variable dimensions replaced by the actual values the variable dimension parameters have been set equal to. In the example below we have chosen $\alpha = 10$ which limits the maximum value K (the order of our submatrices) to 10. Also we have arbitrarily chosen $N2 = 1$, $N3 = 1$ which is where the one's come from in the above equation. Next we have the following common block:

```
COMMON /RI/ NK , NL , NF (2)
```

Since variable dimensioning has been used throughout, it is necessary that certain variables be defined prior to the call statement. In the event that data information needed for FSUB 32 is in common or is read and programmed in, as is the case in this example, the variables $L2$ and $L3$ are extraneous. They were included solely as variable dimension definers for any necessary data arrays that should be variably dimensioned for practical purposes (in all routines we have used the $L2$ and $L3$ parameters for the W array). We use $L1$ to define α of the above dimension statement.

```
L1 =  $\alpha$   
L2 =  $\beta$   
L3 =  $\rho$  (3)
```

Generally NF will be either 1, 2, or 3 and is merely the starting value for the main do loop that carries us through the L partitions. Due to internal subscripting (biased so as to avoid zero and negative subscripts) it is sometimes advantageous to run from 2 to $L + 1$, or 3 to $L + 2$, instead of the standard 1 to L . NK is the order of the submatrices for the particular problem, and NL is the matrix order of the overall coefficient matrix (there are NL^2 submatrices in the coefficient matrix). NK and NL will generally be changing from problem to problem and therefore will most likely be read in or be a function of some input parameter. For simplicity in our rather restrictive example, we have explicitly given them values but all that is necessary is that they be defined in some fashion.

To distinguish between parent, standard, and offspring problems, the variable ML must be set to either a positive, zero, or negative value, respectively.

```

NF - starting value for main do loop

NK }
NL } - defined as a function of the particular problem (see above)

ML - -, 0, + number depending on type of problem

```

The only remaining necessity in the driving program is the call statement itself.

```

CALL TRIP 4 (L1, L2, L3, ML, A, AM1, ATM, C, CM1, D, DT1, CC,
1          DD, FF, W, 1, 3)

```

All the variables have been explained except the last three. The last two are the respective band widths of the outside and center submatrices, and W is our output parameter containing the solution vector. The NL partitions of W have been stacked as columns of a rectangular array, thereby being more directly related to the type of problems this procedure was written for.

Combining this with the appropriate `FSUB` routine, we have the necessary information to use the solution procedure. On the next page is an example use of `TRIP 4` using a restrictive but sufficient `FSUB 32` to demonstrate the minimal requirements necessary to use the procedure.

	SUBROUTINE FSUB32 (L1,L2,L3,BB,CC,DD,FF,ML,JJ,N1,N2)	27MR8
	DIMENSION BB(L1,N1) , CC(L1,N2) , DD(N1,L1) , FF(L1)	27MR8
	COMMON /RI/ NK , NL , NF	27MR8
	DO 100 I = 1 , NK	27MR8
	BB(I) = 1.0	27MR8
100	CONTINUE	27MR8
	READ 10 , (FF(I),I=1,NK)	27MR8
10	FORMAT (3F5.3)	27MR8
	PRINT 11, JJ , (FF(I),I=1,NK)	20AP8
11	FORMAT (5X,* F(*I1,*) = *,10F6.3)	20AP8
	IF (ML.LT.0) GO TO 1000	27MR8
	DO 200 I = 1 , NK	27MR8
	CC(I) = 10.0	27MR8
200	CONTINUE	27MR8
	IF (JJ.EQ.NL) GO TO 900	27MR8
	DO 300 I = 1 , NK	27MR8
	DD(I) = 1.0	27MR8
300	CONTINUE	27MR8
	GO TO 1000	27MR8
900	DO 500 I = 1 , NK	27MR8
	DD(I) = 0.0	27MR8
500	CONTINUE	27MR8
1000	CONTINUE	27MR8
	RETURN	27MR8
	END	27MR8

PHOR 100 - PARENT -

ML = 1
CONSTANT VECTOR

F(1) = 1.000 1.000 1.000
F(2) = 1.000 1.000 1.000
F(3) = 1.000 1.000 1.000
F(4) = 1.000 1.000 1.000

SOLUTION VECTOR PARTITIONED AND PRINTED
OUT AS A K BY L MATRIX

.0917	.0826	.0826	.0917
.0917	.0826	.0826	.0917
.0917	.0826	.0826	.0917

PHOR 101 - OFFSPRING -

ML = -1
CONSTANT VECTOR

F(1) = 1.000 1.000 1.000
F(2) = 1.000 1.000 1.000
F(3) = 1.000 1.000 1.000
F(4) = 1.000 1.000 1.000

SOLUTION VECTOR PARTITIONED AND PRINTED
OUT AS A K BY L MATRIX

.0917	.0826	.0826	.0917
.0917	.0826	.0826	.0917
.0917	.0826	.0826	.0917

PHOR 102 - OFFSPRING -

ML = -1
CONSTANT VECTOR

F(1) = 1.000 2.000 3.000
F(2) = 4.000 5.000 6.000
F(3) = 7.000 8.000 9.000
F(4) = 10.000 11.000 12.000

SOLUTION VECTOR PARTITIONED AND PRINTED
OUT AS A K BY L MATRIX

.0664	.3362	.5721	.9428
.1581	.4187	.6547	1.0345
.2499	.5013	.7372	1.1263

PHOR 103 - OFFSPRING -

ML = -1
CONSTANT VECTOR

F(1) = -0. -0. -0.
F(2) = -0. -0. 5.000
F(3) = 5.000 -0. -0.
F(4) = -0. -0. -0.

SOLUTION VECTOR PARTITIONED AND PRINTED
OUT AS A K BY L MATRIX

.0052	-.0515	.5103	-.0510
0.	0.	0.	0.
-.0510	.5103	-.0515	.0052

PROB 104 - OFFSPRING -

ML = -1
CONSTANT VECTOR

F(1) = 10.000-0. -0.
F(2) = -0. -0. -0.
F(3) = -0. -0. -0.
F(4) = -0. -0. -0.

SOLUTION VECTOR PARTITIONED AND PRINTED
OUT AS A K BY L MATRIX

1.0107	-.1021	.0103	-.0010
0.	0.	0.	0.
0.	0.	0.	0.

THE AUTHORS

Frank L. Endres is a programmer at the Center for Highway Research. He has concentrated his research mainly in numerical analysis, mathematical programming, and graphics, with an emphasis on solution techniques and their relation to structural analysis.



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

