A COMPARISON OF THE CAPABILITIES OF THREE FINITE ELEMENT PROGRAMS

By

David D. Loendorf
Langley Directorate, U.S. Army Air Mobility
Research and Development Laboratory

SUMMARY

Three finite element programs are compared to assess their capabilities as an analysis tool in a structural design process. Because of the need for repetitive analyses as an integral part of a design loop, a candidate program must be capable of handling large problems, operate efficiently and be readily adaptable for use in computer aided design. The three programs considered in the study, ELAS, SNAP, and NASTRAN, range from a relatively small finite element program limited to static structural analysis (ELAS) to a large complex general analysis system (NASTRAN). Results are given in the paper for comparative speeds and computer resources required for each program in the analysis of sample fuselage problems representative of practical aircraft design.

INTRODUCTION

During the past decade, numerous finite element programs have been developed and are available for public use. They range in size from small programs restricted to two-dimensional static analysis to large systems capable of handling virtually any type of three-dimensional structure subjected to static, dynamic, or thermal loads. Most of the finite element programs in current use were initially developed to analyze a prescribed structural design to determine, for example, if stress levels are within allowable limits. More recently, however, researchers have attempted to incorporate these finite element programs as an integral part of an automated design process. In design applications, many cycles are often required to obtain a converged design, and the efficiency of the automated design procedure is strongly dependent on the efficiency of the analysis tool.

It is the purpose of this paper to present selected analysis results obtained with three finite element programs in current use and discuss some of their assets and liabilities when considering their inclusion as the analysis phase of an automated structural design program.

ANALYSIS REQUIREMENTS FOR AUTOMATED STRUCTURAL DESIGN

The analysis program is the backbone of any automated structural design procedure and, hence, must efficiently handle the repetitive computation of stresses and deflections following each structural modification. A very simplified schematic of a structural design process is shown in figure 1. It has been the author's experience with the development of the fuselage design program FADES, reference 1, that the analysis program is executed approximately 11 times to obtain one design and that this accounts for more than 75 percent of the total CPU time required for the design. In an effort to decrease computer cost, the finite element analysis program must be evaluated carefully to determine what design oriented finite element capabilities are needed. Since it is important that the actual structure be closely approximated, many diverse finite elements are required. At the same time, analytical results should be obtained with minimum computer costs. However, one should not look at efficiency alone, but must also consider program capabilities and interface problems associated with integrating the finite element program into the design algorithm. These general requirements lead to specific requirements for a structural analysis program in the following areas:

1. Efficiency
   a. Core requirements
   b. Execution time
   c. Bandwidth minimization or sparse matrix techniques

2. Generality
   a. Static, dynamic, buckling, etc., capabilities
   b. Large problem capability
   c. Checkpoint/restart capability
   d. Large library of elements
   e. Plot capability

3. Interfaceability
   a. Standardized input
   b. User determined, file oriented output
   c. Complete, concise, and accurate documentation
   d. Machine independence
   e. Continued maintenance

To the author's knowledge, no finite element programs currently available are specifically tailored to efficiently meet all of the above design oriented capabilities; thus, one must consider suitable alternatives among existing programs. Three programs were considered in this study ranging from moderate to broad in capability. The programs are ELAS, reference 2, a relatively small finite element program limited to static structural analysis and presently used in the FADES program; SNAP, reference 3, a proprietary static analysis finite element program efficient for large structures; and NASTRAN, reference 4, NASA's general purpose structural analysis program. (Some of the more important capabilities are listed in Table I).

