Structural Stiffness Matrix Wavefront Resequencing Program (WAVEFRONT)

R. Levy
DSIF Engineering Section

The computer program WAVEFRONT is a preprocessor that resequences the stiffness matrix for wavefront reduction prior to processing by structural analysis computer programs. The program operation, deck, input data requirements and suggested usage are described here. Summary data extracted from example applications show that this program is an effective preprocessor for the NASTRAN structural analysis program and is generally preferable to alternative bandwidth reduction preprocessors.

I. Introduction

Computational efficiency and reduction of core storage requirements for finite-element structural analysis computer programs depends upon the ordering of the structural stiffness matrix. Effective ordering makes it possible to arrange for storage and to perform computations only within a compact region that is densely populated with nonzero coefficients and to omit computations and storage provisions for an empty region with zero coefficients. Depending upon the coding and storage method used during decomposition of the stiffness matrix for problem solution, compactness typically depends either on the matrix bandwidth (i.e., the ELAS program), or on the matrix wavefront (i.e., the SAMIS program). It is also possible to employ a mixed coding method[1] (i.e., the NASTRAN program) that can operate on a combination of bandwidth and wavefront.

In preparing the analytical model for processing by one of the finite-element analysis computer programs it is possible to order the node labels by inspection to produce a relatively compact stiffness matrix. Nevertheless, for complex models with many nodes, development of an effective ordering sequence could be laborious and also require a skilled analyst. Consequently, it is often practical to label the nodes in some convenient manner that will facilitate and simplify preparation of all the input data without regard to sequencing effectiveness, and then to produce an improved sequencing by an automated procedure that reorders the nodes.

WAVEFRONT is a preprocessor computer program that reorders the nodes of the structural stiffness matrix prior to entering a structural analysis computer program. It was originally developed to reduce the matrix wavefront for follow-on processing by the SAMIS computer

[1]Bandwidth is the distance of an extreme matrix element from the diagonal. Wavefront at a particular row of the decomposition is the number of columns that affect the arithmetic in lower rows. With the mixed coding method, the term "active columns" is the wavefront minus the number of columns within the bandwidth.

program. However, the wavefront reduction capability has subsequently been found to be effective for NASTRAN program processing and the input and output data are now NASTRAN-oriented.

II. Program Approach

The resequencing objective is to minimize or reduce the number of degrees of freedom (DOF) that are contained in the stiffness matrix wavefront. This has the effect of diminishing both computational time and storage requirements. A specific measure of the objective would depend upon the particular coding of the stiffness matrix decomposition, and measures for computation time efficiency are not necessarily the same as measures for minimum storage. A possible measure for computational time is either the root mean square or the sums of squares of the wavefront for all rows of the stiffness matrix. The measure for storage could be either the maximum wavefront at any row if fixed-wavefront storage is used, or the average wavefront of all the rows if variable wavefront storage is used. For bandwidth-oriented decomposition algorithms, there are equivalent candidate measures defined in terms of bandwidth, rather than wavefront. As a simplification, the measure adopted here is the maximum wavefront at any of the rows. It is proposed on the basis of both intuition and experiment that a superior reordering in terms of the adopted measure tends also to be superior in terms of the other foregoing alternative measures. Furthermore, it is assumed that all nodes of the analytical model tend to have about the same numbers of associated degrees of freedom. Therefore, as a simplification, the approach is designed to minimize or reduce the maximum wavefront of the nodal connectivity matrix, instead of to reduce the wavefront as expressed in degrees of freedom of the stiffness matrix.

The resequencing algorithm is a 'minimum growth' method. That is, assuming that at a particular execution phase the procedure has progressed to identify the first i nodes of the resequenced nodal connectivity matrix, then the new $(i + 1)$-th node is selected as the node that will cause the smallest increase of the wavefront existing for the i-th node. To start the procedure, the first node selected can be either defined by the user or, by default, can be determined by the program as a node with the minimum number of connections. Further details of the algorithm in addition to a hand-executed example are given in Ref. 1.

