M-77-65-5 • AUGUST 1965

JUNCTURE STRESS FIELDS IN MULTICELLULAR SHELL STRUCTURES

VOL. I

NUMERICAL METHODS OF SOLVING LARGE MATRICES

by

F. A. BROGAN

Lockheed

MISSILES & SPACE COMPANY

A GROUP DIVISION OF LOCKHEED AIRCRAFT CORPORATION

SUNNYVALE, CALIFORNIA

August 1965

M-77-65-5

Technical Report

JUNCTURE STRESS FIELDS IN MULTICELLULAR SHELL STRUCTURES

Final Report

Nine Volumes

Vol. I Numerical Methods of Solving Large Matrices

by

F. A. Brogan

Contract NAS 8-11480 to National Aeronautics and Space Administration
George C. Marshall Space Flight Center, Huntsville, Alabama

LOCKHEED MISSILES & SPACE COMPANY

FOREWORD

This report is the result of a study on the numerical methods of solving large systems of linear algebraic equations involved in the juncture stress fields of multicellular shell structures. Work on this study was performed by staff members of Lockheed Missiles & Space Company in cooperation with the George C. Marshall Space Flight Center of the National Aeronautics and Space Administration under Contract NAS 8-11480. Contract technical representative was H. Coldwater.

This volume is the first of a nine-volume final report of studies conducted by the department of Solid Mechanics, Aerospace Sciences Laboratory, Lockheed Missiles & Space Company. Project Manager was K. J. Forsberg; E. Y. W. Tsui was Technical Director for the work.

The nine volumes of the final report have the following titles:

Vol. I	Numerical Methods of Solving Large Matrices
Vol. II	Stresses and Deformations of Fixed-Edge Segmental Cylindrical Shells
Vol. III	Stresses and Deformations of Fixed-Edge Segmental Conical Shells
Vol. IV	Stresses and Deformations of Fixed-Edge Segmental Spherical Shells
Vol. V	Influence Coefficients of Segmental Shells
Vol. VI	Analysis of Multicellular Propellant Pressure Vessels by the Stiffness Method
Vol. VII	Buckling Analysis of Segmental Orthotropic Cylinders Under Uniform Stress Distribution
Vol. VIII	Buckling Analysis of Segmental Orthotropic Cylinders Under Non-uniform Stress Distribution
Vol. IX	Summary of Results and Recommendations

SUMMARY

14245

This volume presents two basic numerical techniques for solving, by digital computers, large systems of algebraic simultaneous equations resulting from the finite-difference approximation of the partial differential equations of thin elastic shells. Of the methods available for solving such systems of equations, the matrix factorization and two-line successive over-relaxation methods have been discussed extensively. It is found that the direct methods generally require more computer running time than the iterative methods, especially when the size of the matrix is large. However, the former methods permit rapidly varying mesh spacing which is desirable for the accurate determination of the boundary-layer behavior of fixed-edge shell elements involved in multicellular pressure vessels.

CONTENTS

NOTATION

$a_{i,j}$	elements of A
A	nonsingular matrix
A^{-1}	inverse of A
$A_{i,j}$	submatrices of A
B	column vector
C	complex matrix
D(D_j)	diagonal matrix (submatrices)
E	error vector
F, G	decomposed matrices of A
i, j, k, m, n	dummy indices
I	identity matrix
k	number of non-zero diagonals in A
$\ell_{i,j}$	elements of L
n	number of iterations
L	lower triangular matrix
M	convergent matrix
$r_{k,j}$	elements of R
R, U	upper triangular matrices
X, Y, Z	column vectors

$\rho(A)$	spectral radius of A
ω	relaxation factor
ω_{opt}	optimum relaxation factor
λ	eigenvalue
$\|\| C \|\|$	matrix norm of C

Section 1
INTRODUCTION

Use of finite-difference expressions to approximate partial derivatives appearing in the general shell equations has been described elsewhere.* This method reduces the solution of the partial differential equations to the solution of a linear equation system:

$$AX = B \tag{1.1}$$

When A is nonsingular, a unique solution vector X exists and can, in principle, be computed by one of several well-known methods such as Gaussian elimination. In practice, however, the equation system, Eq. (1.1), becomes so large that the usual methods can no longer be applied without modification.

For example, if a shell segment is covered with a finite difference mesh of 30 rows and 30 columns, the resulting system will contain 2700 equations since there are 3 dependent variables u, v, and w at each station. The full matrix A would then contain 7,290,000 elements. Clearly the storage requirements and execution time involved in manipulating all of these elements is prohibitive, even for present digital computers. Thus, only methods which avoid generating a full matrix with the dimensions of A can be considered.