RESULTS AND DISCUSSION

A structural configuration representative of an aircraft fuselage section was used in the studies to compare the three programs. A model of the configuration is set up so that the number of rings, stringers, and floor members was easily changed through a model generating program which prepared input to all three programs (see figure 2 for two sample configurations). This procedure facilitated running identical problems with the three programs with a minimum of intermediate effort. Ring elements and transverse floor elements were modeled using a typical beam formulation (combined bending and extension). Stringer elements and longitudinal floor elements were modeled by a rod formulation (extensional properties only). Skin sections were modeled using the constant strain membrane plate element in NASTRAN and ELAS, and the Pian hybrid membrane in SNAP. A comparison of the membrane formulations may be found in references 5 and 6. The loading in all cases was a self-equilibrating compressive force applied to each of the grid points of the end rings. Problem size ranged from 90 grid points, 360 D.O.F., and 267 elements to 1530 grid points, 6102 D.O.F., and 4415 elements.

All results were obtained using the CDC 6000 series computers at the Langley Research Center. ELAS was run using version 75, SNAP using version J, and NASTRAN using level 15.1.1, a pre-release form of level 15. Displacements for the SNAP results tended to be about 5 percent larger than for NASTRAN or ELAS and the difference is attributed to the relatively flexible hybrid element contained in SNAP. The stresses obtained for all programs agreed to within 1 percent for all cases.

The effect of problem size on core requirements for the three programs is shown in figure 3. The steep slope of the ELAS curve is due to the fact that ELAS requires the complete stiffness matrix in-core during execution. Both SNAP and NASTRAN are not limited by this requirement and, therefore, can handle fairly large problems in a minimum of core; the SNAP core requirement is the lesser of the two.

Total execution times for a number of configuration sizes are shown in figure 4 for SNAP and NASTRAN. ELAS times are not included because of problem size; however, for smaller problems, ELAS and NASTRAN execution times are similar. The top three curves indicate NASTRAN execution times increase with increasing problem size. These also show that time increases with increasing number of grid points per ring. The bottom three curves show run times for SNAP and indicate that on the contrary, execution times for SNAP decrease slightly with increase in grid points per ring.

For NASTRAN, the effect of grid points per ring on decomposition times is shown in figure 5. The sharp increase in decomposition time shows that if NASTRAN is to be efficient, the analyst must be able to minimize the problem bandwidth, preferably by the use of an automatic bandwidth minimization scheme such as BANDIT (reference 7).

The effect on solution times of using BANDIT to generate SEQGP cards for NASTRAN is shown in figure 6. The curve labeled NASTRAN 15.1.1 shows results for problem solutions without any resequencing. The curve labeled NASTRAN/BANDIT shows execution times of NASTRAN plus the execution times required by BANDIT which are shown in the curve marked BANDIT. These results show the benefits resulting from improved grid point sequencing. Also shown on the figure are execution times for SNAP which indicate that problem solution times are less than the times required to resequence grid points for NASTRAN, when SNAP is run with a good grid point elimination sequence. However, one must input the reduction sequence for SNAP as there are no available algorithms capable of doing this for the analyst.

The results discussed thus far have been restricted to computer time and storage. A more realistic comparison is to put computer resources on a cost basis. While computer cost algorithms vary among computing centers, any reasonable algorithm provides a basis for comparison. Cost presented herein was calculated by the cost algorithm currently used at the NASA Langley Research Center which takes into account operating systems calls (O/S calls), CPU time, and CORE.

A comparison of SNAP and NASTRAN computer requirements for a large problem consisting of 1530 grid points, 4415 elements, and 6106 D.O.F. is shown in figure 7. Both SNAP and NASTRAN were run in 160 000 octal core locations. As shown in the figure, NASTRAN requirements exceeded those for SNAP in all categories. In particular, the ratio of total time is about 5.3:1 while cost is approximately 4.5:1. These figures do not reflect the fact that BANDIT was run for NASTRAN to resequence the grid points in an effort to minimize NASTRAN costs, as SNAP was running under an optimum reduction sequence.

The above discussion focuses on the efficiency of the solution process. However, one must not exclude the other points made earlier (2. a-e; 3. a-e, p. 278), as they, too, must be considered in overall efficiency considerations. For example, if a design program is built around an efficient static analysis program, no capability will exist for mode shapes and frequencies. Thus, program generality may be as important as efficiency considerations. For such a case, NASTRAN is the only program of the three that has a broad range of analysis capability. The SNAP program has a free vibration counterpart, but at present, it is a separate program and requires regeneration of input.