It is evident that the effectiveness of the final sequencing obtained by this algorithm depends upon the starting point. Consequently, provisions are made within the program to perform user-specified numbers of complete resequencing cycles with starting points after the first cycle chosen at random by the program. During this cyclic repetition, the program will abort any cycle currently in progress as soon as it determines that the maximum wavefront is as large as the terminal maximum wavefront of a previously completed cycle. Consequently, unsuccessful cycles that terminate early require less computation time than cycles that go to completion and achieve an improvement. Each time a resequencing cycle produces a reduction of the previous maximum wavefront, a set of NASTRAN-type sequence cards are punched by the program for insertion in the NASTRAN data deck. When there are several such sequence card sets produced, the last set is the best.

III. Data Input

The input data deck consists of a title card, an option card, a set of cards that supplies the node labels and initial sequencing, and a set of cards that supplies the finite element connections. Specific requirements are as follows:

(1) *Title card:* Format (80A1).
Contains any alphameric title desired by the user for problem identification and echo.

(2) *Option card:* Format (6I10).

Fields 1 and 2: blank.

Field 3: ROWA = integer to supply the original sequence number for the node that is to be first in the first new resequence cycle. If blank or zero and for sequence cycles greater than 1, the program will pick the first row.

Field 4: KCYCLE = the number of cycles of resequencing to be performed. If blank or zero, one cycle will be performed.

Field 5: JPRINT, if not zero, then the initial connectivity matrix terms will be printed.

Field 6: NEWCNT, if not zero, then at the end of a cycle that achieves an improved wavefront, a table of the wavefront at each row of the new connectivity matrix will be printed plus a summary of the maximum, average, and rms wavefront for this sequencing.

(3) *Node label card set.*

(a) First card: a dummy card with any text signifies the beginning of this set. The card will be echoed.

(b) Node cards: Format (A6, 2X, A8).

First field: not blank.

Second field: node label in the form of a string of characters with no intervening blanks. The label can be placed anywhere within the field. NASTRAN "GRID" cards may be used for this set.

Note: initial nodal sequencing is established by the program from the order in which these cards are submitted. The sequence and node list is echoed.

(c) Last card: Format (A16).

Blank to signify termination of this card set.

(4) *Connection card set.*

(a) First card: a dummy card as in item 3 above.

(b) Element connection cards: Format (A3, SX, NA8), where S and N depend upon the type of connection as tabulated below:

Connection type	First field	S	N
2-node	CBA CTU	21	2
2-node	CON	13	2
3-node	CTR	21	3
4-node	CQD CQU CSH CTW	21	4

The last N fields for all of these cards contain associated node labels in the same form as for item 3b. Except for the CROD connection card, most of the customary NASTRAN connection cards can be used.

(c) Last card: Format (A3).
Blank to signify termination of this set. This is also the last input data card.

IV. Program Deck Description

The program is coded in Fortran V for the Univac 1108 Exec-8 computer. One subroutine, TIMER, is installation-dependent. TIMER returns the current central processing unit (CPU) time and is not essential to program execution. It can be replaced by any dummy subroutine of the same name. The program is currently dimensioned to resequence problems containing up to 600 nodes and 1800 distinct edges (one edge per bar, three edges per triangular plate, six edges per quadrilateral plate). Duplicate edges formed by adjacent plates are detected and need not be provided for in the dimensions. Core size for the sum of the instruction and data banks is about 36,000 words. The dimensioning parameters are contained in a Fortran procedure definition processor element PARAM, included under the name PSPEC. The problem size dimensions can be changed here by updating the parameters PE to be equal to the number of edges and PN to be equal to the number of nodes. Comment cards contained in this element list the routines that are to be recompiled when PSPEC is updated. A problem dimensioned for 1000 nodes and 4000 edges was found to require about 58,000 total words of storage. No overlay segmentation is included, although this would be a consideration for problems of larger size.

The program consists of a main program, WAVEFRONT, which calls the subroutine WAVSEQ each time a new sequencing cycle is to be performed. During each cycle, WAVSEQ makes one call to a subroutine FILLUP, which supplies the nodal connectivity matrix, and repeated calls to the subroutine NEXTRO, which selects the next row to be placed in the nodal connectivity matrix, and to ROSTRK, which reorganizes the connectivity matrix to account for rows and columns that are or are not in the current wavefront. The aforementioned main program and four subroutines are the primary operational modules. These and all remaining auxiliary subprograms, which consist of about 600 card images, are listed in Table 1.

