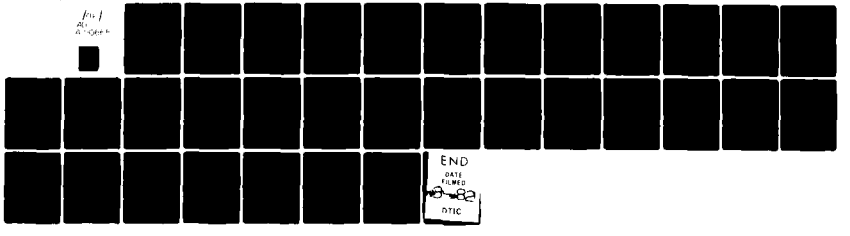
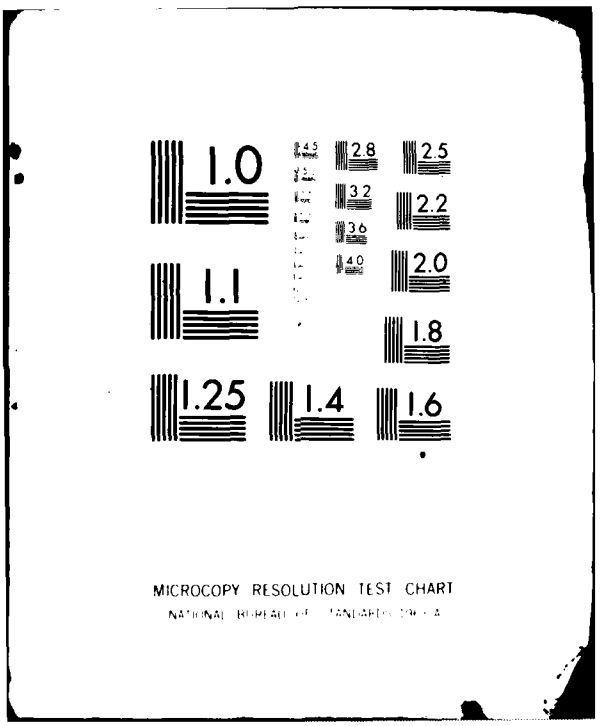
AD-A110 665 APPLIED SCIENCE AND TECHNOLOGY POWAY CA F/G 13/13
THE INTERAGENCY SOFTWARE EVALUATION GROUP: A CRITICAL EVALUATIO--ETC(U)
DEC 81 R E NICKELL N00014-79-C-0620
UNCLASSIFIED AST-81-2 NL



END
DATE FILMED
3-82
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AST-81-2	2. GOVT ACCESSION NO. AD-A110	3. RECIPIENT'S CATALOG NUMBER 665
4. TITLE (and Subtitle) The Interagency Software Evaluation Group: A Critical Evaluation of the ADINA, NASTRAN, and STAGS Structural Mechanics Computer Programs		5. TYPE OF REPORT & PERIOD COVERED Technical Report No. 2 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Robert E. Nickell (1)		8. CONTRACT OR GRANT NUMBER(s) N00014-79-C-0620
9. PERFORMING ORGANIZATION NAME AND ADDRESS Applied Science & Technology 16630 Sagewood Lane Poway, California 92064		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Department of the Navy, Office of Naval Research, Structural Mechanics Program (Code 474) Arlington, Virginia 22217		12. REPORT DATE December 1981 13. NUMBER OF PAGES 30
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) LEVEL II		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sales distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

DTIC ELECTE FEB 9 1982 D H

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software evaluation; structural mechanics; computer programs; finite element; computation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report provides the summary for the first round of structural mechanics software evaluations by the Interagency Software Evaluation Group, consisting of the evaluation of the codes ADINA, NASTRAN, and STAGS. The evaluation criteria are discussed in some detail.

37 916

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

THE INTERAGENCY SOFTWARE EVALUATION GROUP: A CRITICAL EVALUATION OF THE ADINA, NASTRAN, AND STAGS STRUCTURAL MECHANICS COMPUTER PROGRAMS

by

Dr. Robert E. Nickell
Applied Science & Technology
16630 Sagewood Lane
Poway, California 92064

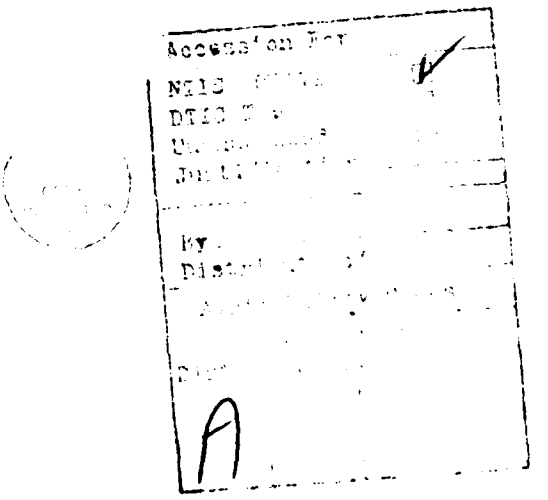
FINAL REPORT
CONTRACT NO. N00014-79-C-0620
Office of Naval Research
Arlington, Virginia 22217

December 1981

FORWARD

This final report is the culmination of almost four years of involvement with the Office of Naval Research (ONR) in developing the concept of critical evaluation of engineering applications software. The mechanism for the evaluations proved to be the Interagency Software Evaluation Group (ISEG), a cooperative venture of Defense Department research and development laboratories, and other government agencies. The initial exercise consisted of the evaluation of the ADINA, COSMIC NASTRAN, and STAGS structural mechanics codes, under the supervision of Dr. Nicholas Perrone, Director of ONR's Structural Mechanics Program, and the ISEG governing body.