User conveniences are also important if the analysis program is to be easily interfaced with a design algorithm. All of the programs have well documented, standardized input for which an input generating program may be developed to help reduce errors in input. All three programs also have some form of internal data checking with diagnostic error printout. However, only one of the programs, SNAP, allows the user to define desired output and on which files it should be placed. This is very desirable from the standpoint of automated design if different elements are designed at different times. Current theoretical documentation is adequate for NASTRAN only. NASTRAN and ELAS are operational on three machines, CDC, UNIVAC, and IBM, while SNAP is operational on CDC and UNIVAC. NASTRAN is being maintained under contract by the NASTRAN System Management Office at Langley Research Center; SNAP is maintained by its developer; and ELAS is being updated by its author at Duke University.

CONCLUDING REMARKS

Three finite element programs were studied to determine their feasibility as the analysis tool in automated structural design. Due to problem size limitations, ELAS does not appear to be suitable for this purpose. The present speed of SNAP makes it desirable in a design environment where many repetitive analyses are required. However, the generality of NASTRAN may overshadow the fact it is less efficient than SNAP. Thus, one must weigh all present and future requirements before deciding on which program to choose. It should be clear, however, that bandwidth can have a significant effect on computer costs and perhaps NASTRAN should be extended to include a band optimization scheme, or the decomposition procedure should be improved. It is very likely that a complex design system could provide the option of using either SNAP or NASTRAN until NASTRAN is extended to provide the speed offered by SNAP.

REFERENCES

1. Sobieszczanski, J. E., and Loendorf, D. D.: A Mixed Optimization Method for Automated Design of Fuselage Structures. Presented at the 13th AIAA/ASME/SAE Structures, Structural Dynamics and Materials Conference, San Antonio, Texas, April 1972.

2. Utku, Senol: ELAS--A General Purpose Digital Computer Program for the Linear Equilibrium Problems of Structures. Structural Mechanics Series No. 11, School of Engineering, Duke University.

3. Whetstone, W. D.: Computer Analysis of Large Linear Frames. J. Str. Div., ASCE, ST11, Nov. 1969.

4. Butler, Thomas G., and Michel, Douglas: NASTRAN--A Summary of the Functions and Capabilities of the NASA Structural Analysis Computer System. NASA SP-260, 1971.

5. Whetstone, W. D., and Yen, C. L.: Comparison of Membrane Finite Element Formulations. Lockheed Missiles and Space Company Report #HREC 6-81-70-3, LMSC/HREC D162553, Huntsville, Alabama.

6. MacNeal, Richard H.: NASTRAN Theoretical Manual. Ed. NASA SP-221.

7. Everstine, Gordon C.: The BANDIT Computer Program for the Reduction of Matrix Bandwidth for NASTRAN. NSRDC Report #3821, March 1972.

TABLE I

CAPABILITIES OF THREE FINITE ELEMENT PROGRAMS

	CAPABILITY	ELAS	SNAP	NASTRAN
1.	Complete Stiffness Matrix In-core	X		
2.	Sparse Matrix Techniques		X	
3.	Good Documentation	X[0]	X[0]	X
4.	Static Solution	X	X	X
5.	Dynamic Solution for Transient Loadings			X
6.	Eigenvalue/Frequency Solution		X	X
7.	Buckling			X
8.	Cholesky Decomp.	X		X
9.	Symmetric Gaussian Elimination		X	
10.	Automatic Bandwidth Minimization	X		X[1]
11.	Plot Capability (Undeformed & Deformed)		X	X
12.	Restart/Checkpoint		X	X
13.	Large Family of Users	X		X
14.	Multiple Loadings		X	X
15.	User Determined Output		X[2]	
16.	Interface Capabilities Good	X		
17.	Many Element Types	X	X[3]	X