Before discussing the methods which can be utilized to obtain a solution of Eq. (1.1), it is important to observe the special "banded" form which the matrix A can be given. For example, when the differential equations can be reduced to a set of ordinary differential equations, the natural ordering of equations and unknowns produces a system in which non-zero elements in a given row (equation) occur only on the main diagonal and

*"Investigation of Juncture Stress Fields in Multicellular Shell Structures," by E. Y. W. Tsui, F. A. Brogan, J. M Massard, P. Stern, and C. E. Stuhlman. Technical Report M-03-63-1, Lockheed Missiles & Space Company, Sunnyvale, Calif., Feb 1964 – NASA CR-61050.

in a limited number of columns preceding or following the main diagonal. The number of such non-zero diagonals, k, is then a function of the number of dependent variables and the type of finite-difference expressions selected, but is independent of the number of mesh stations. The situation is fundamentally different with finite-difference representations of partial differential equations. The coupling of the dependent variables in both independent variable directions makes the number of non-zero diagonals dependent on the mesh size. However, since coupling is generally confined to one or two mesh rows in either direction, the system can still be formed so that the number of non-zero diagonals on either side of the main diagonal is small compared to the total number of diagonals in A. The feasibility of both the direct methods and the so-called block iterative methods is dependent on the "banded" form of A. For the direct methods, the number of elements which are involved in arithmetic computations can be reduced to the number of elements appearing in the set of non-zero diagonals of A. The situation with respect to iterative methods is more complex and will be treated in detail in Sec. 3. However, it turns out that all of the useful iterative methods also require a highly banded matrix A for efficient execution. Several particular methods for obtaining a strongly banded matrix A will be described in Secs. 2 and 3.

Section 2
DIRECT METHODS

Most feasible direct methods for the solution of large systems, $AX = B$, are based on some type of factorization. It is shown in standard works on matrix theory that any nonsingular matrix A can be represented after permutation of the rows as the product of a lower triangular and an upper triangular matrix (Ref. 1)

$$A = LR \tag{2.1}$$

When such a factorization has been obtained, the equation, $AX = B$, can be solved easily in two steps. First consider the equation

$$LZ = B \tag{2.2}$$

Since L is nonsingular and lower triangular, this can be solved for the auxiliary vector Z in one "sweep"

$$Z_1 = B_1/L_{1,1} \tag{2.3a}$$

$$Z_2 = (B_2 - L_{2,1}Z_1)/L_{2,2} \tag{2.3b}$$

$$Z_i = \left(B_i - \sum_{j=1}^{i-1} L_{i,j}Z_j\right)\Big/L_{i,i} \tag{2.3c}$$

Similarly, the equation

$$RX = Z \tag{2.4}$$

can be solved in a single "back sweep"

$$X_n = Z_n/R_{n,n} \tag{2.5a}$$

$$X_{n-1} = (Z_{n-1} - R_{n-1,n}X_n)/R_{n-1,n-1} \tag{2.5b}$$

$$X_i = \left(Z_i - \sum_{j=i+1}^{n} R_{i,j}X_j\right)\Big/R_{i,i} \tag{2.5c}$$

Now combining Eqs. (2.2) and (2.4), we have

$$AX = LRX = LZ = B \tag{2.6}$$

From the form of the recursions, Eqs. (2.3) and (2.5), it is evident that no additional storage is needed to obtain the solution vector X beyond that required by the non-zero diagonals in L and R. There are various methods for obtaining a factorization of A and also many distinct representations of A as a product of two matrices. Consequently, there are a number of possible direct methods for solving the system, $AX = B$, among which a few are particularly suited to strongly banded matrices.

2.1 MATRIX FACTORIZATION

Let A be an n by n nonsingular matrix and assume $A = LR$ where L is lower triangular and R is upper triangular. By the definition of matrix multiplication, the elements of A are given by

$$a_{i,j} = \sum_{k=1}^{n} \ell_{i,k}r_{k,j} \tag{2.7}$$

If s = minimum of i and j, then since L is lower triangular and R is upper triangular,

$$a_{i,j} = \sum_{k=1}^{s} \ell_{i,k} r_{k,j} \tag{2.8}$$

Now, the elements of L and R can be computed recursively, using Eq. (2.8)

$$\text{Set } r_{1,1} = 1 \tag{2.9a}$$

then

$$\ell_{1,1} = a_{1,1} \tag{2.9b}$$

The remaining elements in Column 1 and Row 1 can then be obtained if $\ell_{1,1} \neq 0$.

$$\ell_{i,1} = a_{i,1} \tag{2.9c}$$

$$r_{1,j} = a_{1,j}/\ell_{1,1} \tag{2.9d}$$

Choosing $r_{2,2} = 1$ yields

$$\ell_{2,2} = a_{2,2} - \ell_{2,1} r_{1,2} \tag{2.9e}$$

If $\ell_{2,2} \neq 0$, the elements of Column 2 and Row 2 can be found.

$$\ell_{i,2} = a_{i,2} - \ell_{i,1} r_{1,2} \tag{2.9f}$$

$$r_{2,j} = (a_{2,j} - \ell_{2,1} r_{1,j})/\ell_{2,2} \tag{2.9g}$$

Or in general, if $r_{i,i} = 1$

$$\ell_{i,i} = a_{i,i} - \sum_{k=1}^{i-1} \ell_{i,k} r_{k,i}$$

and if $\ell_{i,i} \neq 0$

$$\ell_{j,i} = a_{j,i} - \sum_{k=1}^{i-1} \ell_{j,k} r_{k,i} \qquad i < j$$

$$r_{i,j} = \left(a_{i,j} - \sum_{k=1}^{i-1} \ell_{i,k} r_{k,j} \right) \Big/ \ell_{i,i} \qquad i < j \qquad (2.9h)$$

When the scheme given by the Eqs. (2.9) is carried out, the matrix R is unit upper triangular. The nonsingular character of A insures that the rows of A can always be permuted in such a manner that $\ell_{i,i} \neq 0$, $i = 1, n$.