We hope that the results of this initial venture justify the continued use of critical third-party evaluation as a means of providing improved software quality assurance, better transfer of information between developers and users, and a standardization of software comparisions. This writer would also like to thank ONR and Dr. Perrone for their continued support, the ISEG for the opportunity to be a part of an exciting project, and the evaluation contractors for many stimulating and thoughtful discussions.



I. INTRODUCTION

The finite element method has attained a level of maturity and acceptability that few would have predicted three decades ago. In common with many other developing technologies, the scientific origins of the method (as a piecewise continuous variational approach) were insufficient to establish its powerful capabilities until practicing aerospace and structural engineers recreated an operational variant. Many years elapsed before the mathematical basis and engineering practice were reconciled. The first decade can therefore be characterized as a time of creativity and original application, and the software was developed principally for use by the developer or by, at most, a few sophisticated and personally-trained associates of the developer.

During the second decade the technology matured to the point that the commercial potential of the associated software began to be realized. The simultaneous improvements in finite element methodology, computer hardware, and higher-order programming languages produced an entrepreneurial climate in which developers began to write software specifically for second-party usage. The second parties, called users, learned of software developments primarily through the medium of the technical conference/symposium sponsored by the professional engineering societies, such as the American Society of Mechanical Engineers (ASME) and the American Society of Civil Engineers (ASCE), but the concept of the short course to intensively train potential users became a rapidly spreading adjunct.

The relationships between the first parties, or developers, and the users evolved during the decade as the market place became more sophisticated. Many developers failed to appreciate the user perspective, to whom the software was opaque, while most users had little appreciation for the problems facing the

developer, who generally operated under severe organizational and financial constraints. It should be noted that the commercial software market place was born from an incorrect set of initial conditions; i.e., the cost of developing and maintaining finite element software--and, in many organizations, the computing costs, as well--were well hidden from the scrutiny of higher levels of management. For most organizations, a finite element capability seemed to be virtually free.

Because of these false precepts, the decade 1965 - 1975 proved to be most instructive with respect to the appreciation of true costs on the parts of both users and developers. Research and development conducted by the federal government was the cornerstone in this learning process. When the National Aeronautics and Space Administration (NASA) undertook the NASTRAN project, not only did the sponsor and the project personnel gather accurate data on the true costs of general purpose finite element software creation, but the elements of the commercial software industry were able to calibrate their efforts accordingly, often to their chagrin. At the same time, organizations essentially characterized as users--including those doing some software development on a cost-concealed basis--began to realize the true costs of software development and maintenance, even when practiced on a small scale.

By the late 1970's, agencies of the federal government began to realize that the NASTRAN project and other budgetary-visible projects were but the tip of a software iceberg, and pressure began to build to evaluate--both from technical and cost-benefit points of view--the software being developed (both directly and indirectly) under federal government sponsorship. One of the lead agencies in this regard was the Office of Naval Research, through its Structural Mechanics Program. One of the early contributions was the

sponsorship of the Conference on Numerical and Computer Methods in Structural Mechanics, held in Urbana, Illinois, September 8-10, 1971 [1]. In addition to a discussion of the ONR STORE Project (The Structures Oriented Exchange), this conference produced critical reviews of a variety of general purpose structural mechanics software, including ASKA, NASTRAN, DAISY, STARDYNE, STRUDL, and MARC.

At that particular stage of development of general purpose finite element software, the market place forces were marginally effective. Developers and users had significant commitments to each other, involving a synergy of effort to develop, debug, enhance, and qualify the particular software for the production computing environment. Thus, although the reviews were meant to be critical, the distinction between the first party and the second party was blurred. In spite of this, the conference offered the first attempt to evaluate software from a user's perspective, generally emphasizing such transparent features as the element library and the results of sample problems. An exception to this emphasis was the article by Tocher and Herness [2] on NASTRAN, a contribution which was an outgrowth of an extensive evaluation of the NASTRAN code by the Boeing Company.

ONR next cosponsored with the National Science Foundation a symposium at the University of Maryland in June 1974 dedicated to a different approach to software evaluation [3]. A group of investigators were asked to write assessments of the state of structural mechanics software capability in prescribed technical subdisciplines -- such as plastic analysis, fracture mechanics, shock wave propagation, welding, composites, thin shells, etc. Such a subdisciplinary assessment would have permitted some direct comparisions between different programs purporting to have the same

subdisciplinary capability. The contributors were selected with a concern toward balance between developer and user, but with some emphasis on a degree of contributor sophistication felt to be necessary to properly assess the field.

The symposium was successful in meeting its objectives. Nevertheless, both during the planning of and following the symposium, the individuals involved developed a greater appreciation for the talent, the degree of commitment, and the resources required for an investigator to assess the applications software in even one subdisciplinary area. Hand in hand with this realization was the fact that such talent is relatively scarce, especially outside the commercial software developer organizations and the national laboratories. From this point on, the thrust of much of the effort within the professional engineering societies and the federal research establishment was reoriented toward more effective use of available human and financial resources, including discussions of such new institutional forms as a national engineering software center.

II. THE INTERAGENCY SOFTWARE EVALUATION GROUP

Based upon the foregoing considerations, a meeting was held at ONR on February 11, 1977, to review the current state of affairs with respect to software evaluation and to discuss the feasibility of formal third-party evaluation. Representatives of all three armed services attended, along with others from the U.S. Nuclear Regulatory Commission and the then Energy Research and Development Administration. Representatives of the National Science Foundation and the National Bureau of Standards were invited, but were unable to attend. The meeting produced a concensus that: (1) third-party

evaluation was useful and, perhaps, essential; (2) a means for institutionalizing such third-party evaluations should be sought; (3) realistic costs for such efforts should be recognized at the outset; and (4) criteria were needed for selecting the software to be evaluated, the institutions and individuals to carry out the evaluations, and criteria for the evaluations themselves.