[0]Input/Output only

[1]BANDIT is available for external re-sequencing

[2]Specific files may be designated

[3]Pian hybrid formulation for plate membrane and bending formulations

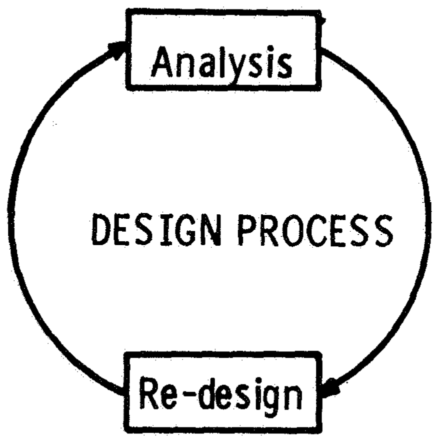
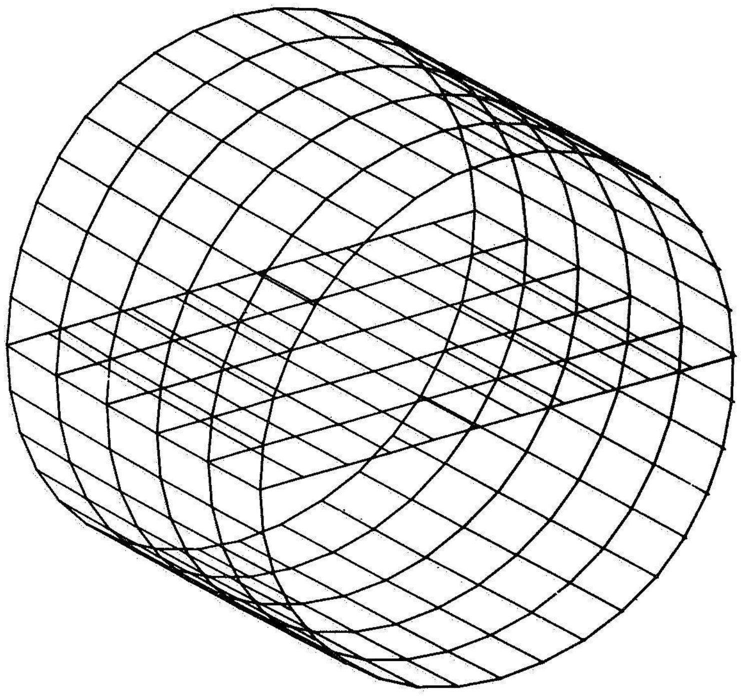
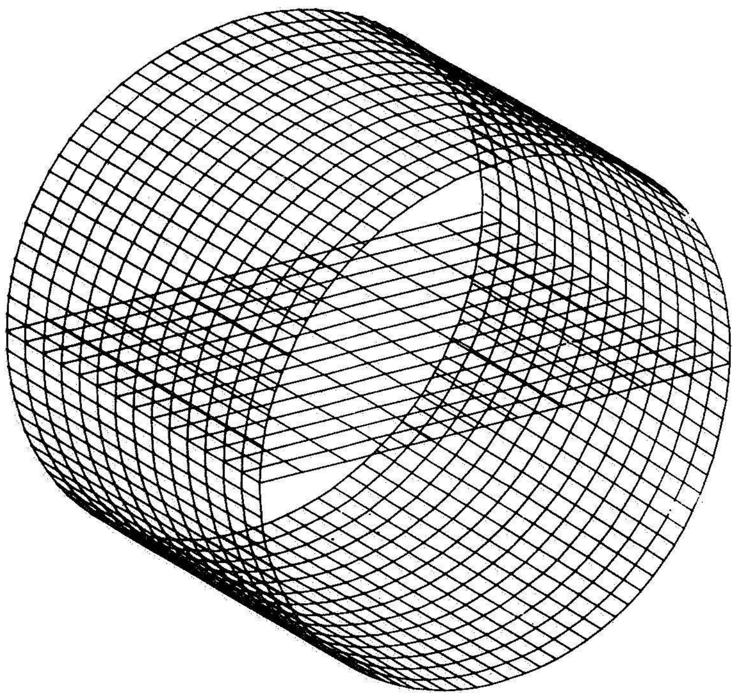


Figure 1. - Simplified schematic of the design process.