2.1.1 Factorization of Banded Matrices

The general procedure for factoring a matrix described in the preceding paragraph can be applied in several ways to the solution of banded matrices. As a concrete example to clarify the situation, suppose A is a 10 by 10 matrix with two non-zero diagonals on either side of the main diagonal as shown below:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & 0 & & & & & & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & 0 & & & & & 0 \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & 0 & & & & 0 \\ 0 & & & & & & & & & 0 \\ & & & & & & & & & a_{8,10} \\ & & & & & & & & & a_{9,10} \\ 0 & & & & & & a_{10,8} & a_{10,9} & a_{10,10} & \end{bmatrix}$$

Applying Eqs. (2.9) to find $\ell_{4,1}$ and $r_{1,4}$, gives

$$\ell_{4,1} = a_{4,1} = 0$$

$$r_{1,4} = a_{1,4}/\ell_{1,1} = 0$$

Also

$$\ell_{5,2} = a_{5,2} - \ell_{5,1} \times r_{1,2} = 0 - 0 = 0$$

If no permutation of A is required, it is not difficult to see that L and R each contain only three non-zero diagonals.

$$L = \begin{bmatrix} \ell_{1,1} & 0 & & & 0 \\ \ell_{2,1} & & & & \\ \ell_{3,1} & & & & \\ 0 & & & & 0 \\ & & & & \\ 0 & 0 & \ell_{10,8} & \ell_{10,9} & \ell_{10,10} \end{bmatrix} \qquad (2.10a)$$

$$R = \begin{bmatrix} 1 & r_{1,2} & r_{1,3} & 0 & 0 \\ 0 & 1 & r_{2,3} & r_{2,4} & 0 \\ & & & & 0 \\ & & & & r_{9,8} \\ & & & & r_{9,10} \\ 0 & & & 0 & 1 \end{bmatrix} \qquad (2.10b)$$

Thus to obtain the factorization A = LR, it is only necessary to provide storage for five non-zero diagonals since the elements on the main diagonal of R are all 1.0. Even more significant is the fact that far fewer operations are required to form L and R than would be the case if A were not banded. In general, for any banded

matrix A with k non-zero diagonals on either side of the main diagonal, a factorization A = LR can be obtained where L and R each contain only k non-zero diagonals. For example, iterative methods described in Sec. 3 involve the repeated solution of a linear system AX = Y for different Y vectors, where A contains 27 non-zero diagonals and up to 180 rows. The factorization described here requires the manipulation of only 4,860 elements as compared with 32,400 elements for a matrix inversion. One disadvantage of factorization occurs when some $\ell_{i,i} = 0$. Row i must then be interchanged with some later row to avoid this possibility. However, the matrices L and R may then contain more than k non-zero diagonals. Fortunately, this seldom occurs in practice and can usually be overcome at the expense of only one or two additional diagonals. More serious is the loss of accuracy which may develop if some $\ell_{i,i}$ is very small in comparison with other diagonal elements. The best remedy again is to find the least extensive permutation which will produce a larger term. The existence of such difficulties cannot normally be predicted in advance and may not be easy to diagnose. In case A is positive definite it has been shown that no permutation of A is needed either to prevent some $\ell_{i,i} = 0$, or to avoid numerical inaccuracy (Ref. 1).

In developing the formulas for factorization, Eqs. (2.9), the choice $r_{i,i} = 1$ is arbitrary and primarily a matter of convenience. Other choices are possible and will produce slightly different factorizations. For example, $r_{i,i}$ and $\ell_{i,i}$ can be chosen to be equal in absolute value. If A is positive definite, and $r_{i,i} = \ell_{i,i}$, then R is the transpose of L and the computational effort and storage requirements can be cut in half.

2.1.2 Factorization of Partitioned Matrices

Suppose the matrix A is nonsingular and has the following partitioned form where each submatrix $A_{i,j}$ is n by n, $i = 1,m$; $j = 1,m$.