A follow-up meeting was held at ONR on April 5, 1977, at which time an draft document was presented by N. Perrone describing the Interagency Software Evaluation Group (ISEG). The ISEG was envisioned as a preliminary step on the way to a national engineering software center concept, an institutional framework that could have responsibility for continuous evaluation of various engineering applications software, in addition to other functions. Since the national engineering software center was felt to require a significantly higher threshold of concensus, the ISEG route was seen as being feasible in the short term. In addition, a set of preliminary criteria for selecting software, selecting contractors, and evaluating software were developed. These criteria will be discussed briefly in subsequent sections of this report, but have already been published [4].

The ISEG is formally composed of Army, Navy, and Air Force contributors, but other government organizations — such as the National Science Foundation, the Department of Energy, and the Nuclear Regulatory Commission -- are advised of progress because of their interest in both engineering applications software and the ISEG. The initial funding received from the contributing armed service units was sufficient to examine between three and four applications software packages, depending upon the complexity of the particular code. The National Science Foundation was expected to idependently

fund another evaluation effort, using the same criteria developed by the ISEG.

Since the directly contributing agencies were the three armed services, the screening of software involved a survey of software usage* within armed service research and development laboratories, such as the U.S. Army Materials and Mechanics Research Center, the U.S. Army Missile Research and Development Center, the David W. Taylor Naval Ship Research and Development Center, the Naval Research Laboratories, the U.S. Air Force Flight Dynamics Laboratory, and the U.S. Air Force Weapons Laboratory. In most cases the software had already been screened for other purposes [5,6], but some additional data was gathered from personal contacts within the R & D laboratories. Based upon this data, the codes were identified by type (large general purpose linear, large general purpose nonlinear, special purpose plate and shell, etc.); frequency of use; multi-laboratory interest; availability of developer support; and potential.

The survey revealed that, although literally hundreds of different structural mechanics codes are available for use by engineers in the armed service R & D laboratories, a few codes constitute the bulk of actual applications. In general, these few codes are also those with the greatest multi-laboratory interest, those with the greater availability of developer support (in a few cases, notably NASTRAN, the armed service R & D laboratories have made the commitment to support the software in house with personnel having qualifications equal to those of a developer), and those with the greatest potential for expanded usage. This generality should be tempered by

---

* Structural mechanics software was selected for this initial series of evaluations, with the understanding that the evaluation criteria should be generally applicable to fluid mechanics and heat transfer software, as well as that from other sub-disciplines of applied physics.

the observation that individual users develop strong loyalties and attachments to software with which they become familiar. Many of these users tend to overestimate the potential value of this software, to the exclusion of other packages with equal or greater capability. It is for this reason that some organizations spend substantial sums of money to create preprocessors that accept input in a variety of formats--corresponding to the special purpose codes with which the organization's users are familiar--but then transform the input into the format required for some standard general pupose package. A similar transformation may be required for the output, as well. The success of this approach depends upon the cost trade-offs between continuing to maintain--either directly or indirectly--a large number of special purpose codes and the costs of developing the encode/decode pre- and post-processors. An additional factor is that all new users can be introduced to the standard general purpose format.

It should be pointed out that the availability of developer support and the potential of the software go hand in hand, and both are related to the life span of the package. A good deal of the undocumented or partially documented structural mechanics software, much of it written at universities for specific research projects by neophyte programmers, has a lifetime of only a year or two. The average lifetime of a piece of applications software is estimated to be between five and seven years, while only those packages with extensive developer support (maintenance, debugging, additional features, reissue of software on a periodic basis) seem to survive for a decade or more. Therefore, the potential of the software to impact the acquiring organization depends on long-term availability which, in turn, depends upon the degree of developer support.

With these items in mind, the ISEG selected one code from each of the major types. Within the category of linear, general purpose structural mechanics software, the selection of COSMIC NASTRAN was obvious—primarily because of its frequency of use and its multi-laboratory interest. The SAP family of codes were a distant second, but had a strong following. For this reason SAP was recommended to the National Science Foundation as a package to be evaluated outside the ISEG framework, although using the same evaluation criteria. No other piece of structural mechanics software in the linear, general purpose category received sufficient support to be considered in the initial series of evaluations.

Within the category of nonlinear, general purpose structural mechanics software, several codes received strong support—including MARC, ADINA, and a relatively new entry, ABAQUS. All of these codes are marketed in a quasi-proprietary mode, involving paid-up leases on source/object code, commercial data center royalties, user group fees, or other arrangements. Because of the quasi-proprietary status, each user organization makes a virtually independent financial arrangement with the developer, which generally restrict the transfer of the code to a third party. Since it was the desire of the ISEG to place no non-technical impediments in the way of the contractor selection process, ADINA was chosen. ADINA is a research (as opposed to a production) code available through Professor K. J. Bathe at the Massachusetts Institute of Technology. The acquiring organization becomes a member of a loosely-defined users group for an annual fee which is nominal in comparison to the fees charged for production codes. Because of the relative simplicity of the coding structure, many organizations use ADINA as the basic building block for in-house capability.

MARC and ABAQUS were reasonable alternatives. However, few potential contractors (other than the developer) were available who would have been in a position to evaluate the MARC programming architecture in any other than a transparent sense. The code is also not widely used within the armed service R & D community because of the training required to use it effectively. The ABAQUS code was a late entry in the discussions with strong support from those who felt that its robust programming and data management architecture offered a potential for long-term use. At the time the decisions were made, however, the code was not yet available for evaluation except in a rather primitive form. Any future ISEG evaluation effort should consider ABAQUS as a prime candidate.

Within the category of special purpose plate and shell structural mechanics software, a type that is widely applied within the armed services R & D laboratories, the leading candidates were the BOSOR series of codes and the STAGS code. All of this software is readily available from the Lockheed Palo Alto Research Laboratory, and the software receives continuing support at various levels and from various sources within the Department of Defense (DOD) and the National Aeronautics and Space Administration (NASA). Both candidates are fairly widely used, but the ISEG collectively reasoned that STAGS had a greater potential for future applications. Also, the BOSOR user community seemed to be relatively comfortable with their product, whereas STAGS was a somewhat more controversial and misunderstood quantity. Therefore, the STAGS code was chosen for evaluation in this category.