276 Nodes 1098 D.O.F. 752 Elements



1530 Nodes 6114 D.O.F. 4415 Elements

Figure 2. - General Configurations.

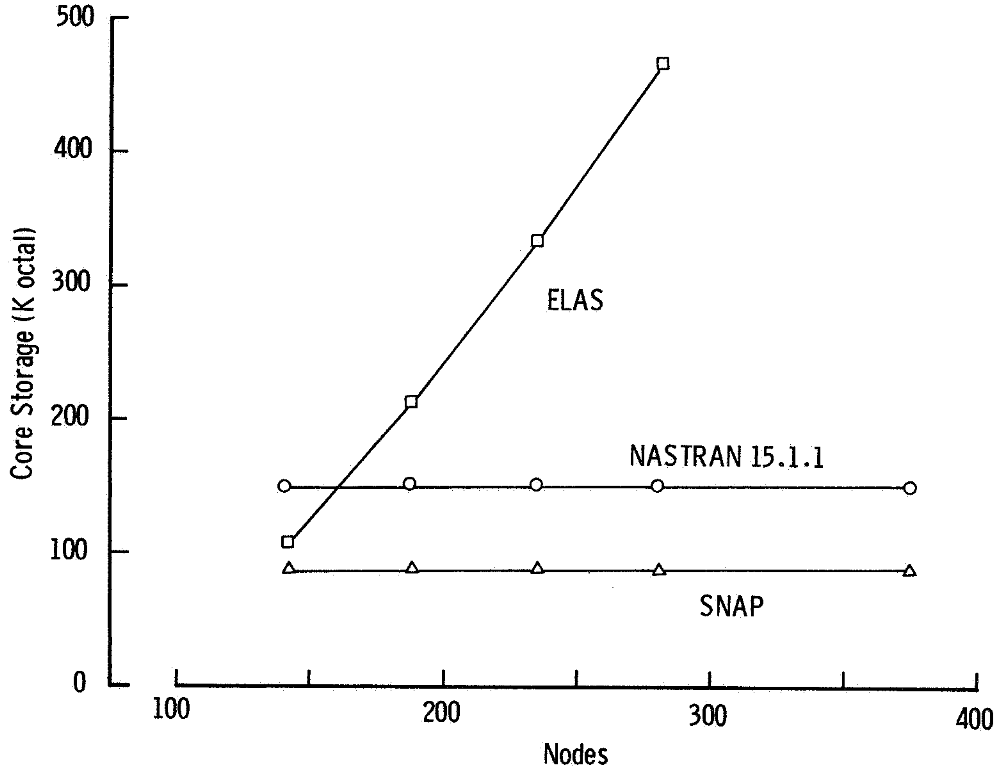


Figure 3. - Core requirements for various problem sizes.