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & & A_{1,m} \\ A_{2,1} & A_{2,2} & & A_{2,m} \\ & & & \\ A_{m,1} & & & A_{m,m} \end{bmatrix} \qquad (2.11)$$

The multiplication of partitioned matrices satisfies exactly the same rule as the multiplication of ordinary matrices. Furthermore, all of the laws of arithmetic used in deriving the factorization formulas also hold for the addition and multiplication of matrices. Hence, the partitioned matrix A can also be expressed as the product of lower triangular and upper triangular partitioned matrices.

$$A = LR \tag{2.12a}$$

$$R_{i,i} = I \tag{2.12b}$$

$$L_{i,i} = A_{i,i} - \sum_{k=1}^{i-1} L_{i,k} R_{k,i} \tag{2.12c}$$

$$L_{j,i} = A_{j,i} - \sum_{k=1}^{i-1} L_{j,k} R_{k,i} \tag{2.12d}$$

$$R_{i,j} = L_{i,i}^{-1}\left(A_{i,j} - \sum_{k=1}^{i-1} L_{i,k} R_{k,j}\right) \tag{2.12e}$$

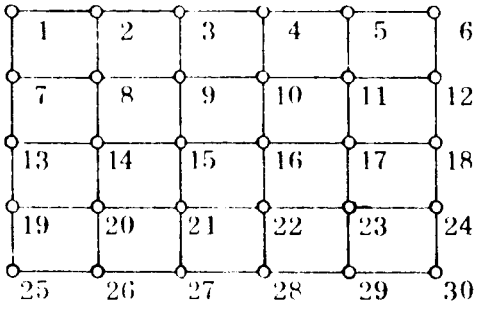
where I is the identity matrix.

The requirement that $\ell_{i,i} \neq 0$ is replaced here by the requirement that $L_{i,i}$ be nonsingular so that an inverse $L_{i,i}^{-1}$ exists. As with the computational scheme [Eqs. (2.9)], Eqs. (2.12) must be carried out in the proper order so that the matrices on the right side of an equation will already have been computed when needed. If the terms appearing in Eqs. (2.3) and (2.5) are understood to denote submatrices instead of elements, the representation of A as the product of partitioned matrices L and R can also be used to solve the equation $AX = B$. Furthermore, if the matrix A contains k non-zero diagonal blocks on either side of the main diagonal of submatrices,

then the matrices L and R will each contain just k non-zero diagonals. Thus block factorization applied to banded matrices achieves computational and storage savings comparable to those obtained by ordinary matrix factorization. Factoring the matrix A in partitioned form also helps to simplify the logical organization of the computational and storage activities required for really large systems. A final advantage derives from the fact that the problem of avoiding small diagonal elements $\ell_{i,i}$ is greatly reduced. Since the inverses of the submatrices $L_{i,i}$ required by Eqs. (2.12), are in general full matrices, there is no loss of efficiency in permitting a full pivotal search while performing the inversion. Consequently block factorization can be expected to exhibit greater numerical stability than ordinary factorization without a pivot search.

2.2 APPLICATIONS OF DIRECT METHODS TO FINITE-DIFFERENCE SYSTEMS

Suppose a rectangular region has been covered with a set of horizontal and vertical lines forming the mesh as indicated in the accompanying figure:



Let

$$AX = B \qquad (2.13)$$

be the linear equation system obtained by approximating the partial differential equations with finite-difference operators at each of the mesh points. The ordering

of points given above is perhaps the most natural one to select and leads to a partitioning of A which is strongly banded. The highest derivatives occurring in the general shell equations are fourth order. The finite-difference expressions used for these derivatives at a given line are coupled to at most the two preceding and two following mesh lines. Thus, writing the finite-difference equations in the same order as the mesh points in the figure, a "block 5-diagonal" partitioning of A is obtained:

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & 0 & & & 0 \\ A_{2,1} & A_{2,2} & & & & & \\ A_{3,1} & & & & & & \\ 0 & & & & & & A_{m-2,m} \\ & & & & & & A_{m-1,m} \\ 0 & & 0 & A_{m,m-2} & A_{m,m-1} & A_{m,m} & \end{bmatrix} \qquad (2.14)$$

With three unknowns at each station, the submatrices $A_{i,j}$ developed for this particular example each have 18 rows and 18 columns. By applying Eqs. (2.12), a block factorization is obtained where each L and R contain just three non-zero block diagonals:

$$A = LR \qquad (2.15)$$

The main diagonal of R consists of identity matrices. A closer inspection of Eq. (2.12d) shows that the leftmost non-zero block diagonal of L is identical with the corresponding diagonal of A. This is a consequence of choosing $R_{i,i} = I$ and holds generally for any block factorization carried out in accordance with Eqs. (2.12). The formulas, Eqs. (2.12) and (2.6), used to solve the equation $AX = B$ can be conveniently arranged in a recursive scheme. This has been described in detail in a previous report.* The computer program developed in accordance with this procedure can handle as many as 4300 equations. In this case each submatrix $A_{i,j}$ is 72 by 72

*"Investigation of Juncture Stress Fields in Multicellular Shell Structures," by E. Y. W. Tsui, F. A. Brogan, J. M. Massard, P. Stern, and C. E. Stuhlman, Technical Report M-03-63-1, Lockheed Missiles & Space Company, Sunnyvale, Calif., Feb 1964 – NASA CR-61050.

and $m = 60$. Then the full matrix A contains 60 block diagonals of which only the central five contain non-zero elements. Execution times on the IBM 7094 for different mesh sizes are presented in Table I (see Sec. 3).