V. Suggested Usage

Before using WAVEFRONT or any alternative resequencing program as a finite element program preprocessor, two questions should be considered:

(1) Is it worthwhile to resequence?

(2) If the decision has been made to resequence, what type of sequencing should be used?

The answers to both questions are problem-dependent and no simple rules can be given. Nevertheless, we will make some suggestions that reflect our present practice and experience, particularly as related to NASTRAN processing.

If the problem is small and the stiffness matrix decomposition will be performed within NASTRAN only a limited number of times, the effort of assembling the data for resequencing and the associated computation time will not be justified by associated savings during the NASTRAN program run. To be a little more specific with respect to the implications of "small," this might represent from 75 to 400 or more degrees of freedom, depending upon how effective is the initial sequence that has been established. On the other hand, it might be preferable to label nodes according to a recognizable pattern that lends itself most readily to interpretation and checking and never to expend effort to obtain effective computational sequencing except by an automatic resequencing program.

With respect to NASTRAN processing, the second question resolves itself into a choice between a wavefront or a bandwidth sequencing procedure. Here, the following facts are pertinent:

(1) Wavefronts for structural models do not exceed and are often considerably less than bandwidths.

(2) Although resequencing by a wavefront approach will usually result in a more compact stiffness matrix than when resequenced by a bandwidth approach, NASTRAN presumably does not process an active column (derived from the wavefront) as rapidly as a column within the band. We estimate the expected time penalty to be between 20% and 50%.

(3) The preprocessor computation time for wavefront resequencing is likely to exceed the preprocessor time for bandwidth resequencing.

Although the first two of the above items tend to offset each other, our experience, based on many NASTRAN tests, has been that the decomposition time for a wavefront-sequenced model varies from slightly less than to considerably less than for the same model with bandwidth sequencing.

The third item above indicates that savings in the decomposition time for wavefront sequencing can be partly, or possibly fully, overcome by the additional computation time for wavefront sequencing. One reason that the wavefront sequencing program uses more computation time is that the user has the freedom to specify large numbers of resequencing cycles. Our experience has been that the first few cycles tend to produce the most significant reductions in the wavefront and many additional cycles often produce only minor further reductions. Consequently, a reasonable way to proceed with WAVEFRONT is to perform only a limited number of resequencing cycles in an initial run. Then, if it appears that useful further reductions are possible, perform another run that begins with the row that gave the best result in the initial run. This will tend to allow unfruitful resequencing cycles in the second run to be identified quickly, which will reduce the associated computation time.

Also, in considering possible disadvantages of the longer computation time to perform wavefront rather than bandwidth sequencing, the number of times that the NASTRAN program will be required to perform the decomposition for the same problem should be considered. Usually it takes several NASTRAN runs for a new structural model to eliminate anomalies in topology or constraints. Constraint changes do not invalidate an existing set of sequence definitions and moderate changes in topology will not have a pronounced effect on the usefulness of an available wavefront sequence. The effectiveness of bandwidth sequencing is more vulnerable to topology changes and such changes might call for either repetition of the sequencing run or a user modification of the sequence to enforce an active column. Furthermore, depending upon the charging algorithm at the particular installation, extra CPU seconds during resequencing could be less costly than the same number of CPU seconds in NASTRAN.

VI. Example Results and Comparisons

Reference 1 contains several examples of NASTRAN decomposition time comparisons that show advantages for sequencing by WAVEFRONT with respect to bandwidth sequencing by Rosen's (Ref. 2) program. Here, Table 2 contains some additional comparisons for bandwidth sequencing by the more recently developed BANDIT (Refs. 3 and 4) program. Some of the actual decomposition times with bandwidth sequencing are missing in the table because in these cases NASTRAN was permitted to proceed with the decomposition for only the type of sequencing that was estimated to result in the fastest decomposition.

Nevertheless, in these absences, examination of the NASTRAN estimated decomposition times or the resulting resequenced wavefronts and bandwidths indicates moderate to strong advantages for wavefront sequencing. It might also be noted that the sequencing that produces the faster decomposition times would also produce some additional time savings during forward and backward substitution phases of the load/deflection solution.

The last problem listed in the table shows the importance of sequencing for a moderate-to-small size structure. A bandwidth sequencing was not processed by NASTRAN, but since BANDIT provided a resequenced band equal to only twice the resequenced wavefront, this also would have provided a significant reduction of the decomposition time in comparison with the time for the unsequenced problem.