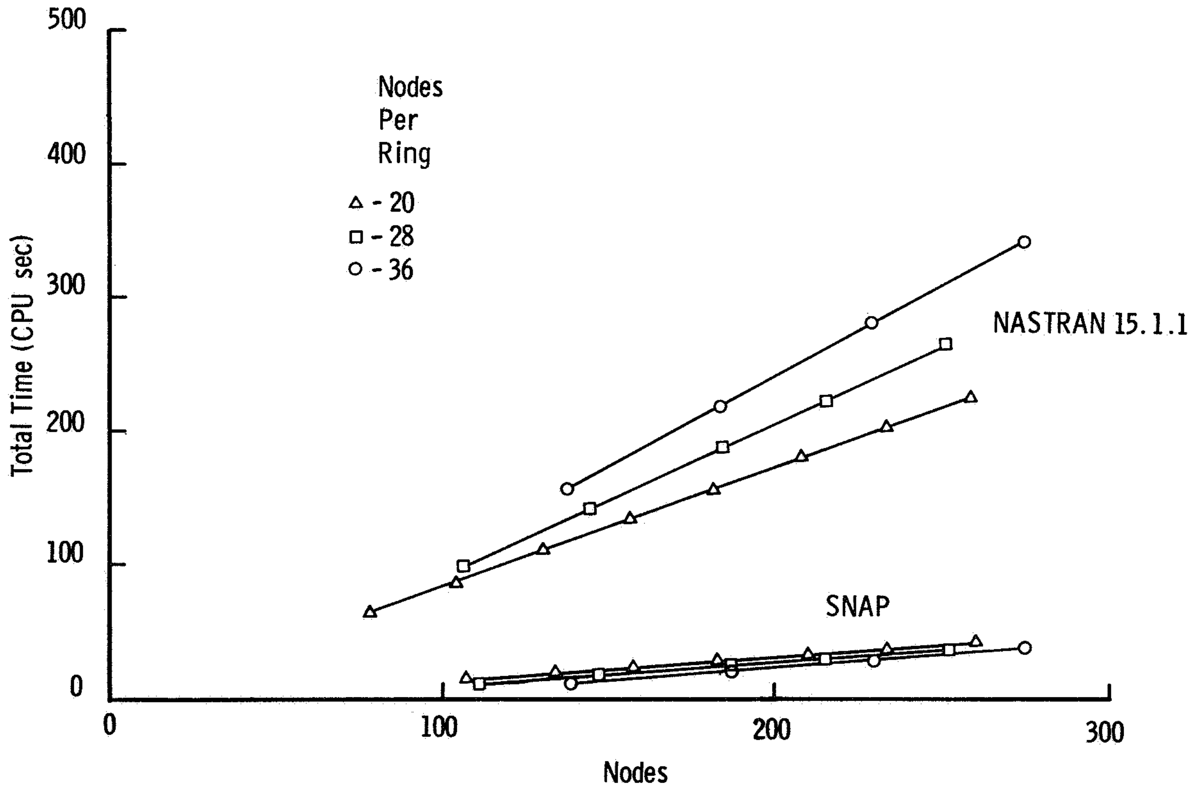


Figure 4. - Total CPU time for various numbers of circumferential nodes.