A second possible ordering of the mesh stations is the two-line ordering considered in detail in Sec. 3. This leads to a partitioned matrix A with 3 non-zero block diagonals, referred to in the literature as block tri-diagonal form. The factorization formulas, Eqs. (2.12), then produce matrices L and R with just two non-zero block diagonals. The resulting recursive formulas for solving $AX = B$ are quite simple in form for this special case and have been described by several authors (Ref. 2). However, the diagonal submatrices which must be inverted now contain four times as many elements as the submatrices of the 5-diagonal form, Eq. (2.14). The storage problems presented by the block tri-diagonal form are also very difficult. Consequently the block 5-diagonal factorization has been considered a more efficient method for solving the finite-difference equation systems generated by fourth-order partial differential equations.

Matrix factorization provides an efficient method for obtaining the determinant of banded matrices. Suppose a nonsingular matrix A has been factored by means of Eq. (2.9):

$$A \quad LR \tag{2.16}$$

where L is lower triangular and R is unit upper triangular. From the properties of determinants, it can be shown (Ref. 5) that:

$$\text{Det } (A) \quad \text{Det } (L)\ \text{Det } (R) \tag{2.17}$$

Furthermore, the determinant of a triangular matrix is simply the product of the elements on the main diagonal. The situation with respect to factored partitioned matrices is very similar since Eq. (2.17) is still valid. The determinant of a partitioned triangular matrix is also easily computed by multiplying the determinants of the submatrices on the main diagonal.

Section 3
TWO-LINE ITERATIVE METHODS

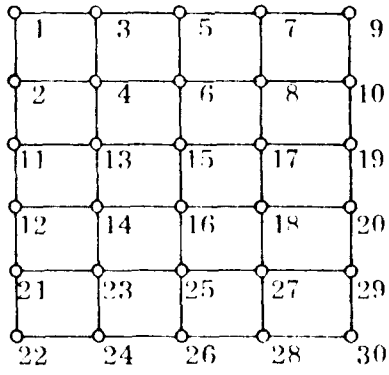
Several cyclic iterative methods have recently been developed for solving linear equation systems generated by finite-difference approximations to boundary value problems. One familiar method closely related to Southwell's "relaxation" technique is the Gauss-Seidel iteration for the system $AX = B$ (Ref. 3):

$$X_i^{(n+1)} = -\frac{1}{a_{i,i}}\left[\sum_{j=1}^{i-1} a_{i,j} X_j^{(n+1)} + \sum_{j=i+1}^{k} a_{i,j} X_j^{(n)} - b_i\right] \qquad 1 \leq i \leq k\,;\ n \geq 0 \qquad (3.1)$$

The use of this method requires that the diagonal elements $a_{i,i} \neq 0$. In fact, reasonably rapid convergence can be expected only if A is strongly diagonally dominant. Since many boundary value problems do not lead to diagonally dominant matrices, it has been necessary to develop more complicated "block" iterative methods. Such methods are obtained by considering partitioned forms of the matrix A and utilizing submatrices or blocks in place of elements in the iteration formulas. When the matrix A is partitioned according to Eq. (2.15), the method is called a one-line block iteration since the submatrices of A correspond to sets of finite-difference equations along one line of mesh points. Unfortunately, it has been found that point and one-line block methods are not efficient for solving the simple but closely related problem of a rectangular plate (Ref. 4). Consequently, several two-line block iteration methods are investigated and computer programs developed for the static analysis of shell segments. The following paragraphs describe the particular methods used and discuss the computational techniques required for efficient program execution.

3.1 GAUSS-SEIDEL BLOCK ITERATION

The methods discussed in this section will be applied to the solution of two-dimensional finite-difference systems. A typical mesh for a shell segment is shown in the accompanying diagram:



A two-line block method requires an ordering of the dependent variables and the difference equations in accordance with the numbering in the diagram. The resulting matrix A may then be partitioned with each submatrix corresponding to two complete lines of the mesh. From the form of the finite-difference expressions such as those described in Sec. 3 of Vols. II, III, and IV, it is readily seen that a derivative on either line of a pair of lines will not require references to stations beyond the preceding or following pair of mesh lines. Thus, the matrix A will be partitioned in "tri-diagonal block" form as follows:

$$A = \begin{bmatrix} D_1 & U_1 & 0 & & 0 \\ L_2 & D_2 & U_2 & & \\ 0 & & & & \\ & & & 0 & \\ & & & U_{m-1} & \\ 0 & & 0 & L_m & D_m \end{bmatrix} \qquad (3.2)$$

The particular mesh of the diagram, for example, leads to a matrix A with m = 3. Each of the three diagonal submatrices D_1, D_2, and D_3 then has 30 rows and 30 columns since there are 3 unknowns and 3 equations at each station.

In order to describe various (two-line) iterative methods and to discuss their convergence properties, it is convenient to first define a decomposition of A.

With A partitioned as in Eq. 3.2, let

$$L = \begin{bmatrix} 0 & & & 0 \\ L_2 & 0 & & \\ 0 & L_3 & & \\ 0 & & L_m & 0 \end{bmatrix} \quad D = \begin{bmatrix} D_1 & & & 0 \\ 0 & D_2 & & \\ & & & 0 \\ 0 & 0 & & D_m \end{bmatrix} \quad U = \begin{bmatrix} 0 & U_1 & & 0 \\ & 0 & U_2 & \\ & & & U_{m-1} \\ 0 & & & 0 \end{bmatrix}$$

Thus $A = L + D + U$.

