DR. KAM W. WONG[*]
University of Illinois
Urbana, Illinois 61801
GERALD M. ELPHINGSTONE
DMA Aerospace Center
St. Louis, Missouri 63118

Recursive Partitioning by Direct Random Access

The combination of random access and the recursive algorithm yields the best approach for the solution of normal equations in block aerotriangulation.

INTRODUCTION

A HIGHLY efficient computer subprogram, code named FILE, was successfully developed for solving large systems of normal equations in simultaneous block aerotriangulation. It was designed to minimize computer core-storage requirement and to maximize computational speed. A recursive partitioning algorithm was used to take full advantage of the positive definite, banded, symmetric properties of the coefficient matrix of the normal equations. In addition, a direct random-access technique was used for data storage and retrieval.

ABSTRACT: *A highly efficient computer subprogram in Fortran IV language was developed for solving large systems of normal equations. It incorporated the recursive partitioning algorithm with the direct access* I/O *technique which provided the capability of directly accessing individual data records. The subprogram was tested in a computer program called* SAPGO *for the simultaneous solution of photogrammetric block. For a block of 4 × 45 photos, the total* CPU *time for one complete iteration on the* IBM *system 360/75 was found to be 5 minutes and 8 seconds. The* CPU *time for the solution of the 1080 normal equations alone was only 1 minute and 53 seconds.*

The recursive partitioning algorithm is a simple variation of the fundamental Gauss elimination method of solving simultaneous equations. It recognizes the fact that the zero elements located outside of a diagonal band in a band matrix remain zero throughout the elimination process, and thus an algorithm can be developed to operate strictly within the non-zero band. The computational advantage of such an algorithm was reported by Stock (1956) and Rutishauser (1958). Tezcan (de Jong and Tezcan, 1965) successfully developed a computer subprogram for such an algorithm in which five tape units were used for auxiliary storage. Snowden (1966) investigated the relative efficiency between a partitioning algorithm and the direct bordering technique for general symmetric normal equations. Brown (1968) and Elassal (1969) have both reported the use of a partition algorithm in photogrammetric block aerotriangulation.

[*] Presented at the Annual Convention of the American Society of Photogrammetry in Washington, D.C., March 1972, under the title "An Efficient Computer Technique for Recursive Partitioning."

A listing of the subprogram developed by Snowden (1966) has been published. Unfortunately, this subprogram also uses five tape units as auxiliary storage device, causing inefficiency in the storing and retrieving of data.

In the subprogram FILE to be described in this paper, the capability of the third-generation computers for direct, random access to data files in disk storage is utilized further to increase the computational speed and efficiency. This technique eliminates the necessity of backspacing records and therefore decreases the total time required for building and solving a set of equations.

Subprogram FILE was developed as an equation solver for an advanced computer aerotriangulation program called SAPGO (Wong and Elphingstone; 1971). This program has the unique capability of incorporating independent geodetic measurements such as horizontal angles and distances in a simultaneous solution of photogrammetric blocks. The computing efficiency of FILE was tested in a SAPGO program using fictitious data and the test results will be reported in this paper.

Recursive Partitioning

Consider a system of n normal equations represented in matrix notation as

$$\underset{(n,n)}{\mathbf{S}}\ \underset{(n,1)}{\mathbf{X}} = \underset{(n,1)}{\mathbf{C}} \tag{1}$$

where $\mathbf{S}$ is a symmetric band matrix with a bandwidth p, $\mathbf{X}$ is a matrix of the unknown parameters, and $\boldsymbol{C}$ is a matrix of constants.

This system of equations may be partitioned as shown in Figure 1. The partition is such that S_{11} is a square matrix with a dimension of $q \times q$. The dimension q must be chosen so that $(n - p)$ is divisable by q. Furthermore, S_{22} must have a dimension of $p \times p$ and S_{13} must be a null matrix.

This partition breaks up Equation 1 into the following three equations:

$$\mathbf{S}_{11}\mathbf{X}_1 + \mathbf{S}_{12}\mathbf{X}_2 + \mathbf{0}\mathbf{X}_3 = \boldsymbol{C}_1 \tag{2}$$

$$\mathbf{S}_{12}{}^T\mathbf{X}_1 + \mathbf{S}_{22}\mathbf{X}_2 + \mathbf{S}_{23}\mathbf{X}_3 = \boldsymbol{C}_2 \tag{3}$$

$$\mathbf{0}\mathbf{X}_1 + \mathbf{S}_{23}{}^T\mathbf{X}_2 + \mathbf{S}_{33}\mathbf{X}_3 = \boldsymbol{C}_3. \tag{4}$$