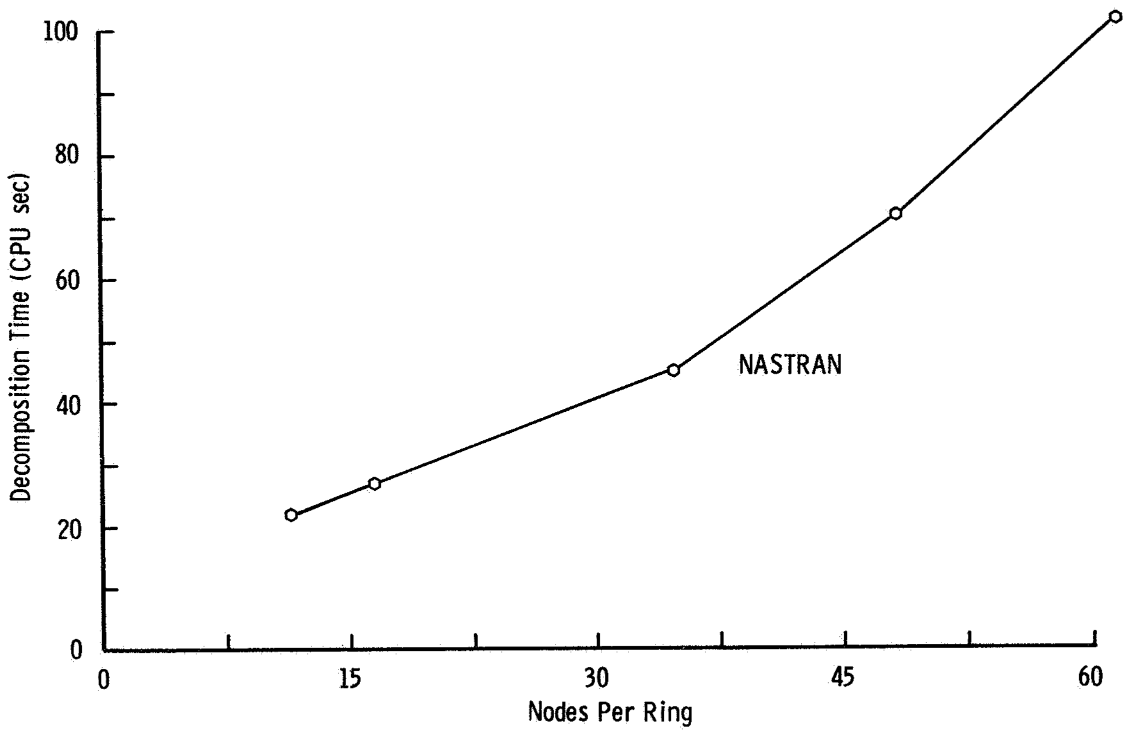


Figure 5. - Effect of nodes per ring on decomposition time.
(Total nodes held constant)

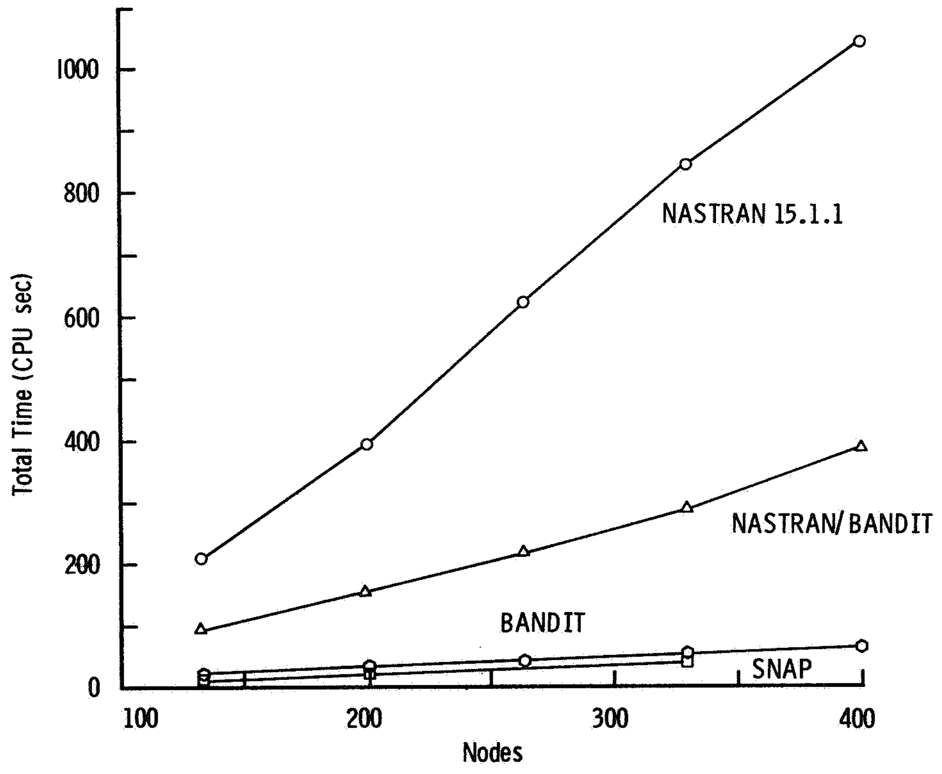


Figure 6. - Effect of BANDIT on NASTRAN 15.1.1 execution time.
(52 nodes per ring)