The form of Gauss-Seidel block iteration is heuristically suggested by rearranging the matrix equation $AX = B$ into the form:

$$(D + L)X = -UX + B$$

The iteration can then be defined by

$$X^{(n+1)} = -(D + L)^{-1}UX^{(n)} + (D + L)^{-1}B \tag{3.3}$$

In view of the block tri-diagonal form of A, this can be arranged as follows:

$$D_i X_i^{(n+1)} = -L_i X_{i-1}^{(n+1)} - U_i X_{i+1}^{(n)} + B_i \tag{3.4}$$

Hence, for each complete iteration, a subsidiary linear system of the form $D_i X_i^{(n+1)} = Z_i$ must be solved m times, presumably by direct methods. The feasibility of block iteration thus depends on the ease with which such systems can be solved and furthermore requires that each diagonal submatrix D_i be nonsingular.

3.2 SUCCESSIVE OVER-RELAXATION

The successive over-relaxation method (SOR) is a natural extension of the Gauss-Seidel iteration.

Let $X^{(n)}$ be the previously computed iteration vector and suppose $Z^{(n+1)}$ is a new vector obtained by Gauss Seidel iteration. The SOR iteration vector $X^{(n+1)}$ is defined by

$$X^{(n+1)} = \omega[Z^{(n+1)} - X^{(n)}] + X^{(n)} \tag{3.5}$$

where ω is called the relaxation factor. Combining Eq. (3.5) with Eq. (3.4) and eliminating $Z^{(n+1)}$ gives

$$D_i X_i^{(n+1)} = \omega\left[-L_i X_{i-1}^{(n+1)} - U_i X_{i+1}^{(n)} + B_i\right] + (1 - \omega) D_i X_i^{(n)} \tag{3.6}$$

Clearly, this reduces to Gauss-Seidel iteration when $\omega = 1$. The iteration is formally defined by

$$X^{(n+1)} = -(D + \omega L)^{-1}[\omega U - (1 - \omega)D]X^{(n)} + (D + \omega L)^{-1}\omega B \tag{3.7}$$

3.3 GENERAL CONVERGENCE PROPERTIES

The iterative methods described here can be considered as special cases of a general iteration scheme based on a decomposition of the matrix A. Suppose $A = F + G$ where F is nonsingular.

Then the equation $AX = B$ implies

$$FX = -GX + B$$

or

$$X = -F^{-1}GX + F^{-1}B \tag{3.8}$$

Defining $M = -F^{-1}G$, Eq. (3.8) leads to an iteration of the following form:

$$X^{(n+1)} = MX^{(n)} + F^{-1}B \tag{3.9}$$

Next define a sequence of error vectors $E^{(n)}$ by

$$E^{(n)} = X - X^{(n)}$$

Noting from Eq. (3.8) that $F^{-1}B = X - MX$, we may write

$$E^{(n+1)} = X - X^{(n+1)} = X - MX^{(n)} - F^{-1}B = X - MX^{(n)} - X + MX = M[X - X^{(n)}]$$

or

$$E^{(n+1)} = ME^{(n)} = M^{(n)}E^{(o)} \tag{3.10}$$

Thus the iteration converges to the solution X if and only if $M^nE^{(o)} \to 0$, $n \to \infty$.

The iteration will converge for an arbitrary starting vector $X^{(o)}$ if and only if $M^n \to 0$ $n \to \infty$. M is then called a convergent matrix.

If C is any complex matrix with eigenvalues $\lambda_1, \ldots \lambda_n$ where $|\lambda_1| \geq |\lambda_2| \ldots \geq |\lambda_n|$, then $\rho(C) = |\lambda_1|$ is called the spectral radius of C (a circle of radius $\rho(C)$ contains

all the eigenvalues of C). Two useful theorems connecting convergence with the spectral radius are (Ref. 2) as follows:

Theorem 1. A complex matrix C is convergent if and only if $\rho(C) < 1$.

Theorem 2. If $||C||$ denotes the matrix norm* of a complex matrix C, then

$$||C^n||^{1/n} \to \rho(C)$$

By Theorem 1, the iteration defined by Eq. (3.9) is convergent for an arbitrary starting vector $X^{(o)}$ if and only if $\rho(M) < 1$.

Theorem 2 provides more precise information concerning the reduction in the error vectors. Using the properties of matrix norms gives

$$||E^{(n)}|| = ||M^n E^{(o)}|| \leq ||M^n|| \, ||E^{(o)}|| \quad , \quad n \geq 0 \qquad (3.11)$$

Thus for sufficiently large n, we have

$$||E^{(n)}|| \sim ||M^n|| \, ||E^{(o)}|| \sim (\rho M)^n ||E^{(o)}||$$

Hence $\rho(M)$ is approximately the factor by which the error is reduced at each iteration.