Another possible choice might have been to examine an entire spectrum of shell-of-revolution codes—finite element, finite difference, and other—so as to compare one code to another directly. This approach was not selected for

two reasons: first, users tend to select a shell-of-revolution code on a somewhat subjective and specialized basis (e.g., branched-shell capability, ability to model discontinuous curvatures, sandwich shell capability, orthotropy, etc.); second, the evaluation criteria do not require direct comparison between codes, since it was realized that such comparisons can often be tailored to produce a desired result.

Therefore, the codes selected for the ISEG evaluation were COSMIC NASTRAN, ADINA, and STAGS, with the SAP code recommended to the National Science Foundation for their evaluation. The ABAQUS code was targeted for future considerations.

III. EVALUATION CRITERIA

The details of the evaluation criteria are provided in [4], but are summarized briefly here for completeness. The original criteria have been slightly modified to reflect the experiences and discussions of the past two years.

First, the criteria must reflect the third-party status of the evaluator. This is a generalization of the independent test organization approach suggested by Deutsch [7] in his discussion of software verification and validation. The independent test organization serves a quality assurance purpose, which is certainly within the scope of the ISEG evaluations, but the concept of a third-party evaluator can go much further. Technology transfer can be accommodated, provided that the third party has the requisite skills to bridge the gap between the developer, or first party, and the user, or second party. The evaluation may help the user to better understand the technical capabilities and limitations of the software, while at the same time helping

the developer to plan orderly improvements. Principally, however, the third-party critical review criteria may help to establish some uniformity in the procurement of structural mechanics software.

Second, the criteria should reflect the several manifestations of the software being evaluated. To some individuals, only the documentation is of interest. These people are concerned about how easily analysts, system programmers, etc. can be trained to use the software, whether or not the theory behind the program has an adequate basis, and whether or not rudimentary steps toward program verification have been taken. To other individuals, the source code itself is the manifestation of interest. These people are concerned about data management, modularity, logical flow, comment cards, and other features that will help or hinder the task of software maintenance and modification. A third group of individuals are interested only in the results generated by the software. They treat the code as a black box which must be characterized for each new range of application.*

The evaluation criteria were divided into four groupings: (i) documentation criteria; (ii) program architecture criteria; (iii) functional description criteria; and (iv) advanced evaluation criteria, also referred to as advanced evaluation exercises.

The documentation criteria were adapted from Henrywood [8] and from personal observations. The documentation should ideally consist of four manuals--a theoretical manual, a programmer's manual, a user's manual, and a verification example manual; an adequate set of comment cards throughout the

---

* We refer to this characterization as qualification; i.e., the user attempts to demonstrate that for this particular combination of geometry, material response, loadings, and program configuration, the code executes successfully.

coding; and adequate documentation throughout the output files. The latter two items are somewhat subjective, but the intent is clear: the number of comment cards should permit the skilled programmer to define the purpose of blocks of coding; also, the output file documentation should enable the user (or, for that matter, a reader other than the user) to identify, for example, stress, strain, and displacement components, their units, material data, etc.

The theoretical manual should contain the basic information on such items as the kinematic formulation, the constitutive theory, the element formulations, the eigenvalue extraction algorithm, the time integration approach, and other features. The programmer's manual should provide the description of the code architecture, subroutine definitions, common block arrangements, array definitions, file management information, and similar data sufficient to permit a transfer of code maintenance responsibility from the developer to another organization. The user's manual is critical, since it will leave a lasting impression on the largest number of people. A good piece of software with a poor user's manual will be poorly received by the market place, whereas a code with limited capability can be commercially successful with an excellent user's manual. The principal criteria are clarity, use of established terminology, good indexing, and judicious use of default options in the coding to avoid excessive user decision-making. Finally, the verification manual should contain a sufficient number of examples to provide confidence in the theories upon which the code is built, as well as to enable the user to find data input profiles that cover a wide variety of applications and options.

The program architecture evaluation begins with the programmer's manual and continues on to a study of the software design. This is particularly

important for applications for software of the type evaluated in the ISEG effort. When software is designed at the outset with multiple users and multiple host machines in mind, greater attention is paid to the trade-offs between efficiency and reliability. Since efficiency is so often dictated by local machine characteristics, such as input/output, loader, and vectorizing options, the developer may choose to isolate those functions within readily accessible modules. The remainder of the code should then adhere strictly to established FORTRAN standards, in order to assure portability. Reliability can be designed into the code in many ways. Some of the more obvious are: (i) Subroutine length and function control, i.e., the length and the range of function of the subroutine are tightly controlled, so that extraneous logic paths are avoided, and the range of input and output variables does not go beyond valid limits; (ii) organization and management of data bases, e.g., in finite element codes, the data bases are rather naturally aggregated into a nodal point data base, an element data base, an integration point data base, and a material data base; (iii) careful use of logical flags, which are one of the foremost contributors to unreliable coding, primarily because of the tendency to override complex logic, rather than restricting the logic, when adding capability or fixing bugs; (iv) global, rather than local, control of dynamic storage; (v) use of calling sequences, rather than common blocks, to transmit data to and from subroutines; and (vi) modularization of common procedures, such as matrix operations, invariant calculations, etc.

The functional description begins with the theoretical manual, where physical principles underlying the code are defined, the mathematical statements of these principles are given, and the mathematical algorithms are developed. The coding practice used by the developer completes the story.

The functions to be described include, but are not limited to:

1. The discretization approach - finite element, with its weak satisfaction of equilibrium and traction boundary conditions, and its strong satisfaction of energy conservation and kinematics; or finite difference, which differs in that the kinematic relations also possess weak solutions; or some other method.