Solving Equation 2 for $\mathbf{X}_1$ yields

$$\mathbf{X}_1 = \mathbf{S}_{11}{}^{-1}(\boldsymbol{C}_1 - \mathbf{S}_{12}\mathbf{X}_2). \tag{5}$$

Substituting this expression into Equations 3 and 4 yields

$$\overline{\mathbf{S}}_{22}\mathbf{X}_2 + \mathbf{S}_{23}\mathbf{X}_3 = \overline{\boldsymbol{C}}_2 \tag{6}$$

$$\mathbf{S}_{23}{}^T\mathbf{X}_3 + \mathbf{S}_{33}\mathbf{X}_3 = \boldsymbol{C}_3 \tag{7}$$

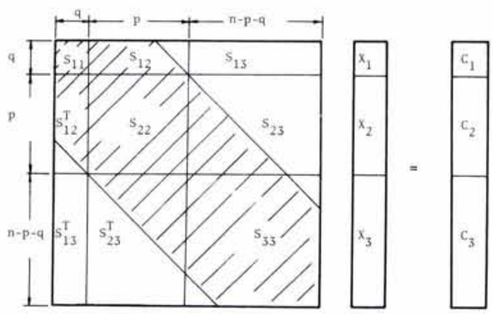

FIG. 1. Partition of normal equations.

where
$$\bar{S}_{22} = (S_{22} - S_{12}^T S_{11}^{-1} S_{12}) \tag{8}$$
$$\bar{C}_2 = (C_2 - S_{12}^T S_{11}^{-1} C_1). \tag{9}$$

Equations 6 and 7 now consist of $(n-q)$ equations and $(n-q)$ unknowns, and can be conveniently represented as

$$\underset{(n-q,n-q)}{S^1} \; \underset{(n-q,1)}{X^1} = \underset{(n-q,1)}{C^1} \tag{10}$$

in which the superscript denotes the number of times the original set of equations in Equation 1 has been partitioned, or the number of sets of q equations that have been eliminated.

At stage 2, Equation 10 is partitioned and reduced to $(n-2q)$ equations in exactly the same manner. This partitioning process is repeated successively until only p equations remain; i.e.

$$\underset{(p,p)}{S^m} \; \underset{(p,1)}{X^m} = \underset{(p,1)}{C^m}. \tag{11}$$

This is accomplished after m stages where $m = (n-p)/q$.

Stages 1 to m in the above partitioning process is identically equivalent to the foreward solution of the standard elimination procedure. The backward solution is initiated by solving Equation 11; i.e.,

$$X^m = (S^m)^{-1} C^m.$$

Equation 5 can now be used successively in backward substitution to solve for one set of q unknowns at each stage. In the first backward stage, the solution can be represented as follows:

$$\underset{(q,1)}{X_1^{m-1}} = (S_{11}^{m-1})^{-1} (C_{11}^{m-1} - S_{12}^{m-1} X^m)$$

It will also require m stages to complete the entire solution.

During the backward solution, instead of solving directly for the unknowns, the banded portion of the inverse matrix S^{-1} can be computed. Let the inverse matrix of the reduced matrix S^i at the i-th stage be denoted as follows

$$(S^i)^{-1} = \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{12}^T & D_{22} & D_{23} \\ D_{13}^T & D_{23}^T & D_{33} \end{bmatrix}. \tag{12}$$

The submatrices D_{12} and D_{11} can be computed by the recursive equations

$$D_{12} = -\bar{S}_{11}^{-1} S_{12} D_{22}^{-1} \tag{13}$$
$$D_{11} = S_{11}^{-1}(I - S_{12} D_{12}^T). \tag{14}$$

These expressions are used recursively during each backward stage, adding q rows to the inverse during each stage.

This algorithm can be easily extended to compute the entire inverse matrix S^{-1}. However, as the inverse is used primarily for error analysis, interest is centered primarily on the elements bordering the diagonal. By not computing the elements outside of the diagonal band, large savings in both storage space and computer time can be achieved without sacrificing any computational accuracy.

Direct File Access

Due to the limited capacity of the core memory of present day computers, the solution of large systems of equations require the use of auxiliary storage devices such as tapes, disks or drums. The major programming problems therefore are: (1) minimizing the time required to transfer the data to and from the auxiliary storage device and (2) keeping record of the exact location of each data set.