For certain problems, it can be proved in advance that $\rho(M) < 1$ and therefore that the iteration converges. Unfortunately there are in general no simple methods for obtaining sharp upper bounds for $\rho(M)$. The analyst must therefore be guided primarily by experience in estimating whether iteration can be used effectively. It should be noted, however, that when a program to carry out the iteration, Eq. (3.9), has been

*The matrix norm may be defined as the last upper bound of the set of positive numbers r (Ref. 5) such that for some vector Y, $||Y|| = 1$ and $r = ||AY||$.

prepared, a rather simple modification will permit the calculation of $\rho(M)$. By setting $B = 0$ in Eq. (3.9), the iteration becomes the well-known power method (Ref. 1) for obtaining the dominant eigenvector of M. Of course, this procedure will require a computational effort comparable to carrying out the full iteration, Eq. (3.9).

3.4 CONVERGENCE RATES

Let $A = L + D + U$ be the block tri-diagonal matrix defined in Eq. (3.1). Let

$$M_1 = -(L + D)^{-1}U \tag{3.12}$$

M_1 is the Gauss-Seidel two-line iteration matrix for A.

The two-line SOR iteration is based on the decomposition $A = F + G$ where

$$F = \omega^{-1}D + L$$

$$G = (1 - \omega^{-1})D + U$$

Then the two-line SOR iteration matrix is defined by

$$M_\omega = -(\omega^{-1}D + L)^{-1}[(1 - \omega^{-1})D + U] \tag{3.13}$$

Although the actual values of $\rho(M_1)$ and $\rho(M_\omega)$ cannot be explicitly calculated, the convergence properties can be compared. Henceforth it will be assumed that $\rho(M_1) < 1$. Then, there will exist a number ω_{opt} such that $\rho(M_{\omega_{opt}}) \leq \rho(M_\omega)$ for $0 < \omega < 2$. If all the eigenvalues of M_1 are positive, and A is block tri-diagonal as in Eq. (3.1), it has been shown (Ref. 2) that the optimum relaxation factor is given by

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho(M_1)}} \tag{3.14}$$

Furthermore, in this case

$$\rho\left(M_{\omega_{opt}}\right) = \omega_{opt} - 1 \tag{3.15}$$

If $\rho(M_1)$ is close to 1.0 as frequently happens for large problems, the SOR iteration is much more efficient. For example, if $\rho(M_1) = 0.99$ then $\omega_{opt} = 1.818$ and $\rho\left(M_{\omega_{opt}}\right) = 0.818$

$$0.99^{100} = 0.3664$$

$$0.818^{100} = 0.188 \times 10^{-8}$$

Hence 100 iterations of the Gauss-Seidel type would only reduce the starting error by a factor of 0.3664, while the SOR would evidently have converged in much less than 100 iterations. One drawback to the SOR iteration is the fact that the value of $\rho(M_1)$ is needed in order to compute ω_{opt}. If no estimates are available, $\rho(M_1)$ can be obtained from the SOR iteration by setting $\omega = 1$ and $B = 0$.

Values of $\rho(M_1)$ for a cylindrical shell segment with fixed boundaries are presented in Table 1.

3.5 COMPUTATIONAL TECHNIQUES

As indicated in Secs. 3.2 and 3.3, the two-line iteration methods under consideration require the solution of m subsidiary linear systems, $D_i X^{(n+1)} = Z_i$, for each iteration. If a mesh with 20 rows and 20 columns is selected, m is 10 and each submatrix D_i has 120 rows and 120 columns. Obviously, to obtain these solutions by the usual methods of Gaussian elimination would be impractical both from the standpoint of computer storage requirements and execution time. Fortunately, the special form of the submatrices D_i permits an efficient direct solution of these linear systems.

A detailed examination of the finite difference system derived from the governing partial differential equations, together with the ordering of discrete variables described in Sec. 3.1, shows that all the non-zero elements of D_i are contained in 27 consecutive diagonals. These 27 diagonals have 3240 elements as compared with 14,400 elements in the full submatrix. Since the D_i do not change throughout the iteration, they can be factored once and the solution of $D_i X^{(n+1)} = Z_i$ can then be easily obtained from the factored form of D_i. See Sec. 2 for a discussion of this method. It is important to notice that for the problems under consideration, the D_i are not all distinct. This occurs because the differential equations for cylindrical, conical, or spherical shell segments have coefficients which are functions of at most one of the independent variables. Thus only three distinct submatrices D_i are required to reflect the interior and boundary blocks. Similarly, each of the off-diagonal submatrices L_i and U_i contains only 25 non-zero diagonals and most of these submatrices are identical.