2. The time integration approach - whether modal superposition or direct integration, whether the particular approach has unconditional stability, built-in stability controls, artificial damping, its order of accuracy, and other factors.

3. The approach for solution of simultaneous equations - (including procedures for treatment of nonlinear terms) whether by iterative, direct, or semi-direct methods; whether pivoting is used; what the measures of ill-conditioning are; storage limitations and file manipulation required; etc.

4. The kinematic approach - the strain-displacement relations incorporated; kinematic constraints and transformations allowed;

5. The constitutive approach - the stress-strain relations allowed; whether strain-rate effects are included; any limitations with respect to anisotropy; and others.

6. Special Features - an example might be a simultaneous solution of heat transfer or fluid mechanical fields, or the ability to interface with such solutions.

The critical evaluator has the responsibility for determining the validity of functions alleged or implied by the developer.

The advanced evaluation exercises are intended to develop an

understanding of the software beyond that which can be gained by examining the verification examples. During the process of describing the program architecture and functions, the evaluator may identify features and characteristics that should be explored further, perhaps through pathological examples. The advanced evaluation exercises are intended to address this need. The exercises themselves can be divided into three groupings, depending upon the module to be investigated--the pre-processor module, the analysis module, or the post-processor module. A series of typical examples are offered here to point out the methodology to be used. For the pre-processor module, these examples will be lumped under the general category of discretization checks. Although the use of automatic mesh generation has led to an easing of the burden of data preparation and an elimination of many associated errors, modern mesh generators should have an additional capability--they should be designed so that the "condition" of the mesh is evaluated and automatically altered, if necessary. The condition of the mesh is related to the acuity of the vertex angles of the individual elements, which then determines the possible energy states and convergence characteristics of a particular mesh. Pathological examples involving reentrant corner geometries and graded mesh interfaces can be used to test these features of a mesh generator.

For the post-processor module, the examples will be lumped under the heading energy checks. Such checks, together with carefully selected benchmark problems, will expose errors due to deformation incompatibility. Also, an energy check can provide information on the convergence of nonlinear problems (stress states not satisfying flow criteria in plasticity, out-of-balance forces due to geometric nonlinearities or creep deformation, etc.).

Also, these energy checks can provide a global measure of the effort expended by the mesh to deform in accordance with applied forces. These energy checks are broken up into two categories: (1) internal/external - where the total (or incremental) internal energy is compared to the total (or incremental) external energy; for elements that lose energy (but often appear to converge to exact solutions with small numbers of elements), selected multi-element models should serve to expose such incompatible deformation behaviour; and (2) internal energy hierarchy - through selective volume integration (centroidal, two-point Gaussian, three-point Gaussian, etc.), the mesh effort can be determined, based on the energy partition between constant-straining modes and higher-order element deformation modes. Nickell [9] has examined this concept for two-dimensional configurations.

For the analysis module, there are several classes of examples designed to address questions of convergence, efficiency, and general capability. After "pseudo-convergence" due to deformation incompatibility has been eliminated, the element libraries should be tested with respect to real convergence. Similar element libraries should converge similarly; however, some errors in formulation may be exposed at this level. Also, convergence of transient and nonlinear solution algorithms, as well as eigenvalue/eigenvector extraction routines, should be examined. Particular topics of concern are:

1. Convergence rates of elements should be determined to be correct (bounds are known a priori),
2. Convergence rates and regions of convergence of Newton-Raphson, modified Newton-Raphson, Picard iteration, or other nonlinear solution algorithm should be tested and deemed to be correct;
3. Transient solution algorithms should be tested for stability

and artificial propagation properties; and

4. Eigenvalue/eigenvector extraction routines should be evaluated for multiplicity, separability, deterioration, and convergence.

IV. EVALUATION RESULTS

The three codes selected for evaluation--ADINA, NASTRAN, and STAGS--demonstrate the diversity of the origins of structural mechanics software. ADINA (Automatic Dynamic Incremental Nonlinear Analysis) is an outgrowth of the SAP family of codes from the University of California, Berkeley. Strictly speaking, it is a derivative of the nonlinear research code, NONSAP, extensively modified at the Massachusetts Institute of Technology by one of the original NONSAP authors--K. J. Bathe. ADINA retains much of its university research orientation. NASTRAN (NASA STructural ANalysis), on the other hand, was developed through a government procurement process, with project management, specifications, and all the rest of the trappings that accrue to software originating from a well-defined (at least, at the project outset) environment. The original version is attributable to the Computer Sciences Corporation, with MacNeal-Schwendler, Martin Baltimore, and Bell Aero Systems as subcontractors. The STAGS series of codes, including the version evaluated by the ISEG effort (STAGSC-1), were developed in a contract research environment at the Lockheed Palo Alto Research Laboratory. Because the funding to develop the code was subject to the vagaries of contract research cycles, STAGS would be expected to be of less uniform quality than software developed under sustained sponsorship.

ADINA was evaluated by Professors T. Y. Chang and J. Padovan of the University of Akron, using the 1977 version of the code. Their two-volume report [10, 11] divided the results of the evaluation into descriptive

material and advanced evaluation exercises (called operating characteristics by the authors). The descriptive material was subdivided into a general description, a discussion of the theoretical basis of the code, and a programming description. The latter item will prove to be a most valuable contribution to the organizations that use ADINA as the basis for a production structural analysis tool, especially Appendix A (Flow Diagrams for Various Solution Phases) and Appendix B (Dynamic Allocation of Arrays in the Blank Common Block). This information will also be invaluable to the developer who, although obviously aware of needed improvements, can use the programming constructions of the third part[illegible] a planning device. The major findings of this phase of the evaluation were:

- The documentation was [illegible] good to excellent--the user's manual, theoretical manual, and the coding practice (including liberal use of comment cards) were all given good marks;
- For a relatively small general purpose finite element program, ADINA has good capability to treat production analyses and the efficient coding practices employed enhance that capability;
- The principal deficiencies of the code are the lack of comprehensive pre- and post-processing capability,* a shortage of element types in the element library, the lack of a structured data base management system, and a limited capability with respect to large deformation problems.