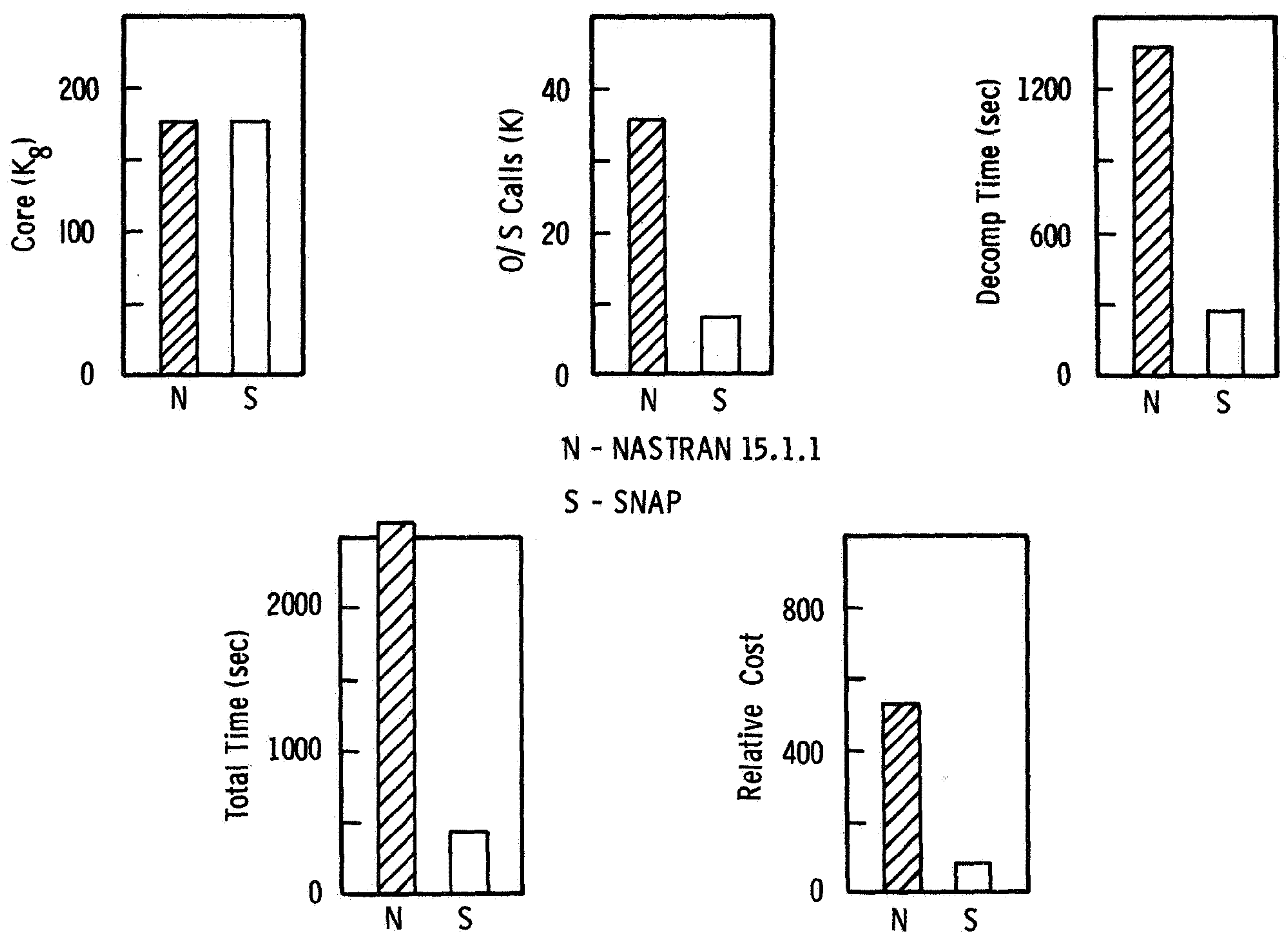


Figure 7. - Comparison of SNAP and NASTRAN computer requirements. (1530 grid points, 6102 D.O.F., 4415 elements)