In summary, the experimental data shown previously in Ref. 1 and here in Table 2 indicate that resequencing of the connectivity matrix prior to NASTRAN processing can produce significant computation time reductions during the NASTRAN run. Furthermore, notwithstanding the possibilities of longer preprocessor computation times to perform wavefront sequencing than to perform bandwidth sequencing, as a general procedure wavefront sequencing appears to be preferential to bandwidth sequencing.

References

1. Levy, R., "Resequencing of the Structural Stiffness Matrix to Improve Computational Efficiency," in *Quarterly Technical Review*, Vol. I, No. 2, pp. 61–70, Jet Propulsion Laboratory, Pasadena, Calif., July 1971.
2. Rosen, R., "Matrix Bandwidth Minimization," in *Proceedings of the 23rd National Conference of the ACM*, p. 585. Brandon/Systems Press, Inc. New Jersey, 1968.
3. Cuthill, E., and McKee, S., "Reducing the Bandwidth of Sparse Symetric Matrices," *Applied Mathematics Laboratory Technical Note*, AML-40-69. Naval Ship Research and Development Center, Washington, D.C., June 1969.
4. Everstine, G. C., "The BANDIT Computer Program for the Reduction of Matrix Bandwidth for NASTRAN," *Computation and Mathematics Department Research and Development Report 3827*, Naval Ship Research and Development Center, Bethesda, Md., March 1972.

Table 1. List of WAVEFRONT routines

Name	Type	Called by	Function
PSPEC	Procedure		Sets dimension parameters
WAVEFRONT	Main program		
WAVSEQ	Subroutine	WAVEFRONT	Controls resequencing cycles
CONNIN	Subroutine	WAVEFRONT	Reads connection card input (bars, rods, plates)
CONV	Subroutine	WAVEFRONT, CONNIN	Converts input free-field node labels to integer
COUNT	Subroutine	WAVEFRONT	Determines maximum, average, rms, wavefront for initial and improved sequencing
UNIFRM	Function	WAVEFRONT	Supplies random number to pick first row
FILLUP	Subroutine	WAVSEQ	Supplies nodal connectivity matrix
NEXTRO	Subroutine	WAVSEQ	Picks the next row in the sequence
ROSTRK	Subroutine	WAVSEQ	Reorganizes connectivity to distinguish currently sequenced and unsequenced rows
CROSS	Subroutine	ROSTRK	Interchanges node labels
SORT	Subroutine	FILLUP	Arranges node labels of connectivity in ascending order
TIMER[a]	Subroutine	WAVEFRONT, FILLUP	Supplies CPU time

[a]Installation: dependent

Table 2. Sequencing and NASTRAN run comparisons

Problem description	Resequencing program	Nodes		Sequencing program		Predominant DOF per node	NASTRAN run			
							B[c]	C[d]	Decomposition time, CPU, s	
		Original	Resequenced	Cycles	CPU, s				Estimated	Actual
Tank, 575 nodes	WAVEFRONT	62[a]	34[a]	1	83	5	15	156	1430	1111
	BANDIT	210[b]	55[b]	3	55		257	0	2500	–
Quadripod, 172 nodes	WAVEFRONT	15[a]	9[a]	95	50	3	22	9	9	6
	BANDIT	37[b]	27[b]	3	41		48	9	22	12
Antenna, 64-m-diameter, 946 nodes	WAVEFRONT	51[a]	44[a]	3	266	3	7	121	783	700
	BANDIT	346[b]	121[b]	3	163		–[e]	–	–	–
Subreflector, 129 nodes	WAVEFRONT	45[a]	6[a]	60	33	6	16	18	14	12
	BANDIT	114[b]	40[b]	3	112		106[f]	96	421	–
Adapter cage, 78 nodes	WAVEFRONT	24[a]	8[a]	25	9	6	12	30	15	12
	None	24[b]	–	–	–		150	0	73	54

[a]In wavefront.

[b]In bandwidth.

[c]Bandwidth, [d]Active columns } DOF allocated by NASTRAN.

[e]No NASTRAN run attempted, severe spilling anticipated.

[f]Approximately 70 decomposition columns spilled.