Some of the criticisms of ADINA offered by the evaluators in their examination of operating characteristics could be directed at virtually all nonlinear

---

* Users may find it advantageous to examine stand-alone pre- and post-processors, such as GIFTS or PATRAN, with the potential to interface with ADINA, which then becomes the analysis module only.

general purpose finite element codes. For example, the modified Newton-Raphson incremental solution method for the nonlinear equations (i.e., tangent stiffness with occasional reformation) can be criticized for a number of situations—such as local unloading in an elastic-plastic analysis. A full Newton-Raphson (i.e., reformation at every iteration of every load step) may be prohibitively expensive. If equilibrium load correction is available as an option, the user may simply have too many solution control parameters to consider. Therefore, a nonlinear solution algorithm should automatically take advantage of the information available at each step/iteration to adjust the load increment (static analysis) or the time step (dynamic analysis), either increasing or decreasing, depending upon equilibrium or momentum error residuals. Such residuals at successive iterations indicate the current convergence status.

The advanced evaluation exercises for the ADINA project tended to concentrate on this aspect of nonlinear analysis, but the extensive benchmarking did produce some praise for the ADINA eigenvalue/eigenvector extraction algorithm. This volume, in addition to a rather complete treatment of convergence properties of the modified Newton-Raphson method for both plasticity and large deformation problems, also contains results for the direct time integration operators contained in ADINA, including an examination of the differences for mild, moderate, and strong nonlinearities. Adaptive solution strategies are offered as an alternative to strict modified Newton-Raphson methods at the close of the report.

The NASTRAN evaluation was understood at the outset by the ISEG to be a difficult undertaking for a number of reasons. First of all, the Level 17.5 Version of COSMIC/NASTRAN selected for the evaluation is not widely used,

since a number of commercial versions (e.g., MacNeal-Schwendler Corporation's MSC/NASTRAN and Universal Analytics Inc. UAI/NASTRAN) have drawn users away by virtue of improved support. Second, the code is so large and so opaque to the user (and to the third-party evaluator) that a description of the programming architecture has to be somewhat superficial. In spite of these difficulties, COSMIC/NASTRAN is so widely used that its inclusion in the initial ISEG effort was assured.

The Swanson Service Corporation was selected as the evaluation contractor, and the report describing the evaluation [12] was issued in August 1980. In addition to the evaluation criteria cited previously, Dr. J. William Jones and his associates produced an instructive user survey, which helped to identify those areas to be stressed during the evaluation itself. The remainder of the report followed the evaluation criteria precisely—including sections on NASTRAN documentation, program architecture, the functional description, verification exercises, and an excellent section of advanced evaluation exercises.

COSMIC/NASTRAN documentation was obviously designed to meet requirements similar to the evaluation criteria, since the NASTRAN Theoretical Manual, the NASTRAN Programmer's Manual, the NASTRAN User's Manual, and the NASTRAN Demonstration Manual are available. However, this set of manuals was rated poor, probably because of inadequate attempts to maintain and update these manuals during the life of the program. Instead, it appears to have been cost-effective to issue a separate NASTRAN User's Guide with each new Level, which supplies in a single volume the information necessary to use the program effectively.

The evaluation of the program architecture turned out to be less

superficial than anticipated. The gross program architecture is well described, with a heavy concentration on input/output and data base management, which is of primary interest to the readers. Of particular interest are Tables 4-1 and 4-2, which illustrate the characteristics of NASTRAN on three main frames—IBM360/370 series, UNIVAC 1108/1110 series with the EXEC 8 operating system, and CDC CYBER 6000 series with NOS--and which compares NASTRAN gross program architecture with MARC, ADINA, AGGIE I, and ANSYS.

The section on functional description is concisely written, considering the wide variety of options available to the NASTRAN user. Beginning with a brief discussion of rigid formats (of which there are 20 in Level 17.5 of COSMIC/NASTRAN), the report goes on to describe the macro programming language of NASTRAN, called DMAP (Direct Matrix Abstraction Program), which enables the skilled user to define additional analysis formats not contained within the set of 20. An example might be an acoustic-structural transient analysis. The geometric library, including elements and constraints, and the rather limited constitutive library are described next, with Table 5-2 providing a summary of elements and their characteristics. The procedures library is dominated by eigenvalue/eigenvector extraction methods and time integration schemes, which are tabulated and compared to other general purpose codes (e.g., MARC, ADINA, AGGIE I, ANSYS, STARDYNE, SAP VI, EASE 2, SUPERB, and MSC/NASTRAN) in Table 5-3. Plotting, restarting, nonlinear capabilities, and substructuring are also addressed. Extensive use of tables enable the reader to grasp the information handily.

The highlight of the report is the section on advanced evaluation. Unlike ADINA, which has a limited element library but an extensive nonlinear

capability, COSMIC/NASTRAN contains a wide assortment of element types. Many of these elements were selected for convergence studies, beginning with single-element checks, then to multiple-element benchmarks, to the patch test for those elements displaying any pathology (e.g., non-monotonic convergence), and element stiffness eigenvalue extraction when "false" deformation modes were suspected. The examination of the plate bending element QDPLT is a classic case of step-by-step investigation of non-conforming behaviour, even though it performed satisfactorily on most benchmark problems. The element TRSHL was also singled out for criticism, and its eventual removal from the COSMIC/NASTRAN element library is foreseen.

In this same section, a comparison of three three-dimensional elements (HEXA1, HEXA2, and IHEX1), in terms of their respective stiffness matrix eigenvalue spectra, offered the evaluators an opportunity to also compare a number of eigenvalue/eigenvector options available to the COSMIC/NASTRAN user.