Third-generation computers offer random access routines that partition disk storage into directly addressable records. Each record of disk storage may be accessed in a random fashion in the Fortran program. The IBM 360/75 computer at the University of Illinois at Urbana-Champaign uses a Fortran compiler which allows random access to disk storage in the following manner:

(1) *An executive control card to reserve a block of disk space.* For example, the following statement reserves 500 records containing 2904 bytes each and the third and fourth digits in the sequence number FT01F001 indicates that this data set is numbered "01"—
//GO.FT01F001 DD UNIT=DISK,SPACE=(2904,(500,1)),DISP=(NEW,PASS).

(2) *A Fortran statement to set up the disk storage in Fortran compilation.* For example, the following statement set up the storage space reserved in the above statement—
DEFINE FILE 1(500,726,U,N1).

The number 726 indicates the number of words in each record (1 word = 4 bytes); the variable U indicates that read or write from this data set is always without format control; and $N1$ is an index variable.

(3) *Data are written into disk storage by a* WRITE *statement.* The following statement writes the first q equations into record No. 12 of disk No. 1—
WRITE(1′12)((S(I,J),J=1,P),C(I),I=1,Q).

(4) *Data are read from disk storage and put into core by a* READ *statement.* The following statement reads the above set of equations from disk storage—
READ(1′12)((S(I,J),J=1,P),C(I),I=1,Q).

Data Management Algorithm

The data management algorithm for FILE was based on the following characteristics of the recursive partitioning method:

(1) The coefficient matrix **S** of the normal equations is symmetric, and hence only the upper or lower triangular portion needs to be stored or computed;

(2) At any stage of the recursive process only matrices $\mathbf{S}_{11}$, $\mathbf{S}_{12}$, $\mathbf{C}_1$ and $\mathbf{C}_2$ need to be in core storage; and

(3) The basic partition of the matrix **S** is the same at each stage.

Consider in particular a system of normal equations represented as

$$\underset{(n,n)}{\mathbf{A}}\ \underset{(n,1)}{\mathbf{X}} = \underset{(n,1)}{\mathbf{U}}\,.$$

Before subprogram FILE can be called to solve this set of equations, the latter must be stored into disk in a specified manner. Figure 2 shows how the matrices A and U must be partitioned, and Figure 3 shows how the submatrices in the upper

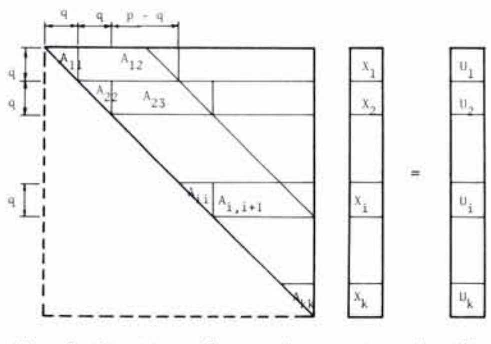

FIG. 2. Partition of normal equations for file storage.

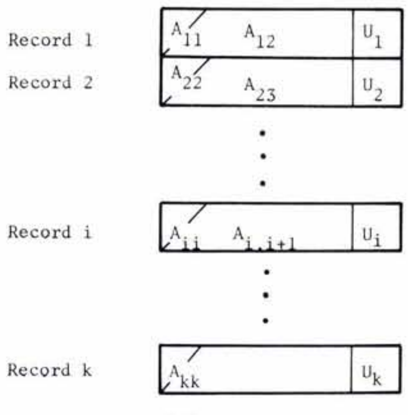

FIG. 3. Contents of data-set records at beginning of solution.

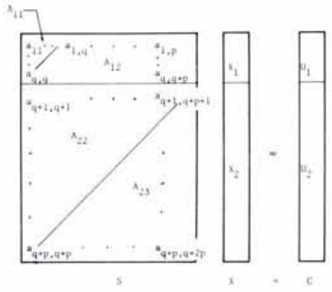
FIG. 4. Collapsed submatrices in core memory.

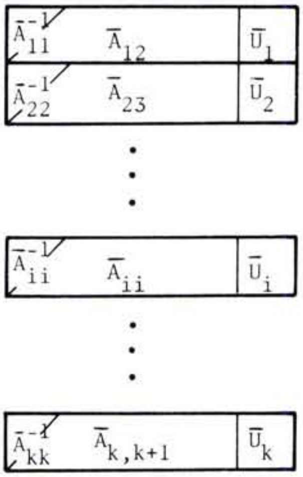

FIG. 5. Contents of data-set records after the forward solution.

triangular portion must be stored into their respective records in the data file. In this manner, the address of any set of q equations can be easily determined.