3.6 PROGRAM OPTIMIZATION

A Fortran language computer program based on the methods previously outlined was developed to carry out the two-line SOR iteration. Thus for a 20 by 20 mesh, each iteration required the solution of subsidiary systems $D_i X_i = Z_i$, $i = 1, 10$. The primary effort in forming each Z_i consisted in multiplying a matrix with 120 rows and 77 non-zero diagonals by a vector. The factored form of D_i involved 120 rows and 27 diagonals. The convergence rates obtained were quite satisfactory. However, the program execution time was similar to that for the direct method as described in Sec. 2. A close study of the computer program revealed that a major portion of the time for each iteration was consumed in the multiplication of the submatrices L_i, D_i, and U_i by the vector $X_i^{(n)}$. Although only the minimum possible number of non-zero diagonals were actually used, these diagonals still contained many more zero elements than non-zero elements. Hence program efficiency required the elimination of as many of these "useless" multiplications as possible. This was accomplished by setting up an auxiliary address array for each distinct equation type and each submatrix. This

array indicated the number of each unknown appearing in an equation relative to that equation number. Thus only the coefficients of these unknowns were stored, eliminating most of the zero elements. The multiplication of the matrix by the vector $X^{(n)}$ could then be conveniently carried out by utilizing simultaneously the indirect addressing and address modification features available on the IBM 7094. The storage requirement of 9240 cells for 120 rows and 77 diagonals was reduced to 2120 cells. Although the programming work involved was rather substantial, the effort was repaid with a four-fold improvement in execution time. In addition, the size of the finite-difference mesh which could be treated without undue use of auxiliary tape storage was substantially increased. Sample execution times for the direct method and the revised iterative method are presented in Table 1.

Table 1

SOLUTION OF A CYLINDRICAL SHELL SEGMENT

Number of Equations	Spectral Radius	Number of Iterations	Time by Iteration (min)	Time by Direct (min)
190	0.885	14	0.35	0.75
300	0.940	20	0.65	1.1
600	0.975	27	0.9	3.0
1200	0.9896	37	2.2	12.0
1728	0.9935	46	4.5	24.0

Section 4
DISCUSSION

Both direct and two-line successive over-relaxation iteration methods have been used to solve the boundary value problems of segmental shells. Table 1 shows the comparison of results obtained by these two methods.

From Table 1 it is apparent that as far as the computer (IBM 7094) running time is concerned the iterative method is more favorable than the direct method. Several qualifications should be noted, however. First, the iterative method has only been developed for uniform mesh spacing while the direct method permits rapid changes in spacing near the boundary which are desirable for the accurate determination of the boundary-layer bending characteristics of shell segments. Unfortunately, as pointed out by Varga, rapid changes in mesh spacings have a disastrous effect on the spectral radius of the iteration matrix (Ref. 2). Second, since the direct method obtains a factorization of the matrix A, it is possible to solve the system $AX = B$ for many different B vectors at very little additional cost. This fact is utilized for the rapid determination of influence coefficients required in the study of juncture stress fields. Besides the methods already mentioned, a semi-iterative method using Chebyshev polynomials has also been investigated in the present work, and results compared with those obtained by SOR iteration. The SOR method converged in $\sim 20\%$ fewer iterations than the Chebyshev semi-iterative method. Such convergence indicates that the relaxation factor used in SOR iteration was close to optimum (Ref. 2).

Another class of iterative methods, generally referred to as "alternating direction block methods," has been developed in recent years to solve the boundary value problems for elliptic partial differential equations (Ref. 2). These methods entail a rather complex iteration procedure which is justified by a more rapid convergence rate for certain specific cases. While the two-line SOR iteration is quite successful for

moderate mesh sizes, the number of iterations required for a rectangular mesh with m rows and m columns is roughly proportional to m. The number of iterations used by a typical alternating direction method with four iteration parameters, for example, is proportional to $m^{1/4}$ (Ref. 6). Because of the greater computational effort required for one complete iteration of the alternating direction method, the two-line SOR method has been found more efficient for small meshes. A cross-over point generally occurs at meshes of moderate size. The alternating direction methods are then significantly faster for large meshes. However, it should be noted that the results described here are for second-order elliptic partial differential equations. No work providing a general theoretical background for equations general enough to include the shell equations has been reported. The central idea of the alternating direction methods is to decompose the finite-difference equations A into two sets H and V. This is done in such a way that only H has coupling in the horizontal direction and only V has coupling in the vertical direction. Then one complete iteration requires two stages, one in which a linear system $HZ_1 = Y_1$ is solved by direct methods and a second in which another system $VZ_2 = Y_2$ is solved. Thus one is alternately solving in horizontal and vertical directions. Intuitively, one expects that some similar technique could be quite efficient for the general shell equations. Efforts to develop such methods may be necessary to handle the large mesh sizes which are often desirable and with which the computers of the immediate future will be able to cope.

Section 5
REFERENCES

1. A. S. Householder, The Theory of Matrices in Numerical Analysis, Ginn & Co., 1964

2. R. S. Varga, Matrix Iterative Analysis, Prentice Hall, 1962

3. F. B. Hildebrandt, Introduction to Numerical Analysis, McGraw-Hall Book Co., 1956

4. D. S. Griffin, "A Numerical Solution for Plate Bending Problems," Bettis Atomic Power Laboratory Report WAPD-230, Feb 1963

5. P. R. Halmos, Finite Dimensional Vector Spaces, 2nd ed., D. Van Nostrand, 1958

6. G. Birkhoff, R.S. Varga, and D. Young, "Alternating Direction Implicit Method," Advances in Computers, Vol. 3, Academic Press, London, 1962