Section 7.9 offers a discussion of pre- and post-processors, ostensibly for NASTRAN, but readers may find the topic of general interest, irrespective of the code being considered. For example, the ADINA code was cited for its lack of adequate pre- and post-processing capability, as well, and many of the commercial pre- and post-processing packages would be applicable to both ADINA and NASTRAN.

A short section on solution efficiency confirmed the intuitive observation that COSMIC/NASTRAN is costly for small problems, but becomes more cost-effective as the problems increase in size.

Among the numerous conclusions and recommendations offered by the evaluators, two should be emphasized: (1) the NASTRAN program architecture is flexible, but outdated, and major changes in the architecture to bring it up

to date would be prohibitively expensive and unjustified from a cost-benefit point of view; (2) the COSMIC/NASTRAN version should continue to be maintained for public use, since the code continues to be popular with many analysts, especially in the armed service R & D laboratories. This latter point should be reexamined periodically, however, since analysts are switching to the commercial versions of the code in sufficient numbers to cause concern about the continued viability of the public version.

The STAGSC-1 code was evaluated by Dr. Kevin Thomas and Dr. Larry H. Sobel of the Westinghouse Electric Corporation's Advanced Reactors Division in Madison, Pennsylvania. Their report [13], issued in August 1981, also follows the prescribed evaluation criteria precisely--beginning with a section on STAGS documentation, followed by sections on program architecture, functional description, verification exercises, and advanced evaluation. Since the STAGS series of codes are basically for special purpose shell analysis with an emphasis on nonlinear geometric and material behavior, the evaluation was relatively less concerned with element convergence (in contrast to NASTRAN) while concentrating more heavily on the performance of the procedure library (e.g., eigensolution and transient response performance).

The section on program documentation is extremely thorough, resulting in the conclusions that: (1) the theoretical manual is obsolete, having not been updated since the code was converted to finite element from a finite difference formulation; (2) a useful problem demonstration manual does not exist, since the problems given are also based upon the earlier finite difference version of the code; and (3) no programmer's manual exists. In fairness, however, the user's manual received a good-to-excellent rating and, similar to the NASTRAN Version 17.5 User's Guide, was deemed sufficient to

permit a user to work problems effectively.

The section on program architecture was also excellent, concentrating on the overlay structure, including the graphical representation in Figures 3.1 to 3.4. The discussion of the data management system could have been enhanced by some qualitative comparisons with the data management procedures in other finite element codes.

The section on functional description gives the reader some insight into the reason for the popularity of STAGS, especially for advanced aerospace and defense structural analysis. Although STAGS is a special purpose shell code, its procedure library together with a wide variety of built-in shell geometries, makes it attractive for nonlinear applications. The code contains five options for transient integration—central difference, trapezoidal rule, Gear's 2nd and 3rd order stiffly stable, and the Park averaging method. The element library contains truss and beam elements, in addition to a wide assortment of membrane, plate, and shell elements. Two other features that make STAGS attractive for aerospace/defense applications are the types of wall construction permitted (e.g., multiple anisotropic layers, walls reinforced by a corrugated skin, smeared stiffeners, etc.) and the ability to treat initial imperfections for an instability analysis.

Following a short section on verification, which was aimed at ensuring that three selected demonstration problems could be solved correctly on the Westinghouse CDC-7600 system, the advanced evaluation exercises were described. Four areas were investigated--element convergence, eigenvalue extraction, transient integration, and features of the nonlinear solution scheme. This section is again excellent, producing results that are invaluable to potential users of the code. The convergence study pointed out

some problems with the 420 series of shell elements, and provided some comparisons between the 410 series of shell elements and elements in other codes, such as ABAQUS, MARC, COSMIC/NASTRAN, and MSC/NASTRAN. The element library, in fact, seems to be a major disappointment for a code dedicated to shell analysis, since the elements in the general purpose finite element programs appear to be superior. Another disturbing observation was made during the eigenvalue/eigenvector extraction exercises, where spurious mode shapes corresponding to identical eigenvalues were calculated. While the reason was not identified precisely, it appeared to be associated with element characteristics, rather than with the eigensolution procedure.

Exercises using the transient integration operators seem to demonstrate that the trapezoidal rule was the most effective for both linear and nonlinear examples. The central difference operator did not perform efficiently because of the relatively small size of the examples.

An interesting conclusion was drawn as the result of geometrically nonlinear collapse load analyses of a point-loaded venetian blind, a pinched cylinder, and a poked cylinder. STAGS permits the user to control the nonlinear solution convergence through parameters that measure error within a load or displacement step, and that establish the frequency of refactoring. The evaluators recommend a simple computational procedure for use with a displacement-controlled analyses of "softening" structures that are near the collapse load.

Finally, program efficiency for STAGS was studied by comparing computer resource statistics on a variety of problems--linear and nonlinear static analyses, linear and nonlinear dynamic transient analyses and nonlinear collapse analyses—with those generated by the MARC general purpose code and

other structural codes available to Westinghouse. Based upon the ratio of computer resource units (CRU) to central processor (CP) hours, STAGS used more total resources per CP hour than MARC, WECAN, ANSYS, and PLACRE. On a direct comparison between MARC and STAGS on one particular problem, however, the total resources used by MARC exceeded those used by STAGS by a factor of three. These apparently contradictory results imply that program configuration (i.e., central memory allocated, backing storage arrangements) at execution dominates other considerations, and also indicates that MARC had been optimized with respect to the charging algorithm while STAGS had not.

Overall, the evaluators rated the code acceptable to good, with the major deficiencies being the documentation, the plasticity models, the element library, and the post-processor. In order for the program to continue to be viable, these deficiencies should be corrected.