When subprogram FILE is called, the first $(p+q)$ equations, which are stored in the first $[1+(p/q)]$ records, are read into the computer core memory. These equations are stored in core as the working S and C matrices in the collapsed form shown in Figure 4. During the first stage of partitioning, the matrices S_{11}^{-1}, $\bar{S}_{22}$ and $\bar{C}_2$ are computed according to Equations 8 and 9. The submatrix S_{11}^{-1}, which is actually A_{11}^{-1}, is then put back to disk storage in record No. 1 together with submatrices S_{12} (which is A_{12}) and U_1 (which is C_1). In turn, a new set of q equations are brought in from record No. $(p/q+2)$. This procedure is repeated for $(n-p)/q$ times to complete the foreward solution. Figure 5 illustrates the contents of the disk records at the end of the foreward solution.

In the backward solution, if a direct solution is being computed, the solution matrix X is partitioned according to Figure 2 and the submatrices are stored in the locations of the corresponding U_i matrices. If the inverse D of the coefficient matrix A is being computed, the inverse matrix D is partitioned as shown in Figure 6(a) and the submatrices lying within the diagonal band are computed and stored in the disk records as shown in Figure 6(b).

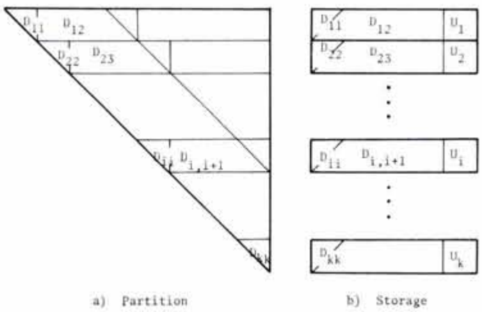

FIG. 6. Partition and storage of inverse matrix.

Operation Count

The computing time required to solve a system of normal equations depends largely on the number of division and multiplication operations and on the number of input/output requests to the auxiliary storage device. The number of divisions and multiplications required by subprogram FILE to solve a system of n normal equations having a bandwidth p is estimated to be as follows:

(1) Foreward solution—

$$N_f = m(p^2q + p(2q^2 + q) + q^3)$$

where

$$m = \tfrac{1}{6}(n - p).$$

(2) Solution of the p equations by Cholesky's square root method during the last stage of the foreward solution—

$$N_c = \tfrac{1}{6}p^3 + \tfrac{3}{2}p^2 + \tfrac{1}{3}p + p \text{ square roots.}$$

(3) Backward substitution for a direct solution:

$$N_b = m(q^2 + pq).$$

Thus, the total number N of operations is estimated by the following expression:

$$N = m(6p^2 + 48p - 252) + \tfrac{1}{6}p^3 + \tfrac{3}{2}p^2 + \tfrac{1}{3}p + p \text{ square roots}$$

For $q = 6$,

$$N = 40n + (8n - 40)p + (n - 6)p^2 - \tfrac{5}{6}p^3 + p \text{ square roots.}$$

Correspondingly, to solve the same system of normal equations, Cholesky's square root method requires

$$(\tfrac{1}{6}n^3 + \tfrac{3}{2}n^2 + \tfrac{1}{3}n + n \text{ square roots})$$

operations and a Gauss elimination will require

$$(\tfrac{1}{3}n^3 + \tfrac{1}{2}n^2 + \tfrac{5}{6}n)$$

operations.

If p approaches n, the number of operations in the recursive partitioning method becomes greater than both the Cholesky and Gauss methods. Therefore, this method and subprogram FILE is not recommended for small block where p approaches n.

Another subprogram based on the bordering method of solving normal equations

Table 1. Recorded Computing Time for Recursive Algorithm

No. of Equations	648	648	648	1080
Block Dimension	4×27	6×18	9×12	4×45
Bandwidth	60	84	120	60
No. of Operations (1×10^6)	0.10	0.20	0.47	0.13
Total CPU Time for one Complete Iteration of Photogrammetric Solution	3′ 1″	4′ 8″	5′ 41″	5′ 8″
CPU Time for Solution of Normal Equations	1′ 8″	*2′ 0″	*4′ 0″	1′ 53″

* By estimation

TABLE 2. ESTIMATED CPU TIME FOR LARGE SYSTEMS OF EQUATIONS

No. of Equations	1080	1080	1728	1728	1728
Block Dimension	6×10	9×20	4×72	6×48	9×32
Bandwidth	84	120	60	84	120
No. of Operations (1×10^6)	0.25	0.52	0.17	0.30	0.60
Total CPU Time for one Complete Iteration of Photogrammetric Solution	8′ 0″	12′ 0″	9′ 0″	12′ 0″	17′ 0″
CPU Time for Solution of Normal Equations	4′ 0″	8′ 0″	3′ 0″	6′ 0″	9′ 0″

was developed for use in small blocks. This subprogram has been incorporated into a special version of SAPGO which requires no auxiliary storage devices and which can solve a block of up to 40 photos in a computer core capacity of 300 K.

TEST RESULTS

Four test cases were conducted on an IBM System 360/75 computer to determine the CPU time required to form and solve large systems of equations using random access programming techniques and the recursive algorithm. The tests were conducted with subprogram FILE operating inside a SAPGO program. The CPU time for the recursive algorithm alone was recorded for only two cases. However, the total CPU time required for one complete iteration of a photogrammetric block solution was recorded for all four cases. The latter total CPU time included the sorting of the measured data (image coordinates as well as control data), loading data records, forming the normal equations, solving the normal equations, computing the corrections to the pass points, updating all the unknowns and printing the final results.

The test cases and their results are tabulated in Table 1. The recorded CPU time has a standard deviation of 10 per cent.

Table 2 shows the estimated CPU time for cases involving larger bandwidths and larger blocks of photos. Table 3 shows for comparison the number of mathematical operations required by the Cholesky and Gauss methods for solving the photo blocks included in Tables 1 and 2. The number of operations does not vary with the bandwidth for these two methods. It is obvious from Table 3 that both of these methods are far inferior to the recursive partitioning methods for large systems of equations.

TABLE 3. OPERATION COUNTS FOR CHOLESKY AND GAUSS METHODS

	No. of Operations (in millions)	
No. of Equations	*Cholesky*	*Gauss*
648	46	91
1080	21	42
1728	864	1721

CONCLUSIONS

The direct random-access technique for the storage and retrieval of data are ideally suited for the recursive partitioning algorithm. The combination of random access and the recursive algorithm yields the best approach for the solution of normal equations in photogrammetric block aerotriangulation.

Direct random-access storage is being used in other parts of the SAPGO program besides subprogram FILE. The efficiency of the complete SAPGO program is evident from the total CPU time recorded in Table 1.

A version of Subprogram FILE has also been developed for use on a UNIVAC 1108. The random-access-to-disk storage is handled differently in this computer, but only minor modifications were needed in the subprogram.

ACKNOWLEDGEMENT

The work reported in this paper was part of a research project supported by the U.S. Army Research Office-Durham and conducted at the University of Illinois. Mr. Elphingstone worked on this project as a research assistant when he was a graduate student at the University.

REFERENCES

Brown, D., "A Unified Lunar Control Network," *Photogrammetric Engineering*, 34:12, December 1968, pp. 1272–1292.
de Jong, S. H., and Tezcan, S. S., "The Electronic Computer in Survey Adjustments at the University of British Columbia," *The Canadian Surveyor*, 19:1, March 1965, pp. 87–89.
Elassal, A. A., "Algorithm for the General Analytical Solution," *Photogrammetric Engineering*, 35:12, December 1969, pp. 1268–1277.
Rutishauser, H., "Solution of Eigenvalue Problems with the LR-Transformation," National Bureau of Standards, *Applied Mathematics Series*, No. 49, January 15, 1958, pp. 47–81.
Snowden, J. M., "An Investigation of Practical Solution of Large Systems of Normal Equations," M. Sc. Thesis at the Ohio State University, Columbus, Ohio, 1966.
Stock, J. R., Die mathematischen Grundlagen für die Organisation der elektronischen Rechenmaschine der Eidgenössische Technische Hochschule (Verlag Birkhäuser, Basel, 1956).
Wong, K. W., and Elphingstone, G., "Aerotriangulation by SAPGO," *Photogrammetric Engineering*, 38:8, Aug. 1972, pp. 779–790.

NICK G. YACOUMELOS
University of Illinois
Urbana, Illinois 61801

Comments on "Stereoscopy—A More General Theory"

MR. GEORGE L. LAPRADE'S "More General Theory of Stereoscopy" has two distinct characteristics. Firstly, it tries to reconcile two basic schools of thought, namely the *projectionists* with the *fixationists*. Secondly, it makes an interesting effort to replace the conventional variables of stereoscopy with a set of abstract ones, namely the angles α, ξ and γ. But, at the end it comes to express the *vertical exaggeration* as a function of the base-to-height ratio and to approximate it by the value

$$q = 5(B/H).$$

To that extend the *more general theory of stereoscopy* comes to support Jackson's approximate expression[1], who proposed the relation $q = 4(B/H)$ in 1959. Furthermore, it presents one of the very rare experimental efforts in this field and, unquestionably, the most interesting one.

Still the fact remains that, to continue the search for a mathematical expression that will give a quantitative measure of what is called (although it does not exist) *vertical exaggeration* is a Quixotic effort. Because

(*Continued on page 282*)