V. CONCLUSIONS AND RECOMMENDATIONS

The major purpose of this report is to examine the evaluation criteria together with the resulting evaluations of ADINA, NASTRAN and STAGS in order to determine their adequacy. Based upon the work of the evaluation contractors—Professors T. Y. Chang and J. Padovan of the University of Akron, the Swanson Service Corporation, and the Westinghouse Advanced Reactors Division--the original evaluation criteria seem to have stood the test of application well. In retrospect, somewhat less emphasis should be placed on program architecture.* The most pleasant surprise was the information gathered during the advanced evaluation exercises. This can be attributed to

---

* The evaluators, in fact, placed the emphasis appropriately--by using a coarse description for the programming architecure and a finer description for the program functions.

the diligence of the evaluation contractors, who were able to translate minimal guidelines into effective action.

- For future studies it is recommended that the programming architecture description be limited to data management, data base structure, and gross program flow; a greater emphasis should be placed on the advanced evaluation exercises--which should be retitled Intrinsic Evaluation (the documentation, programming architecture, and program function evaluations should be sub-topics of the Extrinsic Evaluation); sub-topics under intrinsic evaluation would be element library convergence, transient operator characteristics, eigensolution characteristics, nonlinear solution convergence, program efficiency, and others.

A secondary purpose of this report is to discuss additional evaluation projects, should they be warranted. Based upon the ADINA, NASTRAN, and STAGS evaluations, it would seem appropriate to continue the ISEG effort, with one proviso. One of the major beneficiaries of an ISEG evaluation is the code developer, who is able to have access to informed third-party opinion. To some extent, evaluations of codes such as COSMIC/NASTRAN, which have a less than promising future, or ADINA, which was in the stage of being completely rewritten at the time of its evaluation, could be cost ineffective.

- For future studies it is recommended that the criteria for code selection be slightly altered to emphasize future potential, with somewhat less emphasis on frequency of use and multi-laboratory use; if this recommendation were to be adopted, likely candidates to be evaluated would be ABAQUS, the nonlinear general purpose finite element code being developed by Hibbitt, Karlsson &

Associates, and the upward-compatible microcomputer finite element code being developed by E. L. Wilson at the University of California; other candidates might be the Lawrence Livermore Laboratory code DYNA3D or the Sandia Laboratories code HONDO II.

A final concern of this report is the institutional form and the concentration of talent needed to conduct the evaluations. Normally, a source for this type of research would be educational institutions, one of which—the University of Akron—was used in this ISEG effort. It should be pointed out, however, that most university researchers consider the evaluation of foreign* software not fundamental and perilously close to teaching students how to use existing software, which is an anathematic. Outside the universities, and excluding the national laboratories and the software vendors themselves, the type of talent required—a blend between a developer and a skilled user—is relatively rare, but does exist. One of the positive features of a national engineering software center would be that, as an institutional form, it would be ideal to attract the talent and be the medium for continuing applications software evaluations. Even if the national engineering software center were unable to atttract such talent in sufficient numbers, the institution would be capable of training a new generation of critical evaluators within a short period of time.

. It is recommended that, as the discussion of the national engineering software center concept continues, one of its functions be to institutionalize the critical evaluation of engineering applications software.

---

* Foreign means, in this context, written outside the university or not in current use at the university.

REFERENCES

[1]. Numerical and Computer Methods in Structural Mechanics, edited by S.J. Fenves, N. Perrone, A.R. Robinson, and W.C. Schnobrich, Academic Press, New York (1973).

[2]. Tocher, J.L. and Herness, E.D., "A Critical Review of NASTRAN," in: Numerical and Computer Methods in Structural Mechanics, ed. by S.J. Fenves, et al., Academic Press, New York (1973), pp. 151-174.

[3]. Structural Mechanics Computer Programs : Surveys, Assessments, and Availability, edited by W. Pilkey, K. Saczalski, and H. Schaeffer, University Press of Virginia, Charlottesville (1974).

[4]. Nickell, R.E., "The Interagency Software Evaluation Group : A Critical Structural Mechanics Software Evaluation Concept," Report No. PT-U78-0246, Pacifica Technology, Del Mar, California (August 1978).

[5]. Matula, P., "Navy Engineering Software System (NESS) and Preliminary Selection of Computer Programs," TM-184-77-01, Naval Ship Research and Development Center, Bethesda, Maryland (October 1976).

[6]. Aerospace Structures Information and Analysis Center (ASIAC), Wright-Patterson Air Force Base, Ohio. The Air Force Flight Dynamics Laboratory sponsors ASIAC as a central agency to collect and disseminate information on aerospace structures, including structural software, to Air Force contractors and other government contractors.

[7]. Deutsch, M.S., "Software Project Verification and Qualification," Computer, pp. 54-70 (April 1981).

[8]. Henrywood, R.K., "The Design, Development, Documentation and Support of a Major Finite Element System," Computer Aided Design, Vol. 5, pp. 160-165 (July 1973).

[9]. Nickell, R.E., "Structural Mechanics Software Evaluation : A Bigeneric Diagnostic Framework," Applied Science & Technology, La Jolla, California (June 1980).

[10]. Chang, T.Y. and Padovan, J., "Evaluation of ADINA : Part I, Theory and Programming Descriptions," Report No. AUE-801, The University of Akron, Akron, Ohio (June 8, 1980).

[11]. Padovan, J. and Chang, T.Y., "Evaluation of ADINA : Part II, Operating Characteristics," Report No. AUE-802, The University of Akron, Akron, Ohio (June 8, 1980).

[12]. Jones, J.W., Fong, H.H., and Blehm, D.A., "Evaluation of the NASTRAN General Purpose Computer Program," SSC Report No. 81980, Swanson Service Corporation, Huntington Beach, California (August 1980).

[13]. Thomas, K. and Sobel, L.H., "Evaluation of the STAGSC-1 Shell Analysis Computer Program," Report No. WARD-10881, Westinghouse Electric Corporation, Advanced Reactors Division, Madison, Pennsylvania (August 1981).


